Universidad Autónoma
de Madrid

Escuela Politécnica Superior
Departamento de Ingeniería Informática
Universidad Autónoma de Madrid

# Exploring attributes, sequences, and time in Recommender Systems: From classical to Point-of-Interest recommendation

Dissertation written by

**Pablo Sánchez Pérez**

Under the supervision of

**Alejandro Bellogín Kouki**

2021 March

| | |
|---|---|
| **PhD thesis title:** | Exploring attributes, sequences, and time in Recommender Systems: From classical to Point-of-Interest recommendation |
| **Author:** | **Pablo Sánchez Pérez** |
| **Affiliation:** | Departamento de Ingeniería Informática<br>Escuela Politécnica Superior<br>Universidad Autónoma de Madrid, Spain |
| **Supervisor:** | **Alejandro Bellogín Kouki**<br>Universidad Autónoma de Madrid, Spain |
| **Date:** | March 2021 |
| **Committee:** | **Tommaso Di Noia**<br>Polytechnic University of Bari, Italy<br><br>**Juan de Lara**<br>Universidad Autónoma de Madrid, Spain<br><br>**Ludovico Boratto**<br>University of Cagliari, Italy<br><br>**Saúl Vargas**<br>Infogrid, Estonia<br><br>**Belén Díaz Agudo**<br>Universidad Complutense de Madrid, Spain |

# Abstract

Since the emergence of the Internet and the spread of digital communications throughout the world, the amount of data stored on the Web has been growing exponentially. In this new digital era, a large number of companies have emerged with the purpose of filtering the information available on the web and provide users with interesting items. The algorithms and models used to recommend these items are called Recommender Systems. These systems are applied to a large number of domains, from music, books, or movies to dating or Point-of-Interest (POI), which is an increasingly popular domain where users receive recommendations of different places when they arrive to a city.

In this thesis, we focus on exploiting the use of contextual information, especially temporal and sequential data, and apply it in novel ways in both traditional and Point-of-Interest recommendation. We believe that this type of information can be used not only for creating new recommendation models but also for developing new metrics for analyzing the quality of these recommendations. In one of our first contributions we propose different metrics derived from previously existing frameworks using this contextual information. We also propose an intuitive algorithm that is able to provide recommendations to a target user by exploiting the last common interactions with other similar users of the system.

At the same time, we conduct a comprehensive review of the algorithms that have been proposed in the area of POI recommendation between 2011 and 2019, identifying the common characteristics and methodologies used. Once this classification of the algorithms proposed to date is completed, we design a mechanism to recommend complete routes (not only independent POI) to users, making use of reranking techniques. In addition, due to the great difficulty of making recommendations in the POI domain, we propose the use of data aggregation techniques to use information from different cities to generate POI recommendations in a given target city.

Our experimental work presents our approaches on different datasets belonging to both classical and POI recommendation. The results obtained in these experiments confirm the usefulness of our proposals, not only in terms of recommendation accuracy, but also when novelty, diversity, and coverage (or adaptations of some of the metrics proposed in this work) are considered.

# Resumen

Desde la aparición de Internet y la difusión de las redes de comunicaciones en todo el mundo, la cantidad de datos almacenados en la red ha crecido exponencialmente. En esta nueva era digital, han surgido un gran número de empresas con el objetivo de filtrar la información disponible en la red y ofrecer a los usuarios artículos interesantes. Los algoritmos y modelos utilizados para recomendar estos artículos reciben el nombre de Sistemas de Recomendación. Estos sistemas se aplican a un gran número de dominios, desde música, libros o películas hasta las citas o los Puntos de Interés (POIs, en inglés), un dominio cada vez más popular en el que los usuarios reciben recomendaciones de diferentes lugares cuando llegan a una ciudad.

En esta tesis, nos centramos en explotar el uso de la información contextual, especialmente los datos temporales y secuenciales, y aplicarla de forma novedosa tanto en la recomendación clásica como en la recomendación de POIs. Creemos que este tipo de información puede utilizarse no sólo para crear nuevos modelos de recomendación, sino también para desarrollar nuevas métricas para analizar la calidad de estas recomendaciones. En una de nuestras primeras contribuciones proponemos diferentes métricas derivadas de formulaciones previamente existentes utilizando esta información contextual. También proponemos un algoritmo intuitivo que es capaz de proporcionar recomendaciones a un usuario objetivo explotando las últimas interacciones comunes con otros usuarios similares del sistema.

Al mismo tiempo, realizamos una revisión exhaustiva de los algoritmos que se han propuesto en el ámbito de la recomendación de POIs entre 2011 y 2019, identificando las características comunes y las metodologías utilizadas. Una vez realizada esta clasificación de los algoritmos propuestos hasta la fecha, diseñamos un mecanismo para recomendar rutas completas (no sólo POIs independientes) a los usuarios, haciendo uso de técnicas de reranking. Además, debido a la gran dificultad de realizar recomendaciones en el ámbito de los POIs, proponemos el uso de técnicas de agregación de datos para utilizar la información de diferentes ciudades y generar recomendaciones de POIs en una determinada ciudad objetivo.

Nuestro trabajo experimental presenta varios enfoques en diferentes conjuntos de datos tanto de recomendación clásica como de POIs. Los resultados obtenidos confirman la utilidad de nuestras propuestas, no sólo en términos de acierto de la recomendación, sino también cuando se consideran la novedad, la diversidad y la cobertura (o adaptaciones de algunas de las métricas propuestas en este trabajo).

# Acknowledgements

This thesis, which was the result of so many years of research, would not have been possible without the encouragement of many people, including my family, friends and colleagues. I would like to return part of the favor by dedicating the following lines to them.

Firstly, I would like to thank my supervisor Alejandro Bellogín for trusting me and offering me to work on this thesis with him. I must recognize that my interest in research began to grow more since he directed first my final degree work and my master's thesis. Nevertheless, I have to admit that when he offered me to do a PhD I was quite hesitant because I did not think I was ready for such a challenge. In the end, Alejandro was right and I managed to finish the PhD successfully. However, his supervision has been fundamental since he has always been attentive to my progress, helping me interpreting results, proposing new lines of research, preparing conference presentations... For these and many other reasons, I consider that this thesis also belongs to him. I would also like to thank Pablo Castells, the head of the Information Retrieval Group, and Iván Cantador because thanks to them I could conduct my PhD with a scholarship in their group. Without that research grant and the work environment they offered to me I would not be here today. Thank you also to Fernando Díez for his advice and for helping me integrate into the computer science department.

Of course, the other members of the Information Retrieval Group also deserve a special mention. Thanks to Javier Sanz-Cruzado and Rocío Cañamares for the support they have given me and for helping me to integrate in the group. It's been a privilege to have shared with them a lot of talks and discussions.

También, me gustaría hacer una dedicatoria especial a todos mis compañeros durante la carrera, el máster y el doctorado en la universidad. Muchas gracias sobre todo a Guillermo Sarasa y Alejandro Catalina, mis compañeros de prácticas a lo largo de la carrera y del máster con los que he aprendido muchísimo y con los que mantenía grandes charlas hablando de todo tipo de temas. Gracias también a todos los compañeros de la ex-asociación DEISI (ahora renombrabrada como CODE), por las tardes en las que íbamos al cine o a cenar para descansar un poco de la universidad. Por supuesto, no me olvido de mis amigos ajenos a la universidad: Sergio, Álvaro, Daniel, Anaïs, Belén... y los días que hemos quedado para comer o simplemente a tomar algo en los bares de Madrid. Gracias en especial a Mónica, a quien he conocido en plena pandemia de la COVID-19. A pesar de haber sido una etapa dura también para ella, me ha estado animando en todo momento, sacando huecos para vernos y hacer todo tipo de planes. Sin ella, la última etapa del doctorado hubiese sido mucho más difícil.

Por último, pero quizá lo más importante, quería agradecer por supuesto a mi familia por todo el apoyo que me ha brindado estos años. Muchas gracias a mis padres y a mi hermano y también quiero mencionar de manera especial a mis abuelos, aunque la mayoría no estén ya aquí. Gran parte de lo que soy como persona se lo debo a mi familia y por ello, a ellos va dedicado este trabajo.

# Contents

# List of Figures

## LIST OF FIGURES

# List of Tables

# Part I

# Context and Background

# 1

# Introduction

## 1.1 Motivation

In 2018, the company DOMO estimated that during the year 2020 each person on planet Earth would generate an average of 1.7MB of data every second (Ahmad, 2018). Currently, it is very likely that this quantity of data has increased substantially because the sanitary crisis of COVID-19 has forced a large number of people to spend more time at home and make a greater use of the Internet, both generating and consuming content either by leisure purposes or work.

In this context of exponential growth of information available on the web, the problem of information overload (Maes, 1994), where users may spend an excessive amount of time searching for the information they need, becomes even more evident. Recommender Systems (RS) arise to solve this problem. These software tools are oriented to filter the countless available items in a system to recommend the users those items that are more suitable to their needs based on their previous experiencies. Although the first search engines and Recommender Systems emerged in the 1990s, their use has spread in an unstoppable way over the recent years.

Nowadays, companies such as Google, Amazon, Youtube, Netflix, and many more make use of these technologies to increase the number of users of their platforms while offering personalized content to existing customers. Specifically, Recommender Systems have proven to be an area of research in high demand, specially since the appearance of the Netflix prize between 2006 and 2009 (Bell and Koren, 2007), where 1M dollars were offered to the research group that managed to improve the prediction of its baseline algorithm by a 10%. At the same time, international conferences dedicated to this topic (among which the ACM conference of Recommender Systems[1] stands out) increase the number of attendees every year, as well as the companies interested in sponsoring these congresses. However, Recommender Systems are not perfect by any means. Due to the massive data analysis performed by these algorithms, users have become increasingly aware of aspects such as privacy, intrusiveness, or the explanation of the recommendations (Ricci et al., 2015). Some of these problems are indeed major challenges in the

[1]ACM Conference on Recommender Systems, RecSys, `https://recsys.acm.org/`

Recommender Systems community, however they are beyond the scope of this thesis.

The high degree of adaptability of Recommender Systems enables them to be applied in many different areas like movies, books, music, tourism, or even in dating apps (Ricci et al., 2015). Nevertheless, it is important to mention that each domain has its own particularities. For example, music and movies, which are well known recommendation domains, have substantial differences: the catalog of movies is normally more reduced than the catalog of songs, the music domain has a strong sequential component, while failing a recommendation in the music domain is not too critical as songs are usually less than 5 minutes long, whereas failing a movie or video recommendation may affect more the users as they tend to get frustrated more easily, etc. (Schedl et al., 2018, Jannach et al., 2018). Besides, there are domains that need to incorporate additional sources of information to make useful recommendations. In this sense, assistants such as Alexa (Amazon), Google Assistant, or Siri (Apple) might be particularly useful as they store more information about the users like their tastes, their current location, the time, or even the weather, among other data.

An important area where all this type of information can be applied is the tourism domain, which has a great economic impact on both tourists and the regions they visit. For example, in countries like Spain, Iceland, or Mexico, the percentage of the total Gross Domestic Product (GDP) associated with tourism is higher than 8%[2]. There are a large number of recommendation tasks related to tourism, including route (or trajectory) and group recommendation, but perhaps the best known and most studied one is the Point-of-Interest (POI) recommendation problem, where the items to be recommended are interesting venues or places for the user to visit when they arrive to a city (hotels, bars, restaurants, museums, etc.) (Ye et al., 2011). In this type of recommendation, Location-Based Social Networks (LBSNs) such as Foursquare, Yelp, or Gowalla (see Figure 1.1) are specially relevant as in these social networks the users can register the check-ins they make on the venues they visit and exchange information with the rest of the users in the system (Wang et al., 2013). In fact, the research in this domain has increased in recent years due to several reasons, including the growing number of people that can afford to make trips to different cities, the transport infrastructure improvement, and the facility to access high-level technology (e.g., high-speed networks or more advanced mobile phones).

Initially, the first recommendation strategies only used interactions between users and items to make recommendations. However, in recent years, the use of contextual information has become particularly useful as it allows the algorithms to better adapt to the interests of the users in certain situations. This contextual information can be very varied, including temporal and/or sequential information, weather, popular trends, etc. (Adomavicius and Tuzhilin, 2015, Villegas et al., 2018). It is important to take into account that users may consume particular items depending on the current situation. For example, a user may watch different types of movies depending on whether she is alone or accompanied. The same reasoning applies in the music domain as a user may

[2]Organization for Economic Co-operation and Development, OECD, `https://www.oecd.org/cfe/tourism/OECD-Tourism-Trends-Policies2020-Highlights-ENG.pdf`

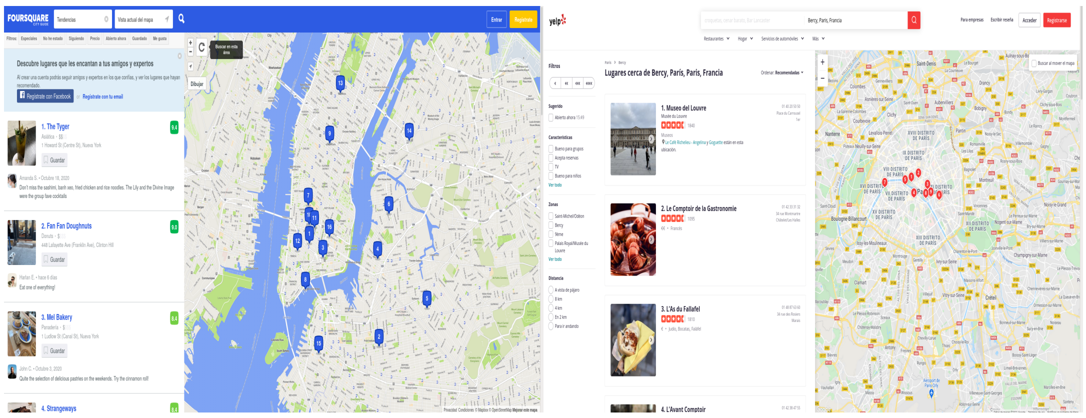


**Figure 1.1:** Example of two LBSNs. Foursquare (left) and Yelp (right).

listen to different songs depending on if she is at work or with friends one Friday night. In tourism, this contextual information is also important since the types of POIs a user visits may be affected by the weather (i.e., in summer it is more feasible to do outdoor activities), time of the day (e.g., not recommending a bar early in the morning), or the geographical distance (Laß et al., 2017).

On the other hand, while research on how to generate better recommendations is important, it is also essential to design mechanisms to evaluate the models. In fact, Recommender Systems evaluation has changed over successive years. Error prediction metrics oriented at measuring the difference between the predicted and real ratings were initially used to analyze the performance of the recommenders but, over the past few years, the evaluation of algorithms has evolved through the use of ranking-based accuracy metrics from the Information Retrieval (IR) area, where the objective is to predict a list of hypothetically interesting items for the user (Steck, 2013, Gunawardana and Shani, 2015). However, this type of analysis, although useful, is incomplete since it is important from the user point of view to also measure the quality of the recommendations in terms of other complementary dimensions such as novelty, diversity, or serendipity (Castells et al., 2015). Besides, recently there has been an increased awareness about providing fair recommendations to the users by avoiding possible counterproductive biases in the algorithms, such as performing higher quality recommendations to specific social groups because of their age, gender, nationality, etc. (Steck, 2018, Abdollahpouri et al., 2019a). For these reasons, one of the goals in the community is to develop algorithms with a good balance between these dimensions and accuracy – trying to obtain accurate techniques while being as fair as possible – although it is generally assumed that it is difficult, or even impossible, to develop an algorithm that can beat any other technique in all possible aspects. Furthermore, currently there is a growing concern about the reproducibility of the performance obtained by the models because it is often not possible to replicate the results reported by the original authors of the papers (Beel et al., 2016, Dacrema et al., 2019). In consequence, for some years now, an attempt has been made to promote the publication of the code of the models developed and, at the same time, more attention has been paid to different aspects of the evaluation process.

Thus, in this thesis we analyze some of the aforementioned issues by proposing solutions in the form of algorithms and metrics to further explore the use of contextual information in recommendations. Although the developed proposals can be used in several recommendation domains, we will make a special emphasis on the POI recommendation problem, since it is a growing area that can benefit from the use of this type of information to alleviate some of its fundamental problems (such as the great sparsity of the data). In our experimental work we demonstrate the usefulness of our proposed approaches as, on the one hand, our derived metrics allow us to obtain a more complete analysis of why algorithms make certain recommendations, and on the other hand, our contextual recommendation models allow us to improve the performance of the algorithms in both accuracy and complementary dimensions such as novelty, diversity, or freshness.

## 1.2 Research Goals

This thesis has two major purposes. On the one hand, we claim that modeling contextual information is important in order to improve the performance of the algorithms and, hence, we investigate how to incorporate contexts like sequentiality or time in classical Recommender Systems. Besides, we argue that these contexts can also be integrated in the evaluation phase by building new metrics in order to detect possible biases in the recommendations produced. Moreover, we focus on the Point-of-Interest recommendation problem by studying the main issues and challenges of the area while proposing solutions to palliate them. In this regard, we demonstrate that we can generate meaningful routes to the users exploiting reranking approaches whereas we can also use cross-domain techniques to improve the performance of the recommenders. Hence, taking these ideas as a starting point, in this thesis we propose the following research goals (RG):

**RG1: Review the state-of-the-art on POI Recommender Systems to identify and characterize the most important works in the area.** Most of the surveys in the POI domain focus on analyzing the types of algorithm that are used in the models and the type of information they exploit. However, an important aspect that is usually overlooked in such surveys is the evaluation methodology and/or the datasets and metrics used. We therefore intend to create a survey to analyze all these aspects in depth in order to detect how comparable are the state-of-the-art POI algorithms.

**RG2: Study evaluation metrics for classical recommendation to adapt and integrate additional dimensions beyond relevance.** When analyzing the performance of a recommender, most researchers focus on how well the recommendation system works in terms of ranking accuracy. Although additional dimensions such as novelty and diversity have been taken into account when evaluating the recommenders, we believe that it is important to incorporate other dimensions such as sequentiality, time, and anti-relevance (items that users have specifically indicated they do not like) in the metrics. We hence plan to develop different metrics to take into account these additional dimensions and analyze the results of the algorithms in these new metrics.

**RG3: Develop mechanisms to add sequentiality in neighborhood-based recommender systems.** Currently, in the area of Recommender Systems, a large number of models have been proposed to incorporate contexts such as sequentiality or time in the recommendations produced. Popular methods, such as Neural Networks and different derivations of Markov Chains, are used nowadays, although in some cases it is difficult to correctly interpret the recommendations they provide. Therefore, we aim to investigate the feasibility to incorporate the sequential context into neighborhood-based algorithms since they are easier to understand and develop than the aforementioned models.

**RG4: Explore the LBSNs data used in POI recommendation to explore new ways to make recommendations and enable full route recommendation.** POI recommendation has traditionally been modeled as a recommendation of relevant but independent venues to the users. However, there is a clear sequential relationship between the different venues that the users have checked-in (e.g., if they follow a trajectory). In this sense, we intend to exploit the information stored in LBSNs and investigate how to obtain routes from user data using simple techniques already explored in the area of Recommender Systems.

**RG5: Improve the performance of the algorithms in the POI recommendation domain.** The POI recommendation problem has specific considerations that must be taken into account when making recommendations. The survey we will conduct in RG1 will allow us to clearly establish these considerations in order to be able to propose new mechanisms to improve the recommendations produced.

## 1.3 Contributions

The work done in this thesis has contributed to the current state-of-the-art of both classical and POI Recommender Systems. The contributions include: a systematic categorization of different types of POI recommendation algorithms, the definition of new metrics in order to analyze the performance of the recommendation models in various dimensions, the definition of new algorithms for classical recommendation that exploit the user context, and the analysis of routes and biases in the data found in Location-Based Social Networks.

First, in **Chapter** 3 we focus on the POI recommendation domain and we conduct a survey analyzing the most important algorithms of the state-of-the-art between 2011 and 2019. In this survey, apart from analyzing the type of information (geographical, content, collaborative, etc.) and algorithms (social, deep learning, factorization, etc.) used in the analyzed approaches, we also examined the most frequent evaluation methodologies followed by the researchers. In this aspect, we determine the most commonly used datasets as well as metrics and types of splits in order to determine how comparable are all these works with each other. Our review on the current POI approaches has been submitted to the following journal:

- **Pablo Sánchez** and Alejandro Bellogín. (2020). Point-of-Interest Recommender Systems: A Survey from an Experimental Perspective. Submitted to *ACM Computing Surveys*. Under Review (1st round of review). **Impact Factor 2019:** 7.990. **JCR 2019:** Q1: 4/108. Computer Science Theory & Methods.

Secondly, in **Chapter** 4 we propose new metrics that incorporate different contexts for evaluating the recommenders. First, we exploit temporal information to determine if an item is novel or not depending on the specific moments in which it was consumed by the users in the system. Secondly, we adapt the Probabilistic Ranking Principle in order to define new metrics that measure how many "anti-relevant" (items with a very low rating) recommendations are made by the algorithms. This is a novel proposal because, even though it is important that a recommender makes good recommendations, it is equally important that the recommender does not suggest items the users have specifically categorized as "bad". After that, we also define metrics that exploit the attributes of the users and items. The item attributes allow us to determine if a recommendation list is better than other depending on how the attributes of the recommended items match the ones of test set. On the other side, the user attributes allow us to determine if the recommneders are providing recommendations of the same quality to different groups of users. Finally, we also modify the traditional ranking-based accuracy metrics from Information Retrieval (IR) to take into account not only the relevance of the recommendations but also the order with respect to the test set of the user, to check if the recommended items follow a sequence. The publications related to this chapter are the following:

- **Pablo Sánchez** and Alejandro Bellogín. Time-aware novelty metrics for recommender systems. In Gabriella Pasi, Benjamin Piwowarski, Leif Azzopardi, and Allan Hanbury, editors, *Advances in Information Retrieval - 40th European Conference on IR Research*, ECIR 2018, Grenoble, France, March 26-29, 2018, Proceedings, volume 10772 of Lecture Notes in Computer Science, pages 357-370. Springer, 2018. **CORE 2018:** A. **Acceptance rate (long papers):** 23%. The code for this paper is available in the following url[3].

- **Pablo Sánchez** and Alejandro Bellogín. Measuring anti-relevance: a study on when recommendation algorithms produce bad suggestions. In Sole Pera, Michael D. Ekstrand, Xavier Amatriain, and John O'Donovan, editors, *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys 2018, Vancouver, BC, Canada, October. 2-7, 2018, pages 367-371. ACM, 2018. **CORE 2018:** B. **Acceptance rate (short papers):** 25%. The code for this paper is available in the following url[4].

- **Pablo Sánchez** and Alejandro Bellogín. Attribute-based evaluation for recommender systems: incorporating user and item attributes in evaluation metrics. In

[3]Time-aware novelty metrics, `https://bitbucket.org/PabloSanchezP/timeawarenoveltymetrics`
[4]Anti-relevance metrics, `https://bitbucket.org/PabloSanchezP/antirelevancemetrics`

Toine Bogers, Alan Said, Peter Brusilovsky, and Domonkos Tikk, editors, *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019., pages 378-382. ACM, 2019. **CORE 2018:** B. **Acceptance rate (short papers):** 24%. The code for this paper is available in the following url[5].

After that, in **Chapter** 5 we adapt a subsequence matching algorithm to measure the similarity between the users in a system and integrate this similarity in a recommendation model. As this algorithm is designed to work with sequences, it allows us to extend the basic formulation with both temporal and content information in a simple way. At the same time it is also suitable to work with repeated interactions, that is, users consuming the same item more than once. This is something useful in domains like music or POI recommendation where users tend to consume or visit the same item more than once. Besides, we have proposed a new recommendation algorithm based on neighbors in which candidate items are selected exploiting the last interaction that the neighbor has with the target user, in order to generate recommendations that take into account temporal and sequential information. The main publication related to this chapter is:

- **Pablo Sánchez** and Alejandro Bellogín. Time and sequence awareness in similarity metrics for recommendation. *Information Processing Management*, 57(3):102228, 2020. **Impact Factor 2019:** 4.787. **JCR 2019:** Q1: 22/156. Computer Science, Information Systems. The code for this paper is available in the following url[6].

This article is influenced by and can be understood as the future work of these previous publications:

- **Pablo Sánchez** and Alejandro Bellogín. Building user profiles based on sequences for content and collaborative filtering. *Information Processing Management*, 56(1):192-211, 2019. **Impact Factor 2019:** 4.787. **JCR 2019:** Q1: 22/156. Computer Science, Information Systems.

- Alejandro Bellogín and **Pablo Sánchez**. Collaborative filtering based on subsequence matching: A new approach. *Information Sciences*, 418:432-446, 2017. **Impact Factor 2017:** 4.305. **JCR 2017:** Q1: 12/148. Computer Science, Information Systems.

Continuing the work on sequences, in **Chapter** 6 we also define mechanisms to obtain routes or trajectories of POIs, not just independent venues, using the data of LBSNs. Hence, we define a method to obtain and filter routes from POI interactions datasets and we then explore how to recommend routes applying and adapting reranking techniques in the area of POI recommendation. The publication related to this chapter is:

[5]Attribute-based evaluation, `https://bitbucket.org/PabloSanchezP/attreval4recsys`
[6]Time and sequence awareness, `https://bitbucket.org/PabloSanchezP/bfrecommendation`

- **Pablo Sánchez** and Alejandro Bellogín. Applying reranking strategies to route recommendation using sequence-aware evaluation. *User Modeling and User-Adapted Interaction*, 30(4):659-725, 2020. **Impact Factor 2019:** 4.682. **JCR 2019:** Q1: 4/22. Computer Science, Cybernetics. The code for this paper is available in the following url[7].

Finally, due to the great sparsity of the POI recommendation area, in **Chapter** 7 we apply techniques based on cross-domain to determine if it is possible to improve the recommendations produced by the algorithms both in terms of accuracy and other dimensions like novelty, diversity, and coverage. At the same time, we classify the users in two different groups, tourists and locals, and we analyze the effect of these techniques in both groups separately. The submitted publications related to this chapter are:

- **Pablo Sánchez** and Alejandro Bellogín. On the effects of aggregation strategies for different groups of users in venue recommendation. Submitted to *Information Processing and Management*. Under Review (2nd round of review). **Impact Factor 2019:** 4.787. Q1: 22/156. Computer Science, Information Systems. The code for this paper is available in the following url[8].

- **Pablo Sánchez** and Alejandro Bellogín. A novel approach for venue recommendation using cross-domain techniques. In the 2nd Workshop on Intelligent Recommender Systems by Knowledge Transfer & Learning (RecSysKTL), held in conjunction with the 12th ACM Conference on Recommender Systems, Vancouver, Canada, Oct, 2018.

## 1.4 Structure of the Thesis

This thesis is structured as follows:

- Chapter 2 presents the state-of-the-art of classical Recommender Systems. First, we define the recommendation problem and categorize the most popular types of models, highlighting their advantages, disadvantages, and the most representative approaches. Besides, we show the evolution on the evaluation methods and protocols that have been used in the area and we define the most well-known evaluation methodologies and metrics currently used.

- Chapter 3 discusses about the POI recommendation problem and categorize the most important approaches in this domain between 2011 and 2019. In our survey, we classify the models according to the type of information used (temporal, sequential, content, etc.), the type of algorithm (similarities, factorization, neural

[7]Applying reranking strategies to route recommendation, `https://bitbucket.org/PabloSanchezP/serersys`

[8]Aggregation strategies for venue recommendation, `https://bitbucket.org/PabloSanchezP/tempcdseqeval`

networks, etc.), where we also make a special emphasis on the reproducibility aspect, showing the main evaluation strategies used in the area (metrics, datasets, types of splits, etc.).

- Chapter 4 proposes a set of new metrics to incorporate additional contexts when evaluating Recommender Systems. First, we extend an existing novelty and diversity framework to incorporate temporal information. Secondly, we show how to take into account the items that are specifically not liked by the user in evaluation. Later, we show how we can use the item attributes in evaluation; finally, we show how to incorporate sequentiality in traditional ranking-based accuracy metrics.

- Chapter 5 proposes a new similarity metric based on the Longest Common Subsequence (LCS) algorithm to be used in neighborhood-based recommenders. Besides, it also presents a reformulation of user-based neighborhood algorithms bringing ideas from ranking fusion techniques.

- Chapter 6 continues exploring the data of Location-Based Social Networks and proposes mechanisms to obtain full routes from these types of datasets. Besides, it also presents mechanisms to obtain routes from independent venues using reranking techniques from the Information Retrieval (IR) area.

- Chapter 7 further investigates the problem of traditional POI recommendation with special emphasis on the high sparsity of this domain. In order to try to partially alleviate this effect, we propose the use of data aggregation techniques (based on the cross-domain area) to improve the recommendations in independent cities by augmenting the information from other cities based on different strategies.

- Chapter 8 summarizes the main conclusions of this thesis and discusses the main limitations and the future work for a further investigation.

# 2

# Recommender Systems

Recommender Systems (RS) are software tools that analyze the users' preferences (i.e., opinions, reviews, tastes) and suggest different items that fit into the users' interests in a personalized way. These items depend on the domain where the recommendation engine is applied. For example, in companies like Netflix or Youtube, the items would be videos while in Spotify, the items would be songs, artists, or both, and in Amazon, the items might be any available product from its catalog. In all these companies, the information that the user can access to is so immense that information filtering, personalized recommendation, and discovery of items provided by the recommenders are indispensable to help users and improve their satisfaction (Ricci et al., 2015, O'Donovan and Smyth, 2005).

In this chapter, we will make an overview of the traditional technologies and techniques that are used in the recommendation domain and introduce the reader to the terminology that will be used in the rest of the thesis. First, in Section 2.1 we show the formulation of the recommendation problem. Later, in Section 2.2 we present the most important classical types of Recommender Systems, and in Section 2.3 we make a more in depth study about context-aware recommenders. Finally, Section 2.4 illustrates the main mechanisms for analyzing the performance of the recommenders in different dimensions, from accuracy to novelty and diversity.

## 2.1 Definition of the recommendation problem

In the recommendation process, there are a noteworthy number of factors that can be taken into account in order to improve the recommendations produced to users (e.g., long term preferences, exploring new content, analyze popular items, etc.). Nevertheless, the main objective of every recommender is to maximize the usefulness of the items that are being suggested to the target user. Although several formulations have been proposed in the area for this problem, one of the most cited is the one introduced by Adomavicius and Tuzhilin (2005), where the recommendation task is presented as an optimization formula. Nonetheless, before showing such formulation, we will clarify hereafter part of the notation that we will use in this thesis. First of all, we will denote the set of users in the system with symbol $\mathcal{U}$ and the set of items with symbol $\mathcal{I}$. We define a utility function $g$: $\mathcal{U} \times \mathcal{I} \rightarrow \mathcal{R}$ that measures the hypothetical interest that user $u \in \mathcal{U}$ may have on item $i \in \mathcal{I}$. Thus, the objective is to provide to each user unknown items ordered by the utility function $g$:

$$i^*(u) = \arg\max_{i \in \mathcal{I}} g(u, i) \tag{2.1}$$

Normally, $\mathcal{R}$ refers to real numbers and aims to capture the usefulness of the items with respect to the users. The relationship between users and items are modeled using a $\mathcal{U} \times \mathcal{I}$ matrix in which each cell represents a value indicating if the user liked or not that item, with empty cells denoting the items that the user has not consumed (actually, the interactions the system is not aware of, since in the real world the users may consume or interact with the same type of items in different systems). Traditionally, this matrix is very sparse (more than 99% of the values are unknown). Although the objective of any recommendation algorithm is to suggest the maximum number of relevant items to the target user $u$ by approximating Equation 2.1, depending on how the user and item information are used, we can characterize several types of Recommender Systems. According to Ricci et al. (2015), we distinguish at least the following: 1) Content-based, that suggest to the users items similar to the ones they liked in the past. 2) Collaborative filtering, which analyze the interactions between users and items and establish patterns between them in order to make recommendations. 3) Demographic, that provide recommendations by exploiting the demographic information of the user (country, language, etc.). 4) Knowledge-based, that suggest items based on specific knowledge of the item features and how they meet the users' interests, normally taking into account additional inputs as queries or constraints of the users. All these techniques may have certain drawbacks in some circumstances. For example, for a collaborative filtering recommender it will be impossible to recommend items to a new user who has just entered the system. On the other hand, content-based algorithms need a very high amount of data to make consistent recommendations and may suffer from overspecialization, that is, most of the recommendations are too similar to the items that the user has already consumed. In order to alleviate some of these problems, hybrid Recommender Systems are often used. These techniques combine different strategies to obtain a better performance of the complete system (Burke, 2007). Nevertheless, among all of the mentioned techniques, the most well-known and analyzed recommenders are the content-based, collaborative filtering, and the hybrids Recommender Systems. Since we will focus on these three families of recommendation algorithms throughout the thesis, in the following subsection we will provide a more detailed explanation of these three families of recommendation algorithms.

## 2.2 Traditional classification of Recommender Systems

In this section, we review the most studied families of traditional Recommender Systems, describing their formulations and discussing their main advantages and disadvantages. Herein, we define as traditional Recommender System (or classical recommendation) those proposals that have been used in the area for several decades and that are still used today as basic baseline algorithms. We will make a special emphasis on the collaborative filtering approaches as they represent the most explored family in the area.

### 2.2.1 Content-based recommenders

Content-based (CB) algorithms recommend to the users items that those users liked in the past. Normally, these algorithms analyze the items and/or user features (content information) like genres in the case of movies, books, and music or categories (bar, restaurant, museum, etc.) in the case of POIs, and use them to create user and item profiles to recommend items to the target user that are similar to the ones she liked previously. Typically, three independent components are distinguished in a CB recommender (de Gemmis et al., 2015, Aggarwal, 2016):

- Content analyzer: in some situations, the available information of the items need to be pre-processed in a structured way from different sources (web pages, descriptions, documents, etc.) in order to extract keywords, concepts, or other information. Once such information has been processed, it can be used as input for the other components.
- Profile learner: using the content information of the items interacted previously by the user, this component builds a profile for every user in the system by applying Machine Learning techniques like regression or classification. The profile learner must have up-to-date information in order to adapt to users' changing tastes.
- Filtering component: once the user profile is learned, recommendations are performed in this step matching the user profile against the items in the system. The items are sorted according to their estimated relevance and recommended to the user.

Many content-based algorithms obtain the features of items from text using common techniques from the Information Retrieval area such as the Vector Space Model (VSM) (Baeza-Yates and Ribeiro-Neto, 2011), where an $n$-size vocabulary in the form of keywords or terms is obtained from documents, and then this vocabulary is used to represent each document $d_j$ in an $n$-dimensional space in which each coordinate represents the weight ($w_k$) for each term $t_k$. That is: $\vec{d_j} = \langle w_{1j}, w_{2j}, \cdots, w_{nj} \rangle$. To build these vectors, a common approach is using schemes based on Term Frequency (TF), Inverse Document Frequency (IDF), and combinations thereof (such as the well-known approaches of TF-IDF or BM25) (Cantador et al., 2010).

The TF-IDF is a well-known and simple mechanism for term-weighting which is based on two main assumptions. On the one hand, it assumes that terms that appear several times in a document are more relevant than those that appear few times and on the other hand, it considers that unusual terms are more important than usual ones. Thus, the TF-IDF scheme can be defined as:

$$TF-IDF(t_k, d_j) = TF(t_k, d_j) \cdot IDF(t_k) \tag{2.2}$$

where

$$TF(t_k, d_j) = \frac{f_{k,j}}{\max_z f_{z,j}} \tag{2.3}$$

$$IDF(t_k) = \log \frac{|D|}{n_k} \tag{2.4}$$

In these equations, $TF(t_k, d_j)$ takes into account the frequency of term $k$ by dividing such frequency by the maximum of the frequencies $f_{z,j}$ of all keywords $k_z$ that appear in document $d_j$. The second part of Equation 2.2, $IDF(t_k)$, will penalize keywords that appear in many documents as they do not help in distinguishing between useful and non-useful items. In this case, $|D|$ denotes the total number of documents of the system and $n_k$ is the number of documents where the term $k$ occurs at least once.

In order to bound the weights between [0,1], the TF-IDF function is usually normalized as follows:

$$w_{kj} = \frac{TF-IDF(t_k, d_j)}{\sqrt{\sum_{s=1}^{|T|} TF-IDF(t_s, d_j)^2}} \tag{2.5}$$

Once we have transformed all the items into vectors, a similarity metric (such as cosine similarity) can be applied to obtain a ranking of similar items with respect to others the user has previously consumed:

$$cos(\overrightarrow{w_u}, \overrightarrow{w_i}) = \frac{\sum_{j=1}^{k} w_{uj} w_{ij}}{\sqrt{\sum_{j=1}^{k} w_{uj}^2} \cdot \sqrt{\sum_{j=1}^{k} w_{ij}^2}} \tag{2.6}$$

In this type of representation, Equation 2.6 can be interpreted as the generic function $g$ defined in Equation 2.1.

Even though modeling this problem with a VSM is still popular nowadays, the use of embeddings has increased lately to exploit possible latent relationships between documents and associated terms (Lops et al., 2019). The process when working with embeddings would be the same as the one mentioned above with the VSM, where the space of the features is lower and more dense, so the sparsity is reduced, hence, allowing to address the curse of dimensionality (Amatriain and Pujol, 2015).

At the same time, probabilistic approaches like Bayesian Classifiers have also been applied in CB recommendation to classify documents usually in two main classes: whether the user considers the document as relevant or not. For this, a classifier is learned based on $P(c|d)$, that is, the probability to classify a document $d$ into class $c$ (e.g., the user likes it or dislikes it, or even one class for each possible rating value), which in turn is based on the probability of observing document $d$ given the class $c$, i.e., $P(d|c)$, and the prior probability of each class $P(c)$. Applying Bayes theorem:

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)} \tag{2.7}$$

The most difficult part is the estimation of $P(d|c)$, although it is common to use a Naïve Bayes classifier, in such a way that the document could be replaced by a vector of keywords following the same procedure as we explained before.

Apart from probabilistic models, relevance feedback and neighborhood-based algorithms are also used with content information. The former is a technique that refines the user profile by taking into account their opinion of the previous suggested items, with Rocchio's formula being one of the most well-known relevance feedback approaches used in Information Retrieval (Baeza-Yates and Ribeiro-Neto, 2011). For neighborhood-based algorithms (see Section 2.2.2) it is common to use a similarity function computed on the VSM representation of the items, and then select the class for the unclassified item taking into account the classes of the nearest neighbor items (de Gemmis et al., 2015).

As we can see from the aforementioned formulations, content-based recommendations will always look for items that resemble those the user has previously consumed, which means that they only rely on the interactions of the target user for building her profile (user-independence). Another important advantage is that items with very few interactions or even none can also be recommended. This is because, as long as we have the features associated to those items, we can build the VSMs and match them to the users profiles. However, it is important to mention that with this type of recommendation, it is difficult to surprise the user with new and diverse items, since the recommendation will always emphasize items with very similar features, as those are the ones enjoyed by the user in the past (de Gemmis et al., 2015, Adomavicius and Tuzhilin, 2005).

### 2.2.2 Collaborative filtering recommenders

Collaborative Filtering (CF) techniques analyze the interactions between users and items and establish patterns between them in order to make recommendations. These techniques are normally divided into two groups: memory-based (or nearest neighbors) that perform the recommendations using the interactions (usually represented as a user $\times$ item matrix as shown in Section 2.1) in a direct way by computing similarities between users and/or items (Ning et al., 2015), and model-based algorithms that build a predictive model by approximating the information stored in the preference or interaction matrix (Koren and Bell, 2015). We now explain some of the fundamental concepts related to these two families of CF algorithms.

#### Memory-based methods

Memory-based methods (also called nearest neighbors or $k$-NN) are one of the most well-known and implemented strategies in traditional recommendation due to its ease of programming and the great interpretability of the recommendations obtained (Ning et al., 2015).

The idea behind these algorithms is to recommend to the target user the most appropriate items by exploiting similarities between the rest of the users/items in the system. To do this, they build neighborhoods– by considering those users/items with highest similarities – and predict the score for new items based on those similarities and the scores provided by such neighbors (Ning et al., 2015). Although we can find both user (UB) or item-based (IB) $k$-NN algorithms, we will show the formulations for the UB approach (the IB formulations can be obtained in a complementary way). In the following equation, we denote how these methods estimate the utility function shown in Equation 2.1:

$$\hat{r}_{ui} = \sum_{v \in \mathcal{N}_i(u)} r_{vi} w_{uv} \tag{2.8}$$

where $\mathcal{N}_i(u)$ represent the top-$k$ neighbors with the highest similarity with the target user $u$, $w_{uv}$ denotes the similarity between users $u$ and $v$ and $r_{vi}$ represents the rating that the neighbor $v$ gave to item $i$.

Note that here we are representing the neighborhood-based formulation without normalizing the score, as in Aiolli (2013), although the original formulation normalizes the score with the sum of the similarities of all neighbors (dividing Equation 2.8 by $\sum_{v \in \mathcal{N}_i(u)} |w_{uv}|$). We decided to show the non-normalized version because this formulation has obtained better results in the task of top-n recommendation (Cremonesi et al., 2010). In this task, we are interested in obtaining a list of items ordered by score, to recover the top-n of the items to recommend a small subset, so that score does not need to be bounded by an interval.

Obviously, the similarity function is the most critical component in this type of algorithms, since it is used to select the neighbors and to weight each of them for the final score. Classical similarity metrics exploit trends in ratings such as Cosine similarity (Equation 2.9) or Pearson Correlation (Equation 2.10), although there are other approaches that do not use ratings, as they directly exploit how many items in common are recorded between user/item interactions, by means of variations of overlap measurements such as the Jaccard index (Equation 2.11):

$$\cos(u, v) = \frac{\sum_{i \in \mathcal{I}_{uv}} r_{ui} r_{vi}}{\sqrt{\sum_{i \in \mathcal{I}_u} r_{ui}^2 \sum_{j \in \mathcal{I}_v} r_{vj}^2}} \tag{2.9}$$

$$\mathrm{PC}(u, v) = \frac{\sum_{i \in \mathcal{I}_{uv}} (r_{ui} - \overline{r}_u)(r_{vi} - \overline{r}_v)}{\sqrt{\sum_{i \in \mathcal{I}_{uv}} (r_{ui} - \overline{r}_u)^2 \sum_{i \in \mathcal{I}_{uv}} (r_{vi} - \overline{r}_v)^2}} \tag{2.10}$$

$$\mathrm{Jaccard}(u, v) = \frac{|\mathcal{I}_u \cap \mathcal{I}_v|}{|\mathcal{I}_u \cup \mathcal{I}_v|} \tag{2.11}$$

where $\mathcal{I}_u$, $\mathcal{I}_v$ and $\mathcal{I}_{uv}$ denote the sets of items rated by users $u$, $v$, and $u$ and $v$ respectively.

### Model-based methods

Model-based algorithms represent the other major family of CF methods (Koren and Bell, 2015, Breese et al., 1998). This type of recommenders rely on machine learning techniques to learn a predictive model from the interactions of the users and items. Generally, these methods introduce a large number of parameters that are learned in a training phase minimizing a loss function. There are a large variety of model-based algorithms, including probabilistic models (Hernando et al., 2016, Li et al., 2007), neural networks (Hidasi and Karatzoglou, 2018, Tang and Wang, 2018), or matrix factorization models (Hu et al., 2008), which have enjoyed great popularity because of their importance on the Netflix Prize (Bell and Koren, 2007).

Matrix factorization approaches approximate the user $\times$ item matrix by transforming both users and items into a latent factor space of low dimensionality so that the user-item interactions can be explained (or recovered) by applying dot products in that space (Koren and Bell, 2015). Hence, these models associate each user $u \in \mathcal{U}$ with a vector $p_u \in \mathbb{R}^k$ and each item $i \in \mathcal{I}$ with a vector $q_i \in \mathbb{R}^k$ so that the inner product between $q_i$ and $p_u$ give us the rating of user $u$ to item $i$:

$$\hat{r}_{ui} = q_i^T \cdot p_u \tag{2.12}$$

The latent space is learned either by applying Stochastic Gradient Descent (SGD) or Alternating Least Squares (ALS) optimization techniques, depending on the domain characteristics and efficiency constraints (Koren and Bell, 2015). To do so, it is necessary to minimize the regularized squared error:

$$\min_{q*,p*} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2) \tag{2.13}$$

In this case, $\mathcal{K}$ is equivalent to the set of pairs $(u, i)$ whose rating is known (training set). The $\lambda$ variable is a constant whose purpose is to avoid overfitting (regularization term), as this is computed over the set of known ratings. For the Stochastic Gradient Descent approach, the system predicts $r_{ui}$ for each rating in the training set. The error is then computed as:

$$e_{ui} = r_{ui} - q_i^T \cdot p_u \tag{2.14}$$

Then the parameters are modified proportionally to $\gamma$ in the opposite direction of the gradient:

$$q_i \leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda q_i) \tag{2.15}$$

$$p_u \leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda q_u) \tag{2.16}$$

Although the optimization shown in Equation 2.13 is the core of many of these techniques, there are also proposals that incorporate additional contexts like temporal, sequential or geographical information if it is available in the domain that they are applied. In Section 2.3 we describe more in depth some of these approaches.

Of all the methods we can find for collaborative filtering, the probabilistic optimization model named Bayesian Personalized Ranking (BPR), proposed by Rendle et al. (2009), deserves special attention. The BPR is an optimization criterion specifically designed to obtain a ranking for the target user, unlike other aforementioned proposals that were optimized to predict if a user liked a specific item. Hence, the objective is to provide to every user $u$ a personalized ranking order $>_u$. To do so, they use item pairs for training so that if the user interacted with item $i_1$ but no with item $i_2$, we assume that the user preferred $i_1$ over $i_2$, that is: $i_1 >_u i_2$. Thus, the objective is to maximize the following posterior probability:

$$p(\Theta| >_u) \propto p(>_u |\Theta)p(\Theta) \tag{2.17}$$

where $\Theta$ represents the parameters of the model we are optimizing for (e.g., matrix factorization or a $k$-NN recommender) and $>_u$ represents the desired latent preference structure for user $u$. The authors assume all users are independent of each other and the ordering of each pair of items $(i, j)$ for each user is unique, hence it is personalized. Therefore, the likelihood function $p(>_u |\Theta)$ can be represented as:

$$\prod_{u \in \mathcal{U}} p(>_u |\Theta) = \prod_{(u,i,j) \in D_S} p(i >_u j|\Theta) \tag{2.18}$$

where $D_S$ is defined as: $D_S = \{(u, i, j)|i \in \mathcal{I}_u \land j \in \mathcal{I} \setminus \mathcal{I}_u\}$ and $p(>_u j|\Theta) = \sigma(\hat{x}_{uij}(\Theta))$, with $\sigma$ being the logistic sigmoid and $\hat{x}_{uij}$ being a function that models the relationship between user $u$ and the items $i$ and $j$. Finally, the prior density is approximated with a normal distribution, that is: $p(\Theta) \sim N(0, \Sigma_\Theta)$, setting $\Sigma_\Theta = \lambda_\Theta I$, with $\lambda_\Theta$ being the regularization parameters of the model, yielding to the following optimization criterion:

$$\begin{aligned}
BPR-OPT &= \ln p(\Theta| >_u) = \ln p(>_u |\Theta)p(\Theta) \\
&= \ln \prod_{(u,i,j)\in D_S} \sigma(\hat{x}_{uij})p(\Theta) = \sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{uij}) + \ln p(\Theta) \\
&= \sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta ||\Theta||^2
\end{aligned}$$

We want to emphasize that BPR is not an algorithm *per se*, but a loss function (optimization criterion). That is, the parameters to be optimized correspond to a recommendation model which can be, among others, a $k$-NN or a MF. In fact, this optimization method has been used in a large number of algorithms, including the next basket recommendation approach of Rendle et al. (2010) (further detailed in Section 2.3), the friend recommendation algorithm of Ding et al. (2017), or the Point-of-Interest recommendation models proposed in Li et al. (2015a) and Yuan et al. (2016). It is important to mention that the BPR is an example of a pair-wise learning to rank approach as it considers two items in the loss function (Karatzoglou et al., 2013).

As we have shown in this section, collaborative filtering techniques are very different from content-based ones as they do not exploit any additional information about the items or the users, just their interactions. Besides, it can be observed that in general the collaborative filtering recommenders do not have the same problem of specialization as the content-based ones, since the neighbors and/or latent factors can surprise the user with different recommendations (Adomavicius and Tuzhilin, 2005, Ning et al., 2015). However, the main drawback is that it is very difficult to recommend items to the users with very few interactions, as there will not be much neighbors who have scored it, nor will the factors associated with that item optimized.

### 2.2.3 Hybrid recommenders

All of the aforementioned types of recommenders may have some drawbacks under certain circumstances. As explained before, if a new item appears in a collaborative filtering recommender, it will not be recommended unless it is consumed by other users in the system. At the same time, most of the recommendations produced by a content-based recommender will be similar as it will only suggest items matching the item features of the previous items liked by the user. In order to alleviate these problems, different hybrid recommendation methods have been proposed, all of them oriented to combine several strategies in order to minimize the limitations of each independent algorithm (Ricci et al., 2015). According to Burke (2007), there are seven main hybridization techniques:

- Weighted: each recommender obtains a score for each candidate item and these scores are combined using a linear formula.
- Switching: it switches between recommenders depending on the situation. That is, it turns on one or another, so there is only one active recommender at each moment.
- Mixed: each recommender makes its own recommendations and the final output is a combination of them.
- Feature combination: features derived from various sources are combined and sent to the recommendation model.
- Feature augmentation: similar to feature combination but instead of deriving features of other recommender, the recommenders augment (generate) new features and send them to the final recommendation algorithm.
- Cascade: these hybrid techniques normally use a weak and a strong recommender. The recommendations are done using the main recommender and the weak model is only used when the strong recommender gives the same score to more than one item.
- Meta-level: a recommender produces a model, which is the input for the second recommender. Similar to feature augmentation but the second recommender does not work with raw data, only with the model provided by the first recommender.

Of all the possible combinations between families of recommender systems, the ones that stand out the most are the recommenders that use both collaborative and content information. In fact, one of the first recommendation models, proposed in Balabanovic and Shoham (1997), is a hybrid recommender system in which the users are provided with items scoring high against their profile and when they are also liked from users with a similar profile (content-based similarity). Other examples include the feature augmentation algorithm based on Bayesian Networks of de Campos et al. (2010) in which the balance between the collaborative and content-based parts is automatically selected and another feature augmentation algorithm proposed in Melville et al. (2002) that uses a content-based recommender to populate the rating matrix and applying then a user neighborhood model. Several examples can be found in recent research, where sometimes it is actually difficult to identify if an algorithm is a hybrid, since several components are typically considered, all interacting in a single model. We will show more examples about these and more complex hybrids in Chapter 3 (referring to POI, where we analyze many more types and combinations for hybrids).

## 2.3 Incorporating contextual information in Recommender Systems

The aforementioned Recommender Systems only made use of the information available from users' interactions with the items, except in the case of the content-based recommenders that also exploit the items features. However, it should be taken into account that users are influenced by other contexts when choosing the items they consume (Adomavicius and Tuzhilin, 2015). For example, the music a user consumes at work may not be the same as the music she consumes in his leisure time. When recommending touristic venues in a city, it may be interesting to take into account the weather and not recommend outdoor activities if it is raining. In fact, the weather is a context that is sometimes taken into account in the Point-of-Interest (POI) recommendation domain (Trattner et al., 2016, Villegas et al., 2018), as well as the geographical influence, as the users normally tend to visit nearby POIs (Bagci and Karagoz, 2015, Zhang and Chow, 2015a, Ren et al., 2017). However, beyond the POI recommendation problem, these contexts are not usually available. In contrast, two contexts are particularly relevant in traditional Recommender Systems and can be obtained more easily: the temporal and sequential information. Although they are sometimes considered as the same context, in this thesis we prefer to make a distinction between them. We argue that they should be treated independently, as not every sequence involves a temporal relationship and vice versa (e.g., a sequence of characters when searching for a word in a document does not contain any temporary information). Therefore, we consider that an algorithm exploits the temporal context as long as it uses any kind of timestamps in the model (like giving more weight to recent ratings, analyzing the different phases of the day, movie release dates, etc.). On the other hand, if the timestamps are only exploited to make sequences of interactions, we will state that it only uses sequential information.

Both temporal and sequential information have proven to be of vital importance to model and understand the evolution of the user behavior and have been applied in a large number of algorithms. One of the best known examples has been the matrix factorization algorithm (shown in Section 2.2.2), extended to incorporate the timestamps of interactions (Koren and Bell, 2015). In the $k$-NN recommenders systems, this temporal component has also been applied. For example, Ding and Li (2005) incorporated a time decay (TD) function in an item-based $k$-NN recommender:

$$\hat{r}_{ui} = \frac{\sum_{j \in N_i} r_{uj} \cdot w_{i,j} \cdot f_\lambda(t_{uj})}{\sum_{j \in N_i} w_{i,j} \cdot f_\lambda(t_{uj})} \tag{2.19}$$

where again $r_{ui}$ is the rating that user $u$ gave to item $i$ and $w(i, j)$ refers to the similarity between items $i$ and $j$ (any similarity function like Pearson Correlation, Cosine Similarity, etc.), and $t_{uj}$ is the moment when $u$ rated item $j$. The time function $f_\lambda(t)$ is defined as $f_\lambda(t) = e^{-\lambda t}$, with $\lambda = \frac{1}{T_0}$ (with $T_0$ representing days).

The authors introduce the exponential function where $\lambda$ is the decay rate or half-life parameter. It is defined so that old data obtains small weights, which means that larger values of $t$ correspond to older interaction times, where $t = 0$ is the present. It is not clear, however, how such values of $t$ are computed or normalized from the original timestamps, since no other information is provided; in particular, it is not obvious what $t = 0$ actually represents for the authors, is it the last interaction in training? in the whole dataset?

This approach based on a time decay function has also been adapted to user-based recommenders, as shown in Campos et al. (2014):

$$\hat{r}_{ui} = \overline{r}_u + \frac{\sum_{v \in N_u} (r_{vi} - \overline{r}_v) \cdot w_{u,v} \cdot e^{-\lambda \cdot (t - t_{vi})}}{\sum_{v \in N_u} w_{u,v}} \tag{2.20}$$

in which the authors used the mean-centering formulation besides incorporating the temporal influence in the user-based recommender. The mean-centering is a normalization scheme that allow us to detect if a the rating that a user $u$ gives to an item $i$ is positive of negative by comparing that rating to the user's mean.

Moreover, in this case the time decay function receives the difference of two timestamps (instead of only one as before): the neighbor's interaction time $t_{vi}$ and the prediction time $t$. Again, it is not obvious how such time $t$ is used in this formulation, although it is very likely that it would represent the timestamp that appears in the test set, which exposes an unrealistic situation since the recommender would be using a different $t$ for every item in the test set, exploiting information that should remain hidden to the algorithm in the prediction stage.

Leaving aside this time decay similarity function, time analysis of user's preferences has also favored the emergence of algorithms that exploit these user-item interactions as sequences in their models. In this type of sequential recommendation, researchers model each user in the system as a sequence of actions $S(u) = (S_1, S_2, S_3, \cdots, S_n)$ – following a similar notation as in He and McAuley (2016) – where each action corresponds to one item the user has consumed. Thus, the recommendation problem here is to predict the next item to be consumed by the user. While most traditional algorithms rely on the full history of the user, in sequence recommendation only the latest interactions of the user are normally considered (or, at least, the most recent ratings receive more attention than the others).

One classical approach to identify these sequential patterns is to use an L-Markov chain model, where L denotes the number of previous actions that are taken into account in order to make the recommendations. The L-order Markov chain for modeling user sequences can be represented as:

$$p(S_t \mid S_{t-1}, S_{t-2}, \cdots, S_{t-L}) \tag{2.21}$$

When considering a first order Markov chain ($L = 1$), the probability of choosing item $j$ given the actual item $i$ at the next step, $p(j \mid i)$, is obtained by using maximum likelihood estimation on the item-to-item transition matrix. This model, although simple, is at the core of many approximations. For example, Rendle et al. (2010) proposed a more complex approach for basket recommendation where each user has its own transition matrix, leading to a global representation of a transition tensor: $\mathcal{A} \in [0,1]^{|\mathcal{U}|\times|\mathcal{I}|\times|\mathcal{I}|}$. To deal with the high sparsity of tensor $\mathcal{A}$, they approximate it using the Canonical Decomposition $\mathcal{A} := \mathcal{C} \times V^{\mathcal{U}} \times V^{\mathcal{L}} \times V^{\mathcal{I}}$, where $V^{\mathcal{U}}$, $V^{\mathcal{L}}$ and $V^{\mathcal{I}}$ represent the feature matrix of the users, the items of the last transition and the items to predict, respectively, all with the same dimension $k$. For modeling the pairwise interactions between the three main components of the model (users, and both types of items), they use two factorization matrices for each of them: hence, for modeling the relationship between $\mathcal{U}$ and $\mathcal{I}$ they have matrices $V^{\mathcal{U},\mathcal{I}}$ and $V^{\mathcal{I},\mathcal{U}}$, for $\mathcal{I}$ and $\mathcal{L}$ they use matrices $V^{\mathcal{I},\mathcal{L}}$ and $V^{\mathcal{L},\mathcal{I}}$ and for the interaction between $\mathcal{U}$ and $\mathcal{L}$ they use matrices $V^{\mathcal{U},\mathcal{L}}$ and $V^{\mathcal{L},\mathcal{U}}$. Apart of exploiting the transitions between the items, they also incorporated matrix factorization techniques in their probabilistic framework, combining both approaches (MF and Markov Chains) into a single model: Factorizing Personalized Markov Chain (FPMC). In that model they estimate the probability of item $i$ belonging to the actual basket $B_t$ as follows:

$$\hat{x}_{u,t,i} = v_u^{\mathcal{U},\mathcal{L}} \cdot v_i^{\mathcal{I},\mathcal{U}} + \frac{1}{|B_{t-1}^u|} \sum_{l \in B_{t-1}^u} v_i^{\mathcal{I},\mathcal{L}} \cdot v_l^{\mathcal{L},\mathcal{I}} \tag{2.22}$$

Another interesting approach that model sequences of actions is defined in He and McAuley (2016). In that article, the authors use an $L$-order Markov Chain making a weighted sum for the short and long term dynamics of the user preferences. The proposed model (named Fossil, from Factorized Sequential Prediction with Item Similarity Models) is a combination of both Factored Item Similarity Models (FISM) and Markov Chains with personalization. In this context, FISM is a method to factorize an item-to-item similarity matrix $W$ (of dimension $|\mathcal{I}| \times |\mathcal{I}|$) into two low-rank matrices in order to reduce the number of parameters and reduce sparsity of matrix $W$ as follows:

$$W = P \cdot Q^T \tag{2.23}$$

where $P$ and $Q$ are $|\mathcal{I}| \times K$ where $K << |\mathcal{I}|$. For exploiting the sequential information, in Fossil the authors model short-term dynamics using High-order Markov Chains. Instead of using a first-order Markov Chain, they allow to exploit the most $L$ recent items that the user has consumed. Hence, the final model combines both FISM and $L$-order Markov Chains:

$$p_u(j \mid S^u_{t-1}, S^u_{t-2}, \cdots, S^u_{t-L}) \propto \beta_j + \left\langle \frac{1}{|\mathfrak{I}_u \setminus \{j\}|^\alpha} \sum_{j' \in \mathfrak{I}_u \setminus j} P'_j + \sum_{k=1}^{L} (\eta_k + \eta^u_k) \cdot P_{S^u_{t-k}}, Q_j \right\rangle \tag{2.24}$$

According to the reported results in their paper, the Fossil approach outperforms many sequential state-of-the-art algorithms, including the FISM techniques, FPMC, and matrix-factorization models optimized with BPR, although the authors did not analyze the performance of other classic recommenders such as standard MF approaches or neighborhood-based algorithms.

In addition to the aforementioned algorithms, neural networks techniques have also acquired special relevance recently in recommendation because of their success in other related areas such as Machine Learning (ML) and in the industry with companies like Google or Netflix (Johnson et al., 2017, Abadi et al., 2015, Liberty et al., 2020, Zhang et al., 2019a). Besides, there are many deep learning techniques that have been applied in recommendation, such as Multi-Layer Perceptrons, Autoencoders, restricted Boltzmann machines, etc. (Zhang et al., 2019a). Many of these techniques are flexible enough to incorporate some of the contextual information mentioned above. For example, Hidasi et al. (2016) proposed GRU4Rec in 2015 (and improved in 2016) a Recurrent Neural Network session-based algorithm that captures the sequential dependencies between the user preferences in order to make recommendations. For next-item recommendation we can find the new type of GRU presented in Donkers et al. (2017) and the behavior-intensive neural network from Li et al. (2018a), consisting on two main components, the neural item embedding (for obtaining a unified representation of the items) and the discriminative behavior learning that modelizes the interests of the user over time and the actual consumption motivation. Finally, Tang and Wang (2018) proposed Caser, a Convolutional Neural Network model designed to learn the sequential patterns by applying convolutional filters to the item embeddings in order to make predictions. The architecture of the model is shown in Figure 2.1.

Despite the large number of proposals on neural networks, a debate has recently opened in the Recommender Systems community to clarify whether they are so useful (or whether they are being implemented correctly). Regarding this issue, Dacrema et al. (2019) analyzed a total of 18 deep learning models presented in the most important conferences in the area, reaching the conclusion that only seven of them could be replicated and that six that seven approaches achieved a worse performance than other simpler models such as neighbor-based algorithms. The authors attributed these differences in results to using not appropriate evaluation methodologies, not tuning properly the baselines, not using competitive baselines, etc. For these reasons, we believe that it is always critical to be concise in the evaluation methodology followed in the experiments, and we advocate for the importance of providing the source code of the algorithms if possible.

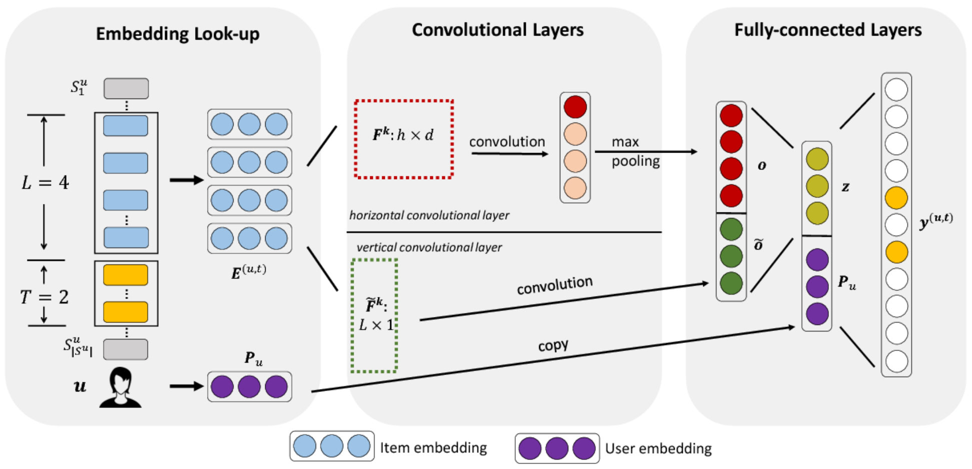


**Figure 2.1:** Network architecture of the Caser model as shown in Tang and Wang (2018).

## 2.4 Recommender Systems evaluation

As well as there is an extensive and an ongoing research in the recommendation models, the evaluation of Recommender Systems remains a very active topic in the area. If the quality of the recommendations is good enough, it will attract more and more users to our system. Normally, when we want to incorporate a recommendation algorithm to an application, we have several possible candidates. For the selection of the final recommender we need to design different types of experiments, considering diverse factors (such as performance in terms of results obtained, memory used, reliability and temporal and spatial scalability (Gunawardana and Shani, 2015)).

Due to the large number of issues that need to be taken into account to evaluate the algorithms, in this section we review some of the most important ones when evaluating Recommender Systems.

### 2.4.1 Evaluation settings

There are principally three major types of evaluation methodologies (Gunawardana and Shani, 2015):

- Offline experiments: in these experiments, we exploit the preferences of the users from a dataset normally obtained from an already deployed system. These type of experiments are widely used as they do not need to interact with real users so they allow a faster comparison within the algorithms. As a counterpart, they usually work with very little information, since we do not have additional data from the users, such as the opinion of the users regarding the recommendations (e.g., if they found them interesting or surprising) or the actual user satisfaction. Some institutions and researchers have made public datasets for offline experiments, such as the University of Minnesota with the GroupLens[1] datasets (including

[1]GroupLens datasets, `https://grouplens.org/datasets/`

MovieLens for movies or Book Crossing for books), the Stanford Network Analysis Project (SNAP)[2] with datasets coming from social networks including Gowalla, Brighkite, and Facebook, or the Amazon product datasets available in Julian McAuley webpage[3].

- User Studies: in user studies, a group of people is selected to perform certain actions using the recommendation system by monitoring their behavior. Sometimes, these user studies are also accompanied by questions to analyze user satisfaction (Hu and Pu, 2009). Although the information that can be obtained from user studies is valuable, it must be taken into account that running this type of study is very expensive, in addition to the fact that there may be users who are not completely sincere as they may be interested only in being paid for participating in the study (Russell et al., 2000).
- Online experiments: these type of experiments are normally performed in systems that are already working where there is a large number of users, studying their behavior using protocols such as A/B testing. In general, under this configuration, the system randomly redirects the users to two different engines or small variations of a single one, the one in production and the alternative one, so that the interactions of the users in the recommendation engines are recorded and analyzed (Kohavi et al., 2009). Normally these types of experiments are done in industry, but they are not so common in academia.

In this thesis we focus on offline evaluation, hence, for a review of other evaluation techniques, we refer the reader to the following works (Knijnenburg and Willemsen, 2015, Kohavi et al., 2009).

### 2.4.2 Evaluation methodologies

In offline evaluation, the performance of recommenders is usually measured by comparing the recommendations produced with the ground truth of the items we know are relevant to the user. Therefore, starting from the complete dataset $\mathcal{R}$, at least one partition is usually made by dividing the dataset into two subsets, the training set $\mathcal{R}_{\text{training}}$ and the test set $\mathcal{R}_{\text{test}}$ so that $\mathcal{R} = \mathcal{R}_{\text{training}} \cup \mathcal{R}_{\text{test}}$. In some research areas such as Machine Learning (ML), it is common to add a third subset to optimize the parameters of the algorithms, called the validation set $\mathcal{R}_{\text{val}}$, so that $\mathcal{R} = \mathcal{R}_{\text{training}} \cup \mathcal{R}_{\text{val}} \cup \mathcal{R}_{\text{test}}$. The percentage of information contained in these subsets usually varies depending on the context but generally the training set always contains more data than both the validation and test set. Thus, the standard procedure is to train the recommendation algorithm with the training set, select the best parameters with the validation set (if available) and evaluate the recommenders using the test set. To create these subsets there are several strategies. A common evaluation procedure is to perform random partitions where each iteration of the complete dataset had a probability $p_{\text{training}}$ to go

[2]SNAP datasets, http://snap.stanford.edu/data/index.html

[3]Amazon products datasets, https://jmcauley.ucsd.edu/data/amazon/

to training, a $p_{\text{val}}$ probability for validation, and a $p_{\text{test}}$ probability to go to test (with $p_{\text{training}} + p_{\text{val}} + p_{\text{test}} = 1$). Other common evaluation methodology is the so-called $k$-fold cross-validation in which the full dataset is splitted in $k$ roughly equal and non overlapping subsets in which one fold is left as the test set and the rest of the folds are used to train the recommenders. Finally, this process is repeated $k$ times.

However, this type of evaluation is unfair in contexts where temporal information is available because there may be interactions in the training set that occurred in a posterior moment of time than other interactions in the test set. This affects the recommenders who use temporal information within their models, but also it is an unrealistic methodology as when using a real environment we cannot access to information that occurs in the future. For that reason, it is becoming more frequent to see temporal splits where the user interactions in the test are always posterior with respect to the ones contained in the training set. Nonetheless, this temporal split can be done according to different hypotheses or scenarios in mind; this is usually translated into splitting the data either at the user level or at the system level, according to certain restrictions. For more information on this type of splits, Campos et al. (2014) present a taxonomy of recommendation evaluation protocols or methodologies, where they are classified according to the following conditions: base set (it specifies if the splitting is applied to the whole dataset or to each subset independently, such as one dataset for each user), ordering (it establishes an ordered sequence for the ratings), and size (criterion to compute the number of ratings to be assigned to each training and test split). As expected, the ordering condition is the one that controls whether an evaluation methodology is sensitive to the temporal dimension, the other two conditions entail different constraints that may produce more or less realistic methodologies, such as fixed vs proportion size (where all the users in the test set contain exactly the same number of interactions or a ratio of their whole user profile) or community-centered vs user-centered base set (where all the ratings are used as a whole or each user is considered as a separate dataset). Nevertheless, we will focus on two common procedures in time-aware recommendation, one with a user-centered base set with fixed size condition (that we will name *per user*) and the community-centered base set with a proportion size condition (that we will name *system*). With a per user methodology for each user in the dataset, their interactions are sorted in ascending order selecting some of the first interactions to the training set, leaving the rest (the more modern ones) for the test set. On the other hand, in the system methodology a timestamp is selected so that all interactions that occurred before that specific moment in time will form the training set and the rest of the interactions will form the test set. It is noteworthy to mention that these same system or per user protocols can be applied in a random partitioning (i.e. select a percentage of the interactions of the whole dataset to test or select a number of interactions of each user randomly to test).

Finally, it is important to note that although in areas such as Machine Learning it is very common to perform a validation step to optimize the recommenders, in the recommendation domain the best parameters are usually reported directly in the test set, as can be observed in some popular works (Hu et al., 2008, Rendle et al., 2009, 2010, Ding and Li, 2005, Qu et al., 2016, Li et al., 2016a, Yang et al., 2017a). Nevertheless, we can also find parameter optimization through different kind of validation subsets in other relevant works (Tang and Wang, 2018, Hidasi et al., 2016, Yuan et al., 2019). On the one hand, if no validation step is performed in all recommenders it might be argued that the overall results should remain stable as the same evaluation methodology is performed in all recommenders. On the other hand, the results will probably be too optimistic and this procedure may favor those models with more parameters as they have a larger space (or more degrees of freedom) to find an optimal solution. Nevertheless, in our opinion, this is an aspect that is overlooked in the community and more attention should be paid in the future to ensure transparent, reproducible, and comparable research.

### 2.4.3 Evaluation metrics

In the emergence of Recommender Systems it was assumed that the models were better if they were able to solve the main task considered then: rating prediction. That is, the system is good if the difference between the predicted rating and the real rating is low. To this end, two main metrics were used, Mean Absolute Error (MAE), and the Root Mean Squared Error (RMSE):

$$\text{MAE} = \frac{1}{|\mathcal{R}_{test}|} \sum_{r_{ui} \in \mathcal{R}_{test}} |\hat{r}(u,i) - r_{ui}| \tag{2.25}$$

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{R}_{test}|} \sum_{r_{ui} \in \mathcal{R}_{test}} (\hat{r}(u,i) - r_{ui})^2} \tag{2.26}$$

where $\hat{r}$ is the predicted rating (provided by the recommender) and $r$ is the real rating that the user $u$ gave to item $i$. $\mathcal{R}_{test}$ refers to the ratings in the test set, which forms the ground truth. Although both metrics are used to measure the error, normally RMSE is preferred as it penalizes more larger errors.

However, in recent years, the evaluation metrics used in Recommender Systems have changed. One important reason is that sometimes the real ratings are not available in the systems, only a list of the items liked by the user. A second, probably more decisive, reason for this shift in evaluation is that recommenders, do not try to predict a rating anymore, but provide users with a list of recommendations normally of not consumed items (top-n recommendation). For this reason, since these metrics are not useful in this new scenario, other evaluation metrics emerged, oriented at comparing the suggested recommendation list to the users' and the list of items consumed in the test set by them. We now show the formulations of these metrics computed for a given user $u$. To obtain the value of the complete metric for a full set of recommendations, we compute the value of the metric for each user individually, and then we divide it by the total number of recommended users. Two of the most famous metrics are Precision@k (P@k) and Recall@k (R@k), shown in Equations 2.27 and 2.28 respectively. The former measures the fraction of recommended items in the top k positions of the ranking that are relevant while Recall indicates the fraction of relevant items that were recommended in the top k positions:

$$\mathrm{P@k(u)} = \frac{\mathrm{Rel_u@k}}{\mathrm{k}} \tag{2.27}$$

$$\mathrm{R@k(u)} = \frac{\mathrm{Rel_u@k}}{\mathrm{Rel_u}} \tag{2.28}$$

where $\mathrm{Rel_u@k}$ denotes the set of relevant items recommended at top k and $\mathrm{Rel_u}$ represents the set of all relevant items of the user.

Mean Average Precision (MAP) (Equation 2.29) is another popular ranking metric in the area. It calculates Precision@k at each k position where a relevant item is found and then computes the average.

$$\mathrm{MAP(u)} = \frac{1}{|\mathrm{Rel_u}|} \sum_{\{k:i_k \in \mathrm{Rel_u}\}} P@k \tag{2.29}$$

where $i_k$ denotes the item at position $k$ in the recommended list.

Finally, Järvelin and Kekäläinen (2002) proposed the normalized Discounted Cumulative Gain (nDCG), shown in Equations 2.30 and 2.31 which, in addition to allowing several levels of relevance, it takes into account the position of relevant items in the ranking by applying a discount model. The discount component models the fact that items located in the lowest position in the ranked list are less likely to be seen by the user. In this sense, those recommendation lists that include non-relevant items in the first positions of the rankings will be further penalized. Because of this, this metric is usually favored with respect to the previous ones, as it encapsulates different levels of relevance and penalization by ranking position.

$$\mathrm{nDCG@k(u)} = \frac{\mathrm{DCG}(u)@k}{\mathrm{IDCG}(u)@k} \tag{2.30}$$

$$\mathrm{DCG}(u)@k = \sum_{n=1}^{k} \frac{2^{rel_n} - 1}{\log_2(n+1)} \tag{2.31}$$

where $rel_n$ denotes the real relevance of item $n$ (normally a value between 0 and 5, with 0 representing a non-relevant value). Finally, IDCG is computed in the same way as DCG but using the list of relevant items (ordered by descending real relevance).

All these metrics are bounded in the $[0, 1]$ interval, with 1 denoting the perfect score and 0 the lowest value.

### Novelty and Diversity

Accuracy evaluation has been the most extended way of evaluating the performance of Recommender Systems for many years, either measured in terms of decreasing the error or as ranking prediction. However, some researchers have alerted for a long time that this type of evaluation is “incomplete” because it does not take into account other possible contexts as not always a good performance in accuracy translates in a better user experience (McNee et al., 2006, Ge et al., 2010, Campos et al., 2014).

Two of the most important perspectives in this regard are novelty and diversity. Although they are related concepts, they are not synonyms. Diversity normally only refers about how different the recommended items are to each other, whereas novelty measures how "unknown" the recommended items are for a community or the target user (Castells et al., 2015). These dimensions, although they do not measure the accuracy of recommendations, should be as fundamental as the evaluation presented previously (Castells et al., 2015, Vargas and Castells, 2011) and, hence, the results that we will show throughout this thesis will include the performance of the recommenders in terms of accuracy as well as in these complementary dimensions.

Some metrics that are used to measure diversity are (Castells et al., 2015, Vargas and Castells, 2014):

**Aggregate diversity**: A diversity metric which counts the total number of items that the system recommends taking into account all recommendation lists $R_u$ received by every user. Sometimes is also referred as "Item Coverage" (IC).

$$\text{AggDiv} = \left| \bigcup_{u \in \mathcal{U}} R_u \right| \tag{2.32}$$

this metric is often reported normalized by the total number of items.

**Intra-List Distance**: One of the first diversity metrics proposed (Smyth and McClave, 2001). It is computed as the average pairwise distance of the items in the set of recommended items in a recommendation list $R_u$:

$$\text{ILD(u)} = \frac{1}{|R_u|(|R_u| - 1)} \sum_{i \in R_u} \sum_{j \in R_u} d(i, j) \tag{2.33}$$

where $d(i, j)$ is a distance metric that measures the difference between items $i$ and $j$.

**Gini Index**: A diversity metric that shows how unequally different the items chosen by a particular system $s$ are:

$$\text{Gini} = 1 - \frac{1}{|\mathcal{I}| - 1} \sum_{k=1}^{|\mathcal{I}|} (2k - |\mathcal{I}| - 1) p(i_k \mid s) \tag{2.34}$$

$$p(i \mid s) = \frac{|\{u \in \mathcal{U} | i \in R_u^s\}|}{\sum_{j \in \mathcal{I}} |\{u \in \mathcal{U} | j \in R_u^s\}|} \tag{2.35}$$

where $p(i_k \mid s)$ is the probability of the $k$-th least recommended item being drawn from the recommendation list generated by $s$, that is, when considering all rankings $R_u^s$ for every user. In this case, we will use the complementary of the Gini Index proposed in Castells et al. (2015), as defined in Vargas and Castells (2014).

With respect to novelty, this dimension has generally been modeled as the opposite of popularity. Two examples of this would be the Inverse User Frequency (see Equation 2.36) and the Expected Popularity Complement (or EPC) shown in Equation 2.37.

**Inverse User Frequency:** A novelty metric based on the IDF model described in Section 2.2.1, in which the novelty of a recommendation list for a user $u$ is defined as:

$$MIUF(u) = -\frac{1}{|R_u|} \sum_{i \in R_u} \log_2 \frac{|\mathfrak{U}_i|}{|\mathfrak{U}|} \tag{2.36}$$

**Expected Popularity Complement:** This metric, derived from the novelty and diversity framework from Vargas and Castells (2011), considers that the novelty of the items is complementary to the probability of being consumed by the rest of the users in the system. Besides, this metric also incorporates a discount and a relevance model, to allow us to measure accuracy and novelty in the same metric.

$$EPC(u) = C \sum_{i_k \in R_u} \text{disc}(k) p(rel \mid i_k, u)(1 - p(\text{seen} \mid i_k)) \tag{2.37}$$

where $C$ is a normalizing constant (generally $C = 1/\sum_{i \in R_u}$), $disc(n)$ denotes a discount function, as in nDCG (a common discount function is $disc(n) = 1/log_2(n)$) and the term $p(rel \mid i_n, u)$ represent the probability of being relevant given the item $i$ and user $u$.

### Technical considerations when computing the metrics

In the evaluation step, there are several factors that may influence the performance of the recommenders, from the quality of the data to the types of processing done to the original inputs. However, there are several, more subtle considerations when running the evaluation that may have profound implications. Even using the same datasets, splits, and metrics reported in a specific paper, different results might be obtained depending on the number of users to whom we are making recommendations and the set of items to be considered as candidates for each user. Regarding the set of items to be recommended, Said and Bellogín (2014) describe two of the most well-known approaches for selecting candidates items to be recommended and show that they have a strong impact on the results of the recommenders: TrainItems and RelPlusN. In TrainItems, every item in the training set are considered as candidate except the ones already rated by the target user. In RelPlusN, one high relevant item is selected from the test set and then a set of non-relevant items is created by selecting $N$ additional items. This latter strategy is often used in neural networks approaches because it is cheaper mechanism than generating a score for every possible candidate item. For both strategies, the metrics are then computed as shown before. However, a recent study conducted by Krichene and Rendle (2020) concluded that the performance of the metrics were highly dependent on the type of mechanism used to select the non-relevant items and hence strategies like RelPlusN should be avoided if possible. Because of that, unless stated otherwise, in this thesis we will make recommendations following the TrainItems methodology.

On the other hand, depending on the type of split, there might be users that appear in the training set but not in the test set and vice versa, so it is generally assumed that the recommendations should be made to the users that appear in the test set to avoid generating useless lists of recommendations. While this seems obvious as it does not affect the performance on accuracy metrics (since we can simply ignore users who do not appear in the test set), for non-accuracy metrics it has a substantial effect, since they do not focus on analyzing the relevance of the recommendations. Hence, the performance of the non-accuracy metrics may vary if we consider making recommendations for all users in the training set, instead of those users who only appear in the test set. Independently of that, other questions still remain. What happens if there are users in the test set who do not appear in the training set? This, in fact, is very common in some types of temporal splits. On the one hand, non-personalized algorithms such as a Random or a Popularity recommender can make recommendations to new users because they do not learn any tastes of the users. However, other recommendation algorithms like those presented previously in Sections 2.2 and 2.3 would be incapable of providing recommendations to those users.

For these reasons, in this thesis, unless stated otherwise, recommendations will be made for users who appear in test to whom the recommenders are capable of providing a recommendation. This means that there may be recommenders who have a lower or higher user coverage than others. At the same time, when computing the average of the metrics, we will only consider the users in the test that have at least one relevant item. To see more in detail the implications of this way of computing the average of the metrics, let us consider the example shown in Figure 2.2. In this Figure we observe three recommendation lists for three different users ($R_{u_1}$, $R_{u_2}$ and $R_{u_3}$). Below the recommended lists, the items in the test for each user are also shown ($T_{u_1}$, $T_{u_2}$ and $T_{u_3}$ with their corresponding ratings). If no relevance threshold is used (i.e., we consider all items in the test set as relevant) and the metric P@5, the final result will be:

$$P@5 = \frac{P@5(u_1) + P@5(u_2) + P@5(u_3)}{|\mathcal{U}_{test}|} = \frac{\frac{2}{5} + \frac{2}{5} + \frac{3}{5}}{3} = \frac{\frac{7}{5}}{3} = 0.4\widehat{6}$$

However, if we consider an evaluation threshold of 4 (i.e., we consider as relevant the items in the test set that have a rating $\geq 4$). In this case, the result would be:

$$P@5 = \frac{P@5(u_1) + P@5(u_2) + P@5(u_3)}{|\mathcal{U}_{test}|} = \frac{\frac{1}{5} + \frac{0}{5} + \frac{3}{5}}{3} = \frac{\frac{4}{5}}{3} = 0.2\widehat{6}$$

But, with the relevance threshold of 4, in this example, for the $u_2$ we can never provide a useful recommendation as that user does not have any relevant item in the test set, thus, we argue that we should ignore this user and use only the users in the test set with at least one relevant item using a relevance threshold of 4 (i.e., $|\mathcal{U}^4_{test}|$):

$$P@5 = \frac{P@5(u_1) + P@5(u_2) + P@5(u_3)}{|\mathcal{U}^4_{test}|} = \frac{\frac{1}{5} + \frac{3}{5}}{2} = \frac{\frac{4}{5}}{2} = 0.4$$

Based on this discussion, in this thesis, we will compute the average of the metrics by all users to whom we have provided recommendations ignoring those who do not have any relevant items in the test set, that is, as in the last formulation. To complement

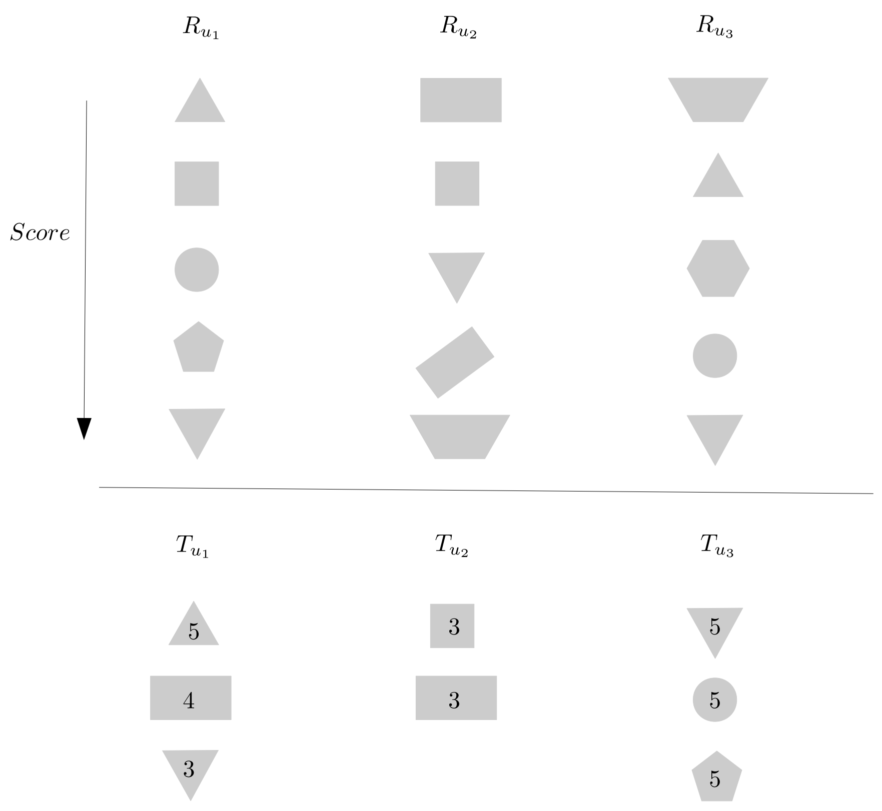


**Figure 2.2:** Toy example for analyzing the difference in results obtained by three users depending on the type of evaluation used. The geometrical figures represent each item and the number denotes the rating the user gave to those items in the test set.

the metrics and provide a full picture of the performance of the algorithms, we will also report the number of users each recommender is able to provide recommendations to (also known as user coverage).

# 3

# Point-of-Interest Recommendation

As aforementioned in the thesis, Recommender Systems can be applied in a large number of domains. In this chapter we will study in more detail the Point-of-Interest (POI) recommendation problem analyzing the similarities and differences with respect to traditional recommendation. Firstly, we define the POI recommendation problem by exploring the specific characteristics of this domain, including the main differences with respect to the original recommendation problem and the alternative sources of information that are sometimes used in the area. Then, we show a review of the works on POI recommendation between 2011 and 2019 (retrieved by three different databases), characterizing the types of algorithms, information, and evaluation mechanisms currently used in the domain. Finally, we focus on those works with more cites and analyze the datasets used to show how comparable the algorithms are with each other.

The work presented in this chapter has been sent for publication to the ACM Computing Surveys journal:

- **Pablo Sánchez** and Alejandro Bellogín. (2020). Point-of-Interest Recommender Systems: A Survey from an Experimental Perspective. Submitted to *ACM Computing Surveys*. Under Review.

## 3.1 Problem definition

The POI recommendation problem is *formally* the same as the traditional recommendation problem defined in Section 2 except that the items in this case are physical locations or venues (museums, hotels, restaurants, parks, etc.). Thus, the main objective is to recommend new places to the users when they visit a specific region (usually a city). In this domain, the Location-Based Social Networks (LBSNs) are specially relevant as the users can establish social links with other users in those systems, share information, and record check-ins to the specific venues they visit when located in a city (Zhang and Chow, 2013, Yuan et al., 2016, Wang et al., 2018a). Foursquare, Gowalla, or Yelp are examples of this kind of social networks that are currently explored by the research community. However, even if POI recommendation is similar to the traditional recommendation problem, there are specific details that differ from classical recommendation that need to be further analyzed (Li et al., 2015a, Wang et al., 2013, Liu et al., 2017). These include, but are not limited, to:

- **Sparsity:** in the RS domain, the user $\times$ item matrix is normally very sparse. However, in venue recommendation this effect is even more severe because there are in general fewer known values of the user $\times$ item matrix. For example, the densities of the Movielens20M and Netflix datasets (classic movie datasets used in standard recommendation) are 0.539% and 1.177% respectively, while the sparsity of datasets from Foursquare (Yang et al., 2016) and Gowalla (Cho et al., 2011) are 0.0034% and 0.0047% respectively.

- **Implicit and repeated interactions:** in classic recommendation, the information encoded in the user $\times$ item matrix has been traditionally modeled using ratings. However, in most POI recommendation datasets (e.g., Brightkite, Gowalla, or Foursquare), we only have the specific moment of time in which a user visited a POI. Moreover, since users may check-in at the same place several times, researchers often build frequency matrices to model these repetitions. In fact, the presence of repeated interactions has a strong effect on performance when exploited properly (Sánchez and Bellogín, 2018a). This differs from traditional recommendation, where it is normally assumed that users rate each item once (Ning et al., 2015).

- **External influences:** while most classic recommendation approaches only exploit the information available in the user $\times$ item matrix (user, item, score, and sometimes the timestamp associated), venue recommendation is highly affected by temporal, social (user friends), and geographical influences. The latter is possibly the most critical effect to consider in POI recommendation to improve the recommendation performance, as it is usually assumed that users prefer to visit venues that are close to each other. As the first law of geography states "*Everything is related to everything else, but near things are more related than distant things*" (Miller, 2004). That is the reason why researchers have proposed algorithms including, as an explicit component, the modeling of the venue location (Liu et al., 2014, Ye et al., 2011, Lian et al., 2014). Nevertheless, these influences are not only important to improve the performance of the algorithms, sometimes it is mandatory to take them into account because they impose certain restrictions on the recommendations. For example, some POIs such as shops, restaurants, museums, etc. are only open for a certain period and users cannot make visits to POIs that are too distant from each other.

## 3.2 Alternative information sources

As presented before, the density of most POI recommendation datasets is very low. For this reason, most POI recommendation approaches use some kind of auxiliary information; in particular, the vast majority of them use more than one source of information. As we shall discuss later, most of the processed articles that we will show in this chapter, work with one or more of the following sources of information.

### 3.2.1 Interaction types

Although we have equated check-ins in LBSNs with the main interaction between users and items (as ratings in classical recommender systems), this is not the only type of interaction recorded in this type of systems. Other LBSNs – such as Yelp – allow users to perform reviews of the POIs they visit and, in some cases, rate the item; through these reviews we can determine whether the user liked the POI or not, either by directly considering the rating or by analyzing the sentiment of the text in the review (Hu and Ester, 2014, Manotumruksa et al., 2016). Other works obtain the items to be processed from the photos that users take and upload to other applications such as Flickr or Instagram (Nie et al., 2016, Wang et al., 2017a), which may include GPS coordinates as their metadata along with visual information, so that the path followed by the users could be recovered. Similarly, user generated content tagged with GPS coordinates – such as tweets from Twitter or the traces left by mobile apps – can potentially be used in POI recommendation applications (Zheng et al., 2018, Zhong and Ma, 2018).

### 3.2.2 Rich side information of items

The items in this type of systems, POIs, can be associated with a richer kind of information than in other domains. First, each POI has a geographic location associated, although this information is not always available in the datasets. This source of knowledge is especially relevant because people tend to go to places that are close to each other. Sometimes this information is exploited to calculate centroids or clusters of activity for either users and items in order to make recommendations (Si et al., 2019, Liu et al., 2014). In relation to the algorithms that we are going to analyze for this chapter, we consider that a model uses this kind of information if it uses the user/POIs coordinates somewhere in the proposed model (e.g., when computing distances, creating clusters, building distributions based on proximity, and so on).

Second, and more similar to the traditional recommendation situation where items usually have associated characteristics – such as genres in the movie, book, or music domains –, in POI recommendation the venues are frequently linked to a specific POI category (e.g. restaurants, hotels, parks, museums, etc.), which may have different levels (thus, building a category hierarchy) depending on how specific the category is – for instance, an item could be labeled as a *Vietnamese restaurant*, an *Asian restaurant*, or simply as *Food*. This information is very useful and, as we will show later, exploited in many works (Ying et al., 2012, Zhang and Chow, 2015b), since some users may be more interested in visiting only certain types of POIs while, at the same time, it is not very common for a user to always visit very similar POIs, affecting the recommendations.

On top of this, we may find approaches that make use of the opening and closing times or the time windows or prices of the POIs, since these are particularly important characteristics when creating practical recommendations for users of real systems. However, it should be noted that this type of information is generally considered in works that are evaluated with user studies or mobile apps, or that try to solve a different problem where constraints on the recommendations need to be taken into account (for example, trajectory instead of POI recommendation), and hence, they are less represented in this review because those approaches are out of its scope.

### 3.2.3 Textual reviews

In some LBSNs, users can not only register their check-ins, but also write reviews about the POIs they have visited and exchange this information with other users of the system, either as long, more elaborated texts (as in Yelp) or as short, concise texts (as the so-called *tips* in Foursquare). This type of textual information can be exploited by recommendation approaches and structure this information using topic modeling techniques like Latent Dirichlet Allocation (LDA) or Latent Semantic Analysis (LSA) (Ren et al., 2017). This textual information may provide more useful and high-quality information about the users' interests since, in combination with check-in data, it is possible to capture when the user visited a venue and whether she liked it, together with the reasons about such opinion.

It is important to note that, even though the textual information available from reviews is different than the aforementioned POI features (since such features are intrinsic and static to the items, they do not change, while the reviews represent a subjective opinion from the user perspective), in our classification we will not make a distinction between these two types of information, counting together those works that exploit textual reviews or POIs features.

### 3.2.4 Social links

As we already know from other domains, users tend to be more interested in a product when their friends have some opinion about it; in the same way, this type of information may influence users when receiving POI recommendations. Because of this, some approaches exploit such social links when predicting the user preferences, for instance, by replacing the collaborative neighborhood in classical CF methods with those users who have some social relation with the user (Ye et al., 2011, Cheng and Chang, 2013), or by building social graphs between the users in the system (Wang et al., 2013).

It should be considered, however, that the social links that exist in LBSNs, even though they are usually denoted as "friends", because of the nature of these networks, it is very likely that they do not correspond to friends outside of the system, but similar-minded people or with close tastes, interested in following their opinions or controlling the places they visit. In fact, some datasets that include this information extract it from a different social network (for instance, the global-scale dataset from Foursquare reports friends from Twitter (Yang et al., 2016)), so this information should be exploited with great care.

### 3.2.5 Sequential and temporal information

Apart from the geographical influence, as we stated in Section 3.1, POI recommendation is also affected by the temporal dimension. The temporal context is also vital in this domain, mainly because it affects in a significant way the type of venues that can be visited, but also because users tend to diversify when deciding the next place to visit. Hence, it becomes paramount to know, and to consider in the recommendation process, their previous visits. Similarly, since the user interactions usually have a timestamp associated, it is possible to exploit this data to know the evolution of users' tastes over time; it can also be used to detect the periods of time where some POIs have more activity than others (e.g., bars and restaurants from midday onwards).

For this chapter, we consider that an algorithm uses sequential information if it processes or analyzes the different events when they occur immediately one after the other or if they exploit successive visits to different POIs. At the same time, we assume that a model uses temporal information if they work with the different timestamps of the check-ins or if they use time schedules of the POIs. As pointed out in Section 2.3, sequential and temporal information, although related concepts, they are not the same. For example, once we know a user visited three venues at 4PM, 8PM, and 10AM, we might be tempted to create a sequence of length 3, however, it is very likely that the user stopped to rest during the night, so the sequence should be splitted; the inverse case is more obvious: if we know the sequence followed by a user, it is impossible to recover the exact timestamps unless we know information about the initial time, and the time involved to go from each venue to the next, together with how much time was spent in each of them.

## 3.3 Characterization of POI recommendation works

In this section, we classify existing research works according to six main classes of algorithms, based on the most frequent approaches we have identified in our analysis: based on similarities, factorization models, probabilistic approaches, deep learning techniques, graph- or link-based methods, and hybrid techniques. These categories may or may not use more than one information source among those presented in the previous section, as we shall discuss in detail in Section 3.6. In the following, we describe these categories together with some representative methods from the state-of-the-art reviewed in this chapter.

In order to select the papers we have analyzed, we searched in three digital libraries:

Scopus[1], ScienceDirect[2], and ACM Digital Library[3]. As each library has a different query language to use within its search system, three different queries were needed to be defined and executed, however, they were designed to be as equivalent as possible[4].

The main characteristics these queries should satisfy are:

- Focus on articles between 2011 and 2019 (both included).
- Each publication should include in the title: “Point of interest recommendation” or (similar texts such as “POI suggestion”).
- Each publication should also include somewhere in the title, abstract, or keywords the terms “location-based social network” (or “LBSN”), since this review is oriented to models using data coming from these systems.

Thus, the final queries issued to each source are shown in Table 3.1. Based on this, Table 3.2 shows the number of papers we initially obtained with each query, as well as the actual number we finally analyzed. The difference was mostly caused to some papers not being available, some of them appeared in more than one source, and some had to be filtered out because they address a different task to the one we want to focus here. In the same way, we have found some “repeated” articles that corresponded to improved versions of other works from previous years, conducted by the same authors; in those cases, we have only taken into account the oldest article. For example, LARS* from 2014 and presented in Sarwat et al. (2014) is an extension of LARS, proposed in 2012 by the same authors; to avoid overrepresenting the same methods, we decided to ignore the second paper and only consider the oldest one (as it is the original formulation of the model).

We also decided to keep only those papers whose final goal is to recommend a list of POIs to each user; this includes related tasks such as next-POI recommendation as long as no trajectory or route recommendation is performed (as in Zhang and Wang (2015)), but discards tasks such as route, category, or friend recommendation (Kurashima et al., 2010, Chen et al., 2015, Symeonidis et al., 2011). As a final note, we have not applied additional filters in the paper collection phase such as selection by conferences or journals, to not incur in any subjective bias, although we had to remove few papers where the proposal was not presented in a clear way.

Figure 3.1 shows the number of articles we include in this review according to their publication venue (conference or journal). We observe that the number of publications has increased steadily since 2014; although initially most of the papers were published in conferences, over the years there has been a growing interest in publishing in journals. This figure shows that the problem of POI recommendation is still relevant today. All the works included in our analysis are available as supplementary information[5].

[1]Scopus, https://www.scopus.com/
[2]ScienceDirect, https://www.sciencedirect.com/
[3]ACM, https://dl.acm.org/
[4]The queries were issued last time on January 2020 so some of the results may have changed.
[5]Available here, http://ir.ii.uam.es/~alejandro/poi_survey/index.html.

**Table 3.1:** Queries issued to the three digital libraries considered. For ScienceDirect the query is used in the field “Title, abstract or author-specified keywords”, indicating 2011-2019 in the field “Years”.

| Source | Query |
|---|---|
| Scopus | ( ( TITLE ( point-of-interest ) OR TITLE ( venue ) OR TITLE ( poi ) OR TITLE ( location ) ) AND ( TITLE ( recommendation ) OR TITLE ( recommender ) ) AND ( TITLE-ABS-KEY ( lbsn ) OR TITLE-ABS-KEY ( ”location-based social network” ) ) AND ( PUBYEAR > 2010 ) ) AND ( PUBYEAR < 2020 ) AND NOT TITLE ( survey ) AND ( LIMIT-TO ( LANGUAGE , ”English” ) ) |
| ScienceDirect | ((lbsn) OR (”location-based social network”)) AND -survey AND (”point of interest” OR venue OR location OR poi) AND (recommendation OR recommender) |
| ACM | [[Publication Title: ”point-of-interest”] OR [Publication Title: ”point of interest”] OR [Publication Title: poi] OR [Publication Title: venue] OR [Publication Title: location]] AND [[Publication Title: recommendation] OR [Publication Title: recommender]] AND [[Abstract: ”location-based social network”] OR [Abstract: ”lbsn”]] AND [Publication Date: (01/01/2011 TO 12/31/2019)] |

**Table 3.2:** Papers retrieved and final papers processed from the three digital libraries considered.

| Source | Papers retrieved | Valid papers |
|---|---|---|
| Scopus | 321 | 238 |
| ScienceDirect | 36 | 22 |
| ACM | 46 | 24 |
| Unique papers | 347 | **244** |

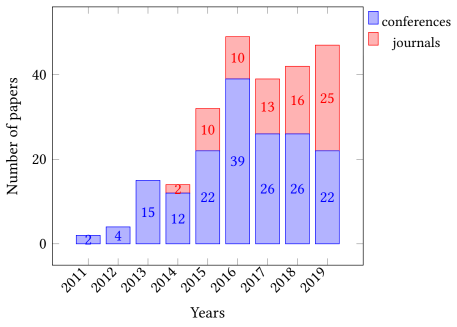


**Figure 3.1:** Number of papers considered in this study based on their publication venue by year.

### Difference with previous surveys on this domain

Due to the growing interest in the general recommendation area on this domain, there is a considerable number of surveys related to POI recommendation and its different ramifications that complement our work. On the one hand, we have the works of Symeonidis et al. (2014), Yu and Chen (2015), Bao et al. (2015) which cannot be considered to be up to date anymore, since they were published more then 5 years ago, thus our review should provide a novel overview of the works developed in this time. On the other hand, Gavalas et al. (2014) presented an overview of optimization approaches that aim to solve a problem with applications on related tasks: the Tourist Trip Design Problem (TTDP); this can be applicable to route recommendation, which, as we specified in the next section, is not completely in the scope of this survey.

Additionally, we found some surveys that were too focused on specific subproblems. For instance, Zheng et al. (2018) considers the problem of location prediction but only based on Twitter information. Another example is Christoforidis et al. (2019), where the authors focus on deep learning techniques while neglecting the other types of recommendation algorithms.

However, even after filtering those works, there are still several reviews dedicated to the problem of POI recommendation based on LBSNs data, as we address herein. Our main differences with the following works is the comprehensive analysis that we present, involving more than 240 papers from 9 years categorized according to their algorithmic and evaluation models. For instance, in Ding et al. (2018a), it is presented a survey where the authors group the articles according to different recommendation objectives, such as location, trip, or activity recommendation, among others. In Zhao et al. (2018a), an overview of the problem from the geographical and temporal perspectives is provided, but then they focus on two specific algorithms.

Besides, since our analysis is also tailored towards the evaluation aspects of the works, it is worth mentioning those reviews where this aspect has been considered. However, we must acknowledge that we could not find any survey that focused on this particular aspect; because of that, we consider our survey is very valuable in this domain at the moment. To somehow overcome this shortcoming, we resort to reporting the experimental comparison presented by Liu et al. (2017), where they compared 12 recommendation models under different evaluation protocols and using three datasets, which could help to analyze the behavior of those methods under the same and different conditions.

We now show the different types of POI recommendation algorithms that we have been able to characterize in this review.

### 3.3.1 Based on similarities

These algorithms correspond to the classic $k$-NN approaches explained in Section 2.2.2, where researchers use similarities between users or items like the well-known cosine similarity. Due to the additional information available in this domain, some authors incorporate a temporal decay in the formulation or even use similarities based on the geographical distance between items. For example, the UTE+SE approach from Yuan et al. (2013) divides the check-in matrix in different time slots and uses it in a user neighborhood CF model; however, since this increases the data sparsity, the authors add a term in the prediction score to account for the similarity between time slots. Nevertheless, as social links between users are often available, instead of calculating similarities between users, in some works, they use the friends of the target user as "nearest neighbors" as they assume that friends in this type of networks may have common interests. This is exactly what is used in LBSMF, the work proposed in Yang et al. (2013), where besides exploiting the sentiment of the reviews of the users, and using a probabilistic MF approach, the users' friends are integrated to compute a social influence term that is later combined to produce the final recommendation score.

It should be noted that in the examined works, neighbor-based models are usually an intermediate phase of a more complex algorithm. That is why we decided to extend the category beyond $k$-NN approaches to consider any proposal that use similarities between items/users and/or use these similarities to establish relationships between them. As a particular example, the LARS approach proposed in Levandoski et al. (2012) would fit in this category, since it takes into account two different similarity spaces: preference locality (users in the same region tend to have similar tastes) and travel locality (users tend to travel short distances when visiting the venues of a region).

### 3.3.2 Factorization

The basic premise of this family of algorithms is to decompose the check-in matrix $\mathbf{C} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$ into two matrices, one for users $\mathbf{U} \in \mathbb{R}^{|\mathcal{U}| \times K}$ and one for POIs $\mathbf{L} \in \mathbb{R}^{|\mathcal{I}| \times K}$, with $K$ being the number of latent factors. We want to note we did not name this class of techniques as the most frequent name *matrix factorization*, because algorithms using tensor factorization (where the additional dimension is used to model time or geographical information) or other types of latent factor models also fit in this category.

Recommendation approaches that belong to this type include GT-BNMF, proposed in Liu et al. (2013a), which is a geographical probabilistic factor analysis framework that takes into account the geographical influence and the textual information of the POIs, to avoid limitations from pure collaborative information such as the cold-start problem. In fact, factorization approaches that exploit the geographical information are very frequent in this domain, as this is a critical information source. The following three models have become state-of-the-art baselines because of their popularity in the area. First, GeoMF, a weighted factorization model proposed in Lian et al. (2014) that divides the full geographical space into different grids to model the following influences: user activity areas and POI influence areas. Second, IRenMF from Liu et al. (2014) incorporates geographical information in the form of neighboring POIs of the target item by exploiting two types of influences: the instance level influence (assuming users tend to visit neighboring locations) and region level influence (to capture user preferences that are shared in the same geographical region). Third, RankGeoFM from Li et al. (2015a) is a geographical factorization method that incorporates the influence of the neighboring POIs of the target item by including a distance weight in the optimization formula. Other methods like GeoIE proposed in Wang et al. (2018b) also incorporate geographical influence, but in this case a power law distribution is used to consider that POIs that are far from other POIs in the system are less likely to be selected. A tensor model is introduced in He et al. (2016), where the authors apply factorization techniques to transition tensors so that transitions between consecutive POIs are modeled, together with a geographical preference term so that far away POIs are less likely to be selected, just as in the previous approach.

The temporal dimension is exploited in the LRT model from Gao et al. (2013). This algorithm is a matrix factorization model that incorporates the temporal effects of the POIs by considering two properties: non-uniformness (the users have different preferences during the day) and consecutiveness (users tend to have similar preferences in consecutive hours). STELLAR, the model proposed in Zhao et al. (2016a), is a time-aware successive POI recommendation model by using a four-tuple tensor while adapting the BPR optimization criteria from Rendle et al. (2009). Geo-Teaser as proposed in Zhao et al. (2017a), on the other hand, combines two different models: a temporal POI embedding for sequential influence that differentiates between weekday and weekends, and a hierarchical pairwise preference ranking model based on BPR that discriminates POIs based on the distance between them.

Social information has also been used in factorization techniques. For instance, TenMF from Yao et al. (2018) is a tensor factorization approach (integrating users, venues, and time frames) that incorporates spatial and social influences in the regularization terms. GeoEISo is an MF approach based on the SVD++ model proposed in Gao et al. (2018a) that incorporates both geographical and social influence (in particular, the trust relationships between the users). The model TGSC-PMF proposed in Ren et al. (2017) also combines different information sources, since its probabilistic matrix factorization method exploits categorical and textual information by using an LDA technique, geographical information with a kernel density estimation, and social information using a power-law distribution.

Categorical or content information, as in the last method described, is easy to be integrated in factorization methods. For instance, CAPRF is proposed in Gao et al. (2015a) where besides the user and POI latent matrices, it incorporates the content and sentiment analysis obtained from the user tips. In a more complex method, ASMF merges social, geographical, and categorical influences, by exploiting check-ins of social, location, and neighboring friends in order to learn the potential locations to recommend, and using an additional score based on a distance distribution between the users's home and their actual check-ins to model the geographical influence, while the categorical information is considered through an additional weight in the recommendation score.

### 3.3.3 Probabilistic

Probabilistic approaches typically consider several random variables that might be related according to some laws or formulations, which in recommendation usually involve users, items, and the potential interaction between the former and the latter. Probabilistic graphical models are one of the most useful frameworks that allow to encode these probability distributions over arbitrary domains, however it is possible to also define simple probability models just by applying Naïve Bayes or other simple approximations with strong (and probably not too realistic) independence assumptions. Besides those techniques that match these formulations, we also extend our categorization as probabilistic to any model that uses some kind of probabilistic distribution in its algorithms to represent or process the data.

In this sense, for example, we consider that those approaches that model the geographic influence by means of power-law distributions such as Wang et al. (2018b) and Ren et al. (2017), those that make use of the Kernel Density Estimation (KDE) like Zhang et al. (2014), or those using bayesian algorithms in the inference or in the optimization steps as in Li et al. (2015b) fit into this category. Another example can be found in WWO from Liu et al. (2016a), which is a model that exploits the sequential preferences of the users to recommend POIs within a time duration; for this, it estimates the distribution of the temporal intervals and creates a low-rank graph to deal with the sparse conditions of the data.

It is important to mention that many proposals can be classified as members of the probabilistic and factorization categories, such as probabilistic matrix factorization (PMF) or some formal topic modeling algorithms, like latent Dirichlet allocation (LDA); a couple of examples can be found in Guo et al. (2015a) and Ren et al. (2017). For instance, LA-LDA is a location-aware probabilistic generative model proposed in Yin et al. (2015) composed of two parts inspired by LDA: ULA-LDA (that exploits the preference locality of the users and assumes each user is affected by her own interests but also by other users in the same home location) and ILA-LDA (that exploits the geographical clustering and the category locality assuming that the POIs that are close to each other should appear in the same topic).

In the same way, we consider proposals based on Markov Chains (MC) as probabilistic since they model the probability of going to the next POI using the immediately previous visited POIs. In fact, this is one of the most popular approaches because of its simplicity and expressiveness. For example, Cheng et al. (2013) propose FPMC-LR, an approach that makes use of Factorized Personalized Markov Chains (FPMC) but adding physical restrictions: instead of building the entire transition tensor, only neighbor POIs are considered after dividing the Earth in different grids; then, a modified version of the BPR optimization technique is used to take into account the sequential components. PRME-G is a next-POI metric embedding method proposed in Feng et al. (2015) that models the sequential influence by borrowing ideas from Markov Chains (MC): instead of computing the transition probabilities by counting, they are estimated by computing the Euclidean distance of the POIs in a latent space. An MC model is also used in Tang et al. (2019), although now to alleviate the sparsity problem; the final probability of a user visiting the target POI also takes into account the influence of communities formed by other users. A more complex method is proposed in Ying et al. (2019), where the approach called MEAP-T considers the sequential component between the POIs (using a first-order MC) and also the temporal influence by modeling the periodicity and the time intervals between the POIs; then, the user preferences, POI transitions, and POI and temporal relationships are transformed into three latent spaces, while exploiting the Euclidean distance and using BPR as optimization criterion.

### 3.3.4 Deep Learning

Deep Learning encompass a set of techniques from the Machine Learning area. While they emerged throughout the 20th century, in the area of recommendation their popularity has grown in the last 10 years. When processing and learning from the data, these types of techniques make use of layers of artificial neurons in order to obtain different representations of the data by optimizing a differentiable function. Although there are many types of neural networks, some of the best known are (Zhang et al., 2019a): the

Multilayer Perceptron (MLP), that is the most basic neural network composed by one or more hidden layers between the output and the input layer using different activation functions in each neuron; the Autoencoder (AE) and Variational Autoencoder (VAE), that are unsupervised techniques oriented as compressing and then rebuilding the original data (VAE also assumes that the input data follows a probability distribution and tries to learn the parameters of that distribution); Convolutional Neural Networks (CNN), oriented at processing images using pooling operations and convolutional layers; and Recurrent Neural Networks (RNN) that memorize previous computations for processing sequential information. As we shall see later, these approaches for POI recommendation have become paramount in the last 3 years; some paradigmatic examples are the following. First, PACE is a deep neural network technique proposed in Yang et al. (2017a), where an architecture with three main components is presented: an embedding layer that takes as inputs the embeddings of the POI and user), the context layer used for context prediction, and the preference layer composed by multiple feedforward layers. Second, VPOI from Wang et al. (2017a) is one of the few approaches that use images for POI recommendation, because of this, here the authors use CNNs to extract the visual contents from the images and which are later exploited in the learning process. Another example is CARA, an approach based on RNNs proposed in Manotumruksa et al. (2018), that consists in two gating mechanisms: the first one to control the influence of ordinary contexts and the second one to model the sequential influence by analyzing time intervals and geographical distance between successive check-ins.

Other deep learning techniques that have been used more recently in the area of POI recommendation are embeddings, specifically graph and word embeddings. The latter consists of learning a latent representation of the words so that those that have a similar meaning also have a similar representation (Naili et al., 2017), while in graph embeddings the objective is to transform a graph into one or more $d$-dimensional vectors which preserve the graph information as much as possible (Cai et al., 2018). Nevertheless, other techniques such as matrix factorization are used to learn these embeddings, so in this review we will include these proposals within the family of deep learning techniques or factorization depending on the specific case. One example from the POI recommendation domain is STA from Qian et al. (2019), where the authors define a graph embedding approach that incorporates both temporal and geographical information.

### 3.3.5 Graph/Link

Link-based or graph-based techniques build one or more graphs using the data stored in the system, which in our case is a LBSN. They typically consider the users or POIs as nodes, and exploit various influences (e.g., geographical, social, temporal, etc.) to create and weight links between these nodes. There are a great number of models based on graphs, among which the following are the most used in POI recommendation approaches: Random Walk, Hypertext Induced Topic Selection (HITS), PageRank, etc. However, as we shall see in the next sections, its popularity in the area of POI recommendation is not as high as other mechanisms like factorization techniques.

Among the few (representative) examples we have found in the literature, the following is a paradigmatic example of how this type of methods are used. In Noulas et al. (2012), a Random Walk approach is proposed, where a graph is built in which both the venues and users are nodes of the graphs, and where a link exists between a user and an item whenever the user has checked-in in that item; additionally, users are linked to each other based on their social relationships. In the model proposed in Yuan et al. (2014), GTAG, also two types of links are used, but considering different information: geographical and temporal influence; on the one hand, POIs are connected by distance to the nearest venues and weighted according to a power-law distribution, on the other hand, users and POIs are connected according to sessions defined based on their check-ins and using an exponential function to weight the edges to account for the temporal influence; with all this information, a Breadth-first Preference Propagation algorithm is used.

Thanks to the flexibility of these models, they can exploit almost any type of information source. For instance, Bao et al. (2012) propose an online POI recommendation model where a weighted categorical tree is built for each user, where a HITS-based approach is used to obtain local experts, which are later used to produce recommendations. A more complex approach is found in UPOI-Walk, where a Dynamic HITS-based Random Walk model is proposed in Ying et al. (2014) that combines several relationships captured in the data: popularity (between POIs and check-ins), social (between POIs and users' social circles), and categorical (between semantic labels and user preferences).

### 3.3.6 Hybrid

Contrary to the more classical understanding of how hybrid methods are defined (Burke, 2007) (see Section 2.2.3), in this review we do not classify approaches using and combining several components within the same algorithm as such – for example, adding a similarity computation in a matrix factorization formulation or using a matrix factorization algorithm in a more complex deep learning model. We decided to follow this procedure because, as we have discussed previously and it can be observed in most of the examples shown before, most algorithms combine several sources (geographical, temporal, sequential, social, etc.) in different ways, in such a way that if we took the more traditional and strict definition of hybrid recommender, almost every recommendation approach could be considered as such.

In the following, we present some of the approaches that we do consider as hybrids, starting with those that integrate social information, since it was identified in many hybrid methods. One possible reason for this is that this information source cannot be easily modeled under a unified framework together with other sources due to its different nature, hence, it needs tailored combinations or aggregations as the ones we present next. For example, the UPOI-Mine approach proposed in Ying et al. (2012) besides considering the individual preferences of the users (through the tags of the previously visited POIs), it exploits the social information from the target user friends and uses the popularity of the venues in order to counter the data sparsity; all of this is then combined into a regression tree model, focused on predicting the next restaurant to visit, instead of general POIs. On the other hand, the USG model proposed in Ye et al. (2011) combines three different scores: user preferences, social information through a combination of the classical user-based CF formulation, and geographical influence with a power-law distribution. Similarly, various information sources are combined in Geng et al. (2019), although in this case the authors model POI recommendation as a multi-objective optimization problem considering social, geographical, and user similarity influences.

As in the previously described method, we found several approaches where two other sources of information besides social are exploited. LORE is a method proposed in Zhang et al. (2014) that combines social (by computing similarities between friends), geographical (using a two-dimensional Kernel Density Estimation), and sequential (with an additive Markov Chain trained with the transition probabilities between all the users) information. Categorical information is exploited in GeoSoCa, a model proposed in Zhang and Chow (2015b) where social, geographical, and categorical influences are combined, using a power-law distribution for the first and last models, whereas a similar method to the one described in LORE is used for the geographical one.

Besides social information, geographical (as already discussed in other parts of this review) is another source that is exploited frequently; indeed, this is evidenced in the following hybrid approaches which exploit this type of signal among others. The so-called LBPR method from He et al. (2017a) adapts the BPR technique to predict the next category and then obtain a ranking of POIs using the predicted category and after incorporating a geographical score for the candidate POIs; the main difference with other approaches based on BPR is that this model uses lists of categories instead of category pairs in the learning step. The APRA-SA model as proposed in Si et al. (2019), on the other hand, takes into account geographical and temporal information by computing the popularity of the POIs in different time periods and using a Kernel Density Estimation component. Finally, the GE method proposed in Xie et al. (2016a) consists of a graph-based embedding model where four types of graphs are considered: a POI-POI graph (to capture the check-in sequences of POIs), a POI-region graph (to exploit the geographical information), a POI-time graph (for temporal and cyclic behaviour), and a POI-word graph (to exploit semantics).

## 3.4 Characterization of evaluation methodologies in POI recommendation

The evaluation methodologies used in the POI recommendation domain are not too different from those traditionally used in classical recommendation and presented in Section 2.4. However, considering the importance some dimensions have in this domain – i.e., time and geographical information, mostly – we describe now in more detail those time-aware evaluation methodologies used in the area (Campos et al., 2014), together with some variations inherent to the POI recommendation problem.

**Table 3.3:** Evaluation protocols characterized by their application level and the type of split; where X+Y denotes the type (X, either Random or Temporal) and application level (Y, either System or User) of the split, and ✓, ✗, and **?** indicate if that characteristic is known to occur, never occur, or it is impossible to know in a protocol.

| Characteristics | R+S | R+U | T+S | T+U |
|---|---|---|---|---|
| All users are evaluated | **?** | ✓ | **?** | ✓ |
| Leaks future data in test from training | Very likely | Very likely | ✗ | For some users |
| Test contains recent check-ins | **?** | **?** | ✓ | ✓ |
| Test is made as a random subset | ✓ | ✓ | ✗ | ✗ |

As explained in Section 2.4.2, the first step in any offline evaluation is to divide the available data into different sets: at least training and test, although an additional validation set is preferred to tune the parameters of the models and not overfit the test set. How the original data is divided is critical to imitate the use of the recommendation algorithm in a real scenario, that is why in the POI recommendation scenario we observe that the time dimension is often used when splitting the dataset, even though these methodologies were already used and formalized in the area, nonetheless, random partitions of the data are still very popular (Campos et al., 2014).

More specifically, we consider a split is *temporal* whenever the check-ins are ordered according to the temporal dimension (either by the actual timestamps or because there is some sequential information in the data) so that the check-ins in the test set are more recent than those in the training set; otherwise we consider the split is *random*, including the cross-validation setting presented in Section 2.4.2. An additional criteria that may have a great impact on the final results is whether the split is done at the system or per user level (see Section 2.4.2); this reflects whether the previous criteria (temporal or random) is applied to the whole dataset or in a user basis – i.e., for each user independently. This criteria affects the number and type of users that belong to each set, since performing a per user split would guarantee that all the users exist in both training and test sets. Finally, a parameter that defines different variations of these protocols is whether the subset is selected according to a percentage or ratio between training and test – typical values are 80% for training and the rest for test – or based on a fixed number of elements, for example, the last 1 or 2 interactions go into the test set; for the sake of a cleaner presentation we will not consider this parameter in the classification we use in the next sections. We present in Table 3.3 a summary of the implications for the four possibilities regarding data splitting that will be used in the rest of the chapter. Even though we have found some papers where the evaluation was performed in other ways (e.g., temporal windows or splits by distance between locations), most of the analyzed articles fit into the aforementioned evaluation protocol classification. Based on this, we argue that the most realistic scenario is a temporal partition at the system level, as it takes into account the temporal dimension while avoiding any leaking of the user interactions from the future into the training set.

Because of the paramount importance of the geographical dimension, some authors include in their experimental settings variations tailored for the POI recommendation problem. In particular, those approaches that exploit geographical information or neighbor venues tend to filter the data by cities or, in general, by geographical regions, such as country or continent. Another important characteristic of the data produced by LBSNs is that users may visit the POIs more than once, hence, this leads to two ways of producing the splits explained before: at the check-in level (keeping all the check-ins, even the repeated ones) or at POI level (removing the duplicated user-POI pairs and keeping only one instance before running the splitting strategy). These repetitions may have a significant effect in training since it allows to capture item frequencies at the user level, but its effect is even more dramatic in the test set, since it may hide the fact that some uninteresting baselines (such as returning those items previously interacted by the user) would perform very well (Sánchez and Bellogín, 2018a).

Finally, regarding the metrics used when evaluating POI Recommender Systems, it should be noted that error-based metrics are not very interesting when the interaction to be predicted is a check-in, since that value is always 1; when the user interaction is different (such as ratings or those described in Section 3.2), then these metrics can be applied, considering the limitations already described in Section 2.4.3. Nonetheless, it is important to mention that in recent years, researchers dealing with the problem of POI recommendation and related tasks (see Section 3.5) have adapted ranking-based accuracy metrics to consider distances between the recommended POIs and the actual order followed by the users in the test set; some examples can be found in Chen et al. (2016a), where the authors use a metric based on $F_1$ that takes into account the pairwise order between POIs, and our work presented in Sánchez and Bellogín (2018a, 2020), where the Longest Common Subsequence algorithm is introduced in ranking-based accuracy metrics to penalize those recommendations less similar with the sequence followed by the user (more details will be given in Section 4.4).

## 3.5 Relation to other recommendation tasks

POI recommendation is not the only task that can be performed using LBSNs, due to the richness of the data of this kind of social network, a large number of related tasks/problems have arisen. Since they are not in the focus of this review, we discuss them briefly now:

- Trajectory (or route) recommendation: typically, POI recommendation approaches provide each user with a list of POIs that hopefully may be of interest; however, there is generally no intrinsic relationship between these recommended POIs. Instead, in route recommendation, a complete trajectory is generated and provided to the target user. Because of this, additional restrictions must be taken into account, such as the duration or length of the route or the schedule of the venues (Chen et al., 2016a).

- Friend recommendation: this is a well studied problem in the context of traditional social networks, like Twitter or Facebook. Considering the importance of the social dimension in LBSNs in general, and of social information for POI

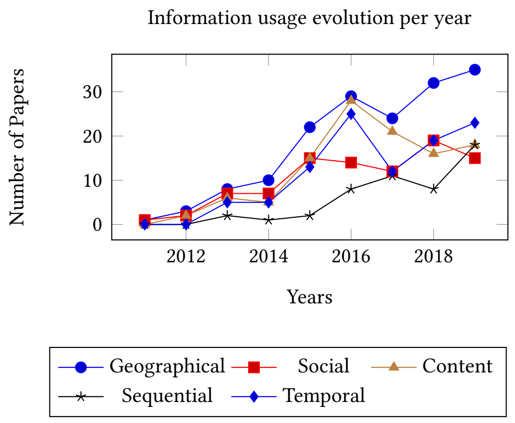


**Figure 3.2:** Number of papers using different information sources by year.

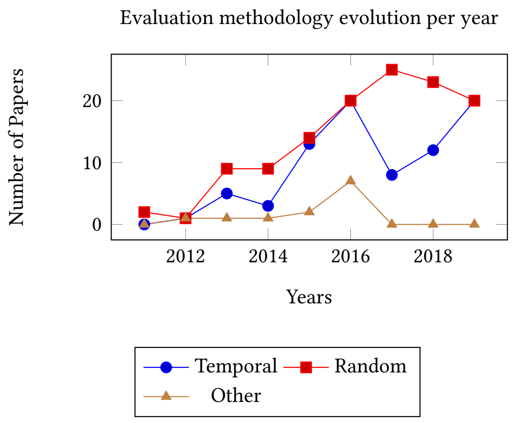


**Figure 3.3:** Number of papers using different evaluation methodologies by year.

recommendation (discussed in previous sections), there is an increasing interest in this problem by the community like the approaches from Bagci and Karagoz (2016) and Chu et al. (2013).

- Group recommendation: quite frequently users visit a city in groups, either composed by friends, family, or even as organized tours. In this case, instead of recommending POIs to a unique target user, the algorithms should be tailored to groups of users (Ayala-Gómez et al., 2017, Purushotham et al., 2014). This problem needs to take into account additional factors, such as the difference between passive and active users in such groups, or the balance between individual preferences.

## 3.6 Systematic review of state-of-the-art algorithms

In this section, we analyze the state-of-the-art algorithms according to the classification presented in Section 3.3 for the papers considered in this study.

First, we selected the *most representative* papers for each year and include their whole characterization in Table 3.4. When selecting the most representative papers, we considered the top-5 most cited articles per year according to Scopus with at least one citation. Nevertheless, the full tables analyzing all the papers reported in Table 3.2 can be seen in Tables A.1, A.2, A.3 and A.4, located in Appendix A.

We also include in Tables 3.4 and 3.5 two summary rows that count how many papers (among the sets of most representative or the entire collection) satisfy each condition in Table 3.4. Each of the conditions (columns) correspond to the categories described in Sections 3.2, 3.3, and 3.4, respectively.

Based on these tables, let us first analyze the trends on information sources used throughout the years. We observe that most of the algorithms make use of geographic information in some way (either by calculating distances between POIs, grouping users

**Table 3.4:** Summary of analyzed POI recommendation approaches sorted by publication year. The ✓mark denotes that the proposed model has the feature indicated in the column, whereas (N.A.) shows that no acronym was given.

| Details | | | Information used | | | | | Model | | | | | | | Split type | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Year | Reference | Acronym | Geographical | Social | Content | Sequential | Temporal | CF sims | Factorization | Probabilistic | DNN | Graph/Link | Hybrid | Other | Random | Temporal | Other |
| 2011 | Ye et al. (2011) | USG | ✓ | ✓ | | | | ✓ | | ✓ | | | ✓ | | ✓ | | |
| 2012 | Levandoski et al. (2012) | LARS | ✓ | | | | | ✓ | | | | | | | ✓ | | |
| 2012 | Bao et al. (2012) | (N.A.) | ✓ | | ✓ | | | ✓ | | | | ✓ | | | | | ✓ |
| 2012 | Ying et al. (2012) | UPOI-Mine | ✓ | ✓ | ✓ | | | ✓ | | | | | ✓ | | | | |
| 2012 | Noulas et al. (2012) | RW, Weighted-RW | | ✓ | | | | | | ✓ | | ✓ | | | | ✓ | |
| 2013 | Yang et al. (2013) | LBSMF | | ✓ | ✓ | | | ✓ | ✓ | ✓ | | | | | ✓ | | |
| 2013 | Liu et al. (2013a) | GT-BNMF | ✓ | | ✓ | | | | ✓ | ✓ | | | | | ✓ | | |
| 2013 | Cheng et al. (2013) | FPMC-LR | ✓ | | | ✓ | | | ✓ | ✓ | | | | | | ✓ | |
| 2013 | Gao et al. (2013) | LRT | | | | | ✓ | | ✓ | | | | | | ✓ | | |
| 2013 | Yuan et al. (2013) | UTE+SE | ✓ | | | | ✓ | ✓ | | ✓ | | | ✓ | | ✓ | | |
| 2014 | Ying et al. (2014) | UPOI-Walk | ✓ | ✓ | ✓ | | | | | | | ✓ | | | | | |
| 2014 | Yuan et al. (2014) | GTAG | ✓ | | | | ✓ | | | | | ✓ | | | ✓ | | |
| 2014 | Lian et al. (2014) | GeoMF | ✓ | | | | | | ✓ | | | | | | ✓ | | |
| 2014 | Liu et al. (2014) | IRenMF | ✓ | | | | | | ✓ | | | | | | ✓ | | |
| 2014 | Zhang et al. (2014) | LORE | ✓ | ✓ | | ✓ | | ✓ | | ✓ | | | ✓ | | | ✓ | |
| 2015 | Yin et al. (2015) | LA-LDA | ✓ | | ✓ | | | | ✓ | ✓ | | | | | ✓ | | |
| 2015 | Li et al. (2015a) | RankGeoFM | ✓ | | | | ✓ | | ✓ | | | | | | | ✓ | |
| 2015 | Zhang and Chow (2015b) | GeoSoCa | ✓ | ✓ | ✓ | | | | | ✓ | | | ✓ | | | ✓ | |
| 2015 | Feng et al. (2015) | PRME-G | ✓ | | | ✓ | ✓ | | ✓ | ✓ | | | | | | ✓ | |
| 2015 | Gao et al. (2015a) | CAPRF | | | ✓ | | | | ✓ | | | | | | ✓ | | |
| 2016 | Xie et al. (2016a) | GE | ✓ | | ✓ | ✓ | ✓ | | ✓ | ✓ | | | ✓ | | | ✓ | |
| 2016 | Li et al. (2016a) | ASMF | ✓ | ✓ | ✓ | | | | ✓ | | | ✓ | | | | ✓ | |
| 2016 | Zhao et al. (2016a) | STELLAR | | | | | ✓ | | ✓ | | | | | | | ✓ | |
| 2016 | He et al. (2016) | (N.A.) | ✓ | | | ✓ | | | ✓ | ✓ | | | | | | | |
| 2016 | Liu et al. (2016a) | WWO | | | | ✓ | ✓ | | | ✓ | | | | | | ✓ | |
| 2017 | Zhao et al. (2017a) | Geo-Teaser | ✓ | | | ✓ | ✓ | | ✓ | ✓ | ✓ | | | | | ✓ | |
| 2017 | Yang et al. (2017a) | PACE | ✓ | ✓ | | | | | | | ✓ | | | | | ✓ | |
| 2017 | Ren et al. (2017) | TGSC-PMF | ✓ | ✓ | ✓ | | | | ✓ | ✓ | | | | | ✓ | | |
| 2017 | He et al. (2017a) | LBPR | ✓ | | ✓ | ✓ | | | ✓ | ✓ | | | ✓ | | | ✓ | |
| 2017 | Wang et al. (2017a) | VPOI | | | ✓ | | | | ✓ | ✓ | ✓ | | | | ✓ | | |
| 2018 | Ma et al. (2018) | SAE-NAD | ✓ | | | | | | | | ✓ | | | | ✓ | | |
| 2018 | Yao et al. (2018) | TenMF | ✓ | ✓ | | | ✓ | | ✓ | | | | | | ✓ | | |
| 2018 | Manotumruksa et al. (2018) | CARA | ✓ | | | ✓ | ✓ | | | | ✓ | | | | | ✓ | |
| 2018 | Gao et al. (2018a) | GeoEISo | ✓ | ✓ | | | | | ✓ | ✓ | | | | | ✓ | | |
| 2018 | Wang et al. (2018b) | GeoIE | ✓ | | | | | | ✓ | ✓ | | | | | | ✓ | |
| 2019 | Ying et al. (2019) | MEAP-T | | | | ✓ | ✓ | | ✓ | ✓ | | | | | | ✓ | |
| 2019 | Geng et al. (2019) | MLR | ✓ | ✓ | | | | ✓ | | ✓ | | | | ✓ | ✓ | | |
| 2019 | Tang et al. (2019) | (N.A.) | | ✓ | ✓ | ✓ | | ✓ | | ✓ | | | ✓ | | | | |
| 2019 | Si et al. (2019) | APRA-SA | ✓ | | | | ✓ | | | ✓ | | | ✓ | | ✓ | | |
| 2019 | Qian et al. (2019) | STA | ✓ | | | | ✓ | ✓ | ✓ | | | | | | | ✓ | |
| Most Representatives | | | 31 | 14 | 14 | 11 | 14 | 10 | 23 | 23 | 5 | 5 | 9 | 1 | 18 | 17 | 1 |
| Total | | | 164 | 92 | 111 | 50 | 102 | 74 | 118 | 110 | 35 | 37 | 79 | 17 | 123 | 82 | 14 |

**Table 3.5:** Evaluation details of analyzed POI recommendation approaches sorted by publication year.

| Details | | | Evaluation configuration | | | | | | | Baselines | | | Split type | | | Split level | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Year | Reference | Acronym | Filter data | Validation | Error | Ranking | Region Split | Check-in(✓) or POI(✗) split | Cold Start Analysis | Classic Non Personalized | Classic Personalized | Geographical | Random | Temporal | Other | system | per user |
| 2011 | Ye et al. (2011) | USG | | | | ✓ | | ✗ | ✓ | | ✓ | ✓ | ✓ | | | | ✓ |
| 2012 | Levandoski et al. (2012) | LARS | | | | | ✓ | ✗ | | | ✓ | ✓ | ✓ | | | ✓ | |
| 2012 | Bao et al. (2012) | (N.A.) | ✓ | | | ✓ | ✓ | ✗ | | | ✓ | | | | ✓ | | |
| 2012 | Ying et al. (2012) | UPOI-Mine | | | ✓ | ✓ | | ✗ | | | ✓ | ✓ | | | | | |
| 2012 | Noulas et al. (2012) | RW, Weighted-RW | | | | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | | ✓ | | ✓ | |
| 2013 | Yang et al. (2013) | LBSMF | ✓ | | ✓ | | ✓ | ✓ | | | ✓ | | ✓ | | | ✓ | |
| 2013 | Liu et al. (2013a) | GT-BNMF | | | | ✓ | ✓ | ✓ | | | ✓ | | ✓ | | | ✓ | |
| 2013 | Cheng et al. (2013) | FPMC-LR | ✓ | | | ✓ | | ✓ | | | | | | ✓ | | ✓ | |
| 2013 | Gao et al. (2013) | LRT | ✓ | | | ✓ | | ✗ | | | ✓ | | ✓ | | | | ✓ |
| 2013 | Yuan et al. (2013) | UTE+SE | ✓ | ✓ | | ✓ | ✓ | ✗ | | | ✓ | ✓ | ✓ | | | | ✓ |
| 2014 | Ying et al. (2014) | UPOI-Walk | | | ✓ | ✓ | ✓ | ? | | | ✓ | ✓ | | | | | |
| 2014 | Yuan et al. (2014) | GTAG | ✓ | ✓ | | ✓ | ✓ | ✗ | | | ✓ | ✓ | ✓ | | | | ✓ |
| 2014 | Lian et al. (2014) | GeoMF | ✓ | | | ✓ | | ✗ | | | ✓ | | ✓ | | | | ✓ |
| 2014 | Liu et al. (2014) | IRenMF | | ✓ | | ✓ | ✓ | ✗ | | | ✓ | ✓ | ✓ | | | | ✓ |
| 2014 | Zhang et al. (2014) | LORE | | | | ✓ | | ✓ | | | | ✓ | | ✓ | | ✓ | |
| 2015 | Yin et al. (2015) | LA-LDA | | ✓ | | ✓ | | ✗ | ✓ | | | ✓ | ✓ | | | | ✓ |
| 2015 | Li et al. (2015a) | RankGeoFM | | ✓ | | ✓ | ✓ | ✓ | | | ✓ | ✓ | | ✓ | | | ✓ |
| 2015 | Zhang and Chow (2015b) | GeoSoCa | | | | ✓ | ✓ | ✓ | | | | ✓ | | ✓ | | ✓ | |
| 2015 | Feng et al. (2015) | PRME-G | ✓ | ✓ | | ✓ | ✓ | ✓ | | ✓ | ✓ | | | ✓ | | ✓ | |
| 2015 | Gao et al. (2015a) | CAPRF | ✓ | | | ✓ | ✓ | ✗ | | | ✓ | | ✓ | | | | ✓ |
| 2016 | Xie et al. (2016a) | GE | | ✓ | | ✓ | ✓ | ✓ | ✓ | | | ✓ | | ✓ | | | ✓ |
| 2016 | Li et al. (2016a) | ASMF | ✓ | | | ✓ | ✓ | ✗ | ✓ | | ✓ | ✓ | | ✓ | | | ✓ |
| 2016 | Zhao et al. (2016a) | STELLAR | ✓ | | | ✓ | | ✓ | | | ✓ | | | ✓ | | ✓ | |
| 2016 | He et al. (2016) | (N.A.) | ✓ | | | ✓ | ✓ | ? | | | ✓ | | | | | | |
| 2016 | Liu et al. (2016a) | WWO | ✓ | | | ✓ | | ✓ | | | | | | ✓ | | ✓ | |
| 2017 | Zhao et al. (2017a) | Geo-Teaser | ✓ | | | ✓ | | ✓ | | | ✓ | ✓ | | ✓ | | | ✓ |
| 2017 | Yang et al. (2017a) | PACE | ✓ | | | ✓ | | ✓ | | | | ✓ | | ✓ | | | ✓ |
| 2017 | Ren et al. (2017) | TGSC-PMF | ✓ | | ✓ | ✓ | | ✓ | | | | ✓ | ✓ | | | ✓ | |
| 2017 | He et al. (2017a) | LBPR | | | | ✓ | ✓ | ✓ | | | ✓ | ✓ | | ✓ | | | ✓ |
| 2017 | Wang et al. (2017a) | VPOI | ✓ | | | ✓ | ✓ | ✗ | ✓ | | ✓ | | ✓ | | | | ✓ |
| 2018 | Ma et al. (2018) | SAE-NAD | ✓ | | | ✓ | ✓ | ✗ | | | ✓ | ✓ | ✓ | | | | ✓ |
| 2018 | Manotumruksa et al. (2018) | CARA | ✓ | | | ✓ | | ✓ | ✓ | | ✓ | ✓ | | ✓ | | | ✓ |
| 2018 | Yao et al. (2018) | TenMF | ✓ | | | ✓ | | ✗ | | | ✓ | ✓ | ✓ | | | | ✓ |
| 2018 | Gao et al. (2018a) | GeoEISo | ✓ | | | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ | | | ✓ | |
| 2018 | Wang et al. (2018b) | GeoIE | ✓ | ✓ | | ✓ | | ✓ | | | | ✓ | | ✓ | | | ✓ |
| 2019 | Ying et al. (2019) | MEAP-T | ✓ | ✓ | | ✓ | ✓ | ✓ | | ✓ | ✓ | | | ✓ | | | ✓ |
| 2019 | Geng et al. (2019) | MLR | | | | ✓ | ✓ | ✗ | | | ✓ | ✓ | ✓ | | | ✓ | |
| 2019 | Tang et al. (2019) | (N.A.) | ✓ | | | ✓ | ✓ | ? | | | ✓ | | | | | | |
| 2019 | Si et al. (2019) | APRA-SA | | | ✓ | ✓ | | ✓ | | | ✓ | ✓ | ✓ | | | | ✓ |
| 2019 | Qian et al. (2019) | STA | | ✓ | | ✓ | | ✓ | ✓ | | | ✓ | | ✓ | | | ✓ |
| Most Representatives | | | 24 | 10 | 5 | 38 | 23 | C:21 P:16 | 7 | 3 | 30 | 26 | 18 | 17 | 1 | 13 | 22 |
| Total | | | 135 | 37 | 22 | 229 | 135 | C:150 P:66 | 27 | 29 | 147 | 142 | 123 | 82 | 14 | 101 | 104 |

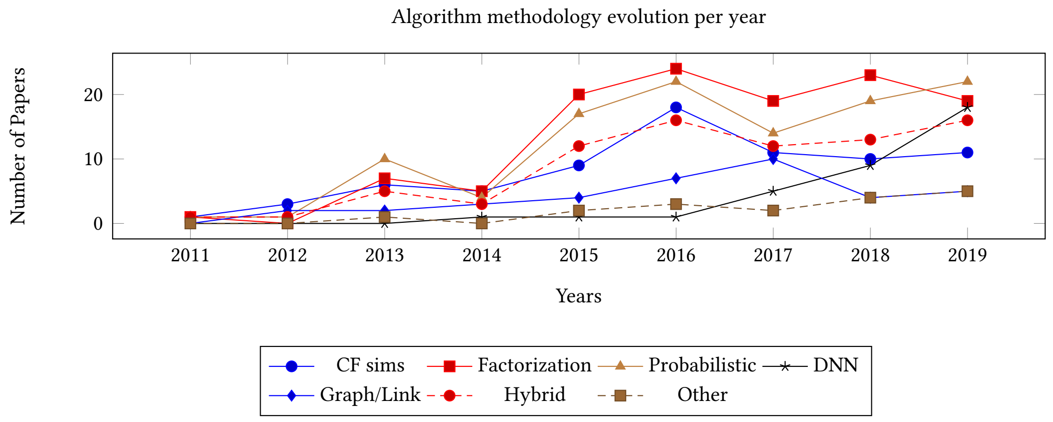


**Figure 3.4:** Number of papers using different algorithms by year.

and POIs in clusters according to regions, modeling movement distributions, etc.). Most researchers argue that this type of information is critical since users tend to visit POIs close to where they are and this conclusion can be obtained by performing a preliminary analysis of the LBSNs data. On the other hand, social information is also widely modeled, partly because many of the datasets used, such as Gowalla, also provide the links of friendship between users. However, while there is more or less consensus on the importance of geographical information, this is not so clear for social information, where some researchers claim it is not so important (Gao et al., 2012, Cheng et al., 2016) while others state it plays an important role (Cheng and Chang, 2013, Gao et al., 2018a). One possible explanation for this effect is that even though users may share their tastes with friends (from the same or different cities), they may not visit the same POIs, in part because it is common for users to visit the locations closest to their centers of activity (home and work, basically) and most likely they will be different from their friends', even if they are "similar" in terms of tastes.

Textual or content information is also exploited by many approaches, especially those using some kind of probabilistic model such as topic modeling or the POI categories. This is because the features of the items (categories) in this domain are very distinctive and may even discriminate between different types of users in a LBSN; for example, a tourist may prefer to visit museums and restaurants, whereas a local may prefer a bar or a shopping center. Finally, regarding temporal and sequential information, we observe that the latter is not so exploited (although some Deep Learning techniques make use of sequential information implicitly), but temporal information is taken into account in many approaches regardless of the technique used by the model under analysis, probably because of its flexibility to be introduced in almost any recommendation technique (usually at the cost of increased sparsity). The same trend can be found in Figure 3.2, where all the selected papers in the review, not only the representative ones, are shown in a year basis.

Now, when we analyze the type of model, a change in trend can be seen in the years between 2011 and 2015 and subsequent years, since for the former, proposals that used some type of collaborative system based on neighbors were reasonably popular, but not anymore. However, in subsequent years there has been a greater dominance of probabilistic and factorization proposals. This is something that already happened in traditional recommendation, where since the Netflix prize in 2009 (Bell and Koren, 2007) in which matrix factorization models outperformed other traditional approaches, they have received more attention from the RS research community. In the same way, Neural Networks techniques have experienced a significant growth since 2017 in the POI recommendation area. This becomes evident in Figure 3.4 where, again, all the papers are included. Here, we observe that until 2017 there are less than 5 Deep Learning techniques included in the selected papers, but this type of model increases steadily year after year. Finally, with respect to graph/link-based and hybrid models we observe that, in general, graph models are not widely used, whereas hybrid techniques, even though they are not widely used, they have been used throughout all the years collected in our analysis. One reasonable assumption for this is that hybrid proposals allow several elements to be combined into one, thus alleviating the possible drawbacks that each of them may show separately.

In the next section, we analyze in detail the evaluation aspects, including the split types that appear in the already discussed tables.

## 3.7 Systematic review of state-of-the-art evaluation methodologies

In this section, we continue the analysis on the state-of-the-art in POI recommendation presented previously but focusing on the evaluation aspects. Thus, we analyze the last columns of Table 3.4, together with Figure 3.3, which shows the characterization of some evaluation protocols, as shown in Section 3.4. We observe they are well distributed between random and temporal splits, although it is interesting to note that until 2014 the random partition predominated over the temporal split. However, from that year onwards the use of temporal splitting has increased steadily.

Nonetheless, there is still no common evaluation protocol to evaluate the performance of POI recommenders; this is an interesting but also a concerning conclusion, since this means that we might be comparing models that try to solve the same problem (POI recommendation) but, at the same time, we are evaluating them in very different ways, which in turn affects the performance of the algorithms. In particular, we have found surprising combinations regarding this, such as works with models using temporal information in their formulation that were running a random evaluation protocol in their evaluation, like Yuan et al. (2013), Gao et al. (2013), Zhao et al. (2016a), Yuan et al. (2014).

In any case, as mentioned in Section 2.4 it should be noted that even if the entire community moves to a common splitting method, there are other aspects of the experimental settings that could affect the final performance of the algorithms and, hence, the validity of the published results.

Because of this, in Table 3.5 we extend the evaluation aspects to be considered for the same works presented before. We now include whether some kind of data filtering is performed (to avoid both users and POIs with very few interactions), if a validation subset is used, the type of metric (error or ranking) reported, if the split was used based on geographical information, and if repetitions or not were considered (i.e., if the split was done by check-ins or POIs), as discussed in Section 3.4. We also decided to include if the authors performed some kind of cold-start analysis and the types of baselines considered in the experiments: whether they use classic non-personalized recommendation baselines (popularity and/or random), classic and customized baselines (UB, IB, BPR, or MFs), and geographical baselines (any algorithm that uses a geographical component).

Based on this information, we first observe that a relatively large number of articles apply some kind of filter in the data, the most typical one being to remove users or POIs with less than $n$ interactions. It is important to note that we only put the mark ✓ if the authors specifically state in their paper that they filter the data, so there might be other proposals that use a pre-filtered dataset that do not count in the table; hence, these numbers are probably underestimating this aspect. Nonetheless, it is true that in some situations it might be necessary to make some pre-filtering of the data, but we must be careful since, if the filtering is too strict, we may end up evaluating the system with very little data, making the obtained results not generalizable. On the other hand, sometimes, instead of performing a simple training and test splits, researchers obtain a third subset of data to tune the parameters (called validation but different from a k-fold cross-validation). However, as we observe in the table, this is not very common in POI recommendation (mirroring what happens in traditional recommendation). With respect to the type of metrics reported, there seems to be more consensus, since the vast majority of papers use some kind of ranking-based accuracy metric. Besides, most approaches that evaluate using rating prediction also use ranking evaluation, although there are very few approaches that only use rating prediction, like Yang et al. (2013).

Regarding the region split column, we believe it is quite important when comparing research works in this domain, since algorithms executed in a worldwide dataset are not comparable against those executed in independent cities, mostly because the geographical influence is indeed affecting the recommenders in a very different way, depending on the type of split we are using. Similarly, depending on whether the split is done by check-ins or by POIs, it might affect the obtained results. Although this distinction might be subtle, if we analyze this aspect we observe there is a lot of disparity in the works, not leading to any clear conclusion. Let us consider for example that we select for each user 80% of their check-ins to train and the rest 20% to test, as we mentioned before, on many LBSNs there may be repetitions so the test set may be composed by check-ins that appear in the training set; however, if the split is made by POIs, we make sure to remove such repetitions and therefore we would not be recommending POIs that the user has already visited. At the same time, even though datasets in this domain are very sparse, few researchers perform a specific analysis on cold start, as evidenced from the values shown in the table (we denote as cold start those works that explicitly consider users or POIs with very few interactions, e.g., less than 5).

Now, in terms of the baselines used, although most of the approaches compare against baselines that can be categorized as classic algorithms such as MFs or $k$-NN, and many others use geographical influence, it is surprising that there are a limited number of works that test their approaches against very basic baselines like popularity, when it has been shown to be quite effective in domains with a high sparsity (Bellogín et al., 2017).

## 3.8 Systematic review of datasets used in state-of-the-art

There are several LBSNs that researchers use to explore the problem of POI recommendation and related tasks, but among all of them, there are four that stand out: Foursquare[6], Gowalla[7], Brightkite[8], and Yelp[9], as it is evidenced by Table 3.6, that shows the number of papers reporting data from each LBSN. Hence, researchers obtained data from these systems and used them for their experiments, even though the same data could be used for different purposes – i.e., not only the pure POI recommendation task we address here, but also for social or review recommendation.

Besides the differences in the actual recommendation task, which might be more or less obvious when comparing two research works, we noticed remarkable differences in the statistics reported for datasets that (in principle) belong to the same LBSN. The reason might be obvious: the datasets are obtained and pre-processed differently, however, since there is no *canonical name* for the datasets (as it occurs in other domains, e.g., with the MovieLens or Lastfm datasets), they are indistinctively referred as the name of the corresponding LBSN, which confuse the reader and other researchers into thinking that the same data is used in two works. To shed some light on this aspect, we now present some details about the most important datasets based on these LBSNs and later analyze some of these differences.

- Foursquare: it is possibly the most famous LBSN, which agrees with our statistics (see Table 3.6) that show it is the LBSN most frequently used by researchers, among the works included in our analysis. Users in Foursquare can visit a place, mark it as visited in the system (by *checking-in* in the venue) so their followers or friends could track it, like a venue, comment on it (by writing *tips*), and obtain recommendations from the system (since 2014 most of this functionality was derived to Swarm). In general, these check-ins cannot be obtained directly neither from its website nor its API; because of that, most researchers rely on other social networks where users share their interactions with Foursquare, mostly Twitter. Even though we will show different datasets from this LBSN in Tables 3.7 and 3.8, we consider important to emphasize that many papers that report using Foursquare include a url[10] that does not work anymore; these include the original

[6]Foursquare, `https://foursquare.com`
[7]It does not exist anymore since 2012.
[8]It was acquired by another social network in 2009 and does not exist anymore since 2012.
[9]Yelp, `https://www.yelp.com`
[10]A Foursquare dataset, `http://www.public.asu.edu/~hgao16/dataset.html`

work Gao et al. (2012) and many more, such as Gao et al. (2013), Zhang et al. (2014, 2015a), Stepan et al. (2016), Hosseini and Li (2016).

- Gowalla: a LBSN that was acquired by Facebook in 2011. Most papers use the Gowalla dataset that can be found in the SNAP repository[11], such as Wang et al. (2013), Guo et al. (2015a), Chen et al. (2016b). As it also happens with other LBSNs, some researchers claim they use Gowalla, but they fail to provide any source to obtain such dataset (Ying et al., 2012, Noulas et al., 2012, Zhou and Wang, 2014, Li et al., 2017a).

- Brightkite: a less popular LBSN, but used in a large number of research works because of its availability. In the same way as Gowalla, a dataset from this LBSN is included in the SNAP repository[12], which makes it easy to be used by researchers, since it is not available since 2012; some examples include Hosseini and Li (2016), Chen et al. (2016b), Yao et al. (2015).

- Yelp: this is a LBSN that focuses on businesses, rather than generic POIs like other LBSNs. It also differs from the other LBSNs in that users provide a rating based on 5 stars to the different businesses they visit; besides, users can also write a review about them. The Yelp dataset is available on its website[13] and can be obtained after agreeing on the dataset license; however, many papers refer to a different url[14] that does not work anymore, like Li et al. (2015b), Gupta et al. (2015), Yang et al. (2017a), Baral et al. (2018); this is because this dataset was first released in the context of a challenge ran by Yelp, which has gone at least through 12 rounds where the data has been increased each time; this makes the comparisons even more difficult since it is not possible to get the dataset corresponding to a specific round, and this information is usually omitted in the papers.

- Others: in addition to the aforementioned LBSNs, some proposals work with datasets extracted from other systems, such as Jiepang (a Chinese LBSN similar to Foursquare) used in Lian et al. (2014, 2015, 2016), Ravi and Subramaniyaswamy (2017a), Weeplaces used in Baral and Li (2016), Baral et al. (2016), GeoLife used in Abdel-Fatao et al. (2015), Zhu et al. (2017), and others less popular in our context, like Twitter and TripAdvisor.

While doing our systematic review, we found several versions of datasets coming from the same LBSN. For the sake of space and clarity, we show in Table 3.7 the LBSNs used by the research work with more citations (according to Scopus) for each year, together with some statistics of the dataset and other evaluation details reported in the experiments, such as the type of split and the evaluation metrics. Based on

[11]Gowalla, http://snap.stanford.edu/data/loc-gowalla.html
[12]Brightkite, http://snap.stanford.edu/data/loc-brightkite.html
[13]Yelp dataset, https://www.yelp.com/dataset
[14]Yelp dataset, https://www.yelp.com/dataset_challenge

**Table 3.6:** Papers included in our review that use a dataset from each LBSN.

| Number of Papers | LBSN | | | | |
|---|---|---|---|---|---|
| | Gowalla | Foursquare | Yelp | Brightkite | Other |
| Most Representatives | 26 | 31 | 4 | 3 | 7 |
| Total | 118 | 153 | 37 | 32 | 36 |

**Table 3.7:** Details of the experimental settings for the work with most citations each year (note each work appears once for each reported dataset). NA denotes that value is not provided in the paper. The columns Ref. and Cit. denote the original reference for that work and the number of citations, as of June 2020.

| Details | | | | Statistics | | | | Evaluation Details | |
|---|---|---|---|---|---|---|---|---|---|
| Year | Ref. | Acronym | Cit. | Dataset | Users | POIs | Check-ins | Metrics used | Type of Split |
| 2011 | Ye et al. (2011) | USG | 697 | Foursquare | 153,577 | 96,229 | NA | P, R | Random per user |
| | | | | Whrrl | 5,892 | 53,432 | NA | | |
| 2012 | Bao et al. (2012) | NA | 431 | Foursquare (NY) | 2,886 | NA | 10,687 | P, R | Other |
| | | | | Foursquare (LA) | 228 | NA | 9,836 | | |
| 2013 | Yuan et al. (2013) | UTE, SE, UTE+SE | 463 | Foursquare | 2,321 | 5,596 | 194,108 | P, R | Random per user |
| | | | | Gowalla | 10,162 | 24,250 | 456,988 | | |
| 2014 | Lian et al. (2014) | GeoMF | 281 | Jiepang | 276,450 | 574,095 | NA | P, R | Random per user |
| 2015 | Li et al. (2015a) | RankGeoFM | 180 | Foursquare | 2,321 | 5,596 | 194,108 | P, R | Temporal per user |
| | | | | Gowalla | 10,162 | 24,250 | 456,988 | | |
| 2016 | Li et al. (2016a) | ASMF | 132 | Gowalla | 52,216 | 98,351 | 2,577,336 | P, R, MAP | Temporal per user |
| | | | | Foursquare | 2,551 | 13,474 | 124,933 | | |
| 2017 | Yang et al. (2017a) | PACE | 98 | Gowalla | 18,737 | 32,510 | 1,278,274 | P, R, nDCG, MAP | Temporal per user |
| | | | | Yelp | 30,887 | 18,995 | 860,888 | | |
| 2018 | Wang et al. (2018b) | GeoIE | 23 | Foursquare | 6,118 | 88,193 | 172,961 | P, R | Temporal per user |
| | | | | Gowalla | 1,624 | 3,585 | 115,890 | | |
| 2019 | Qian et al. (2019) | STA | 14 | Foursquare | 114,508 | 62,462 | 1,434,668 | R, nDCG | Temporal per user |
| | | | | Gowalla | 107,092 | 1,280,969 | 6,442,892 | | |

**Table 3.8:** Statistics of reported versions for the Foursquare dataset in works included in our review, sorted by number of check-ins.

| Users | POIs | Check-ins | References using this dataset |
|---|---|---|---|
| 18,107 | 36,907 | 2,073,740 | Huang et al. (2015), Stepan et al. (2016) |
| 114,508 | 62,462 | 1,434,668 | Ren et al. (2017), Qian et al. (2018, 2019), Christoforidis et al. (2018) |
| 11,326 | 182,968 | 1,385,223 | Zhang and Chow (2013), Zhang et al. (2014, 2015a), Zhang and Chow (2015a) Stepan et al. (2016), Zhang and Chow (2016), Gao et al. (2018b), Gao et al. (2018) |
| 10,766 | 10,695 | 1,336,278 | Manotumruksa et al. (2017a, 2018, 2019a,b) |
| 12,422 | 46,194 | 738,445 | Liu et al. (2013a, 2015) |
| 10,034 | 16,561 | 865,647 | Zhao et al. (2016a, 2017a) |
| 4,163 | 121,142 | 483,813 | Zhang and Chow (2015b), Hosseini and Li (2016), Xie et al. (2016b), Hosseini et al. (2017, 2019) |
| 49,823 | 18,899 | 419,509 | Xia et al. (2017a, 2018) |
| 5,269 | 26,381 | 288,079 | Gao et al. (2013), Stepan et al. (2016) |
| 2,321 | 5,596 | 194,108 | Yuan et al. (2013, 2014), Kojima and Takagi (2015), Li et al. (2015a), Zhao et al. (2017b), Si et al. (2017) Yu et al. (2017), Xu et al. (2018a), Chen et al. (2018a), Kala and Nandhini (2019), Gao et al. (2019), Zeng et al. (2019) |
| 2,579 | 97,013 | 157,404 | He et al. (2017a), Li et al. (2019a) |
| 2,823 | 84,937 | 130,583 | He et al. (2016), Li et al. (2019a) |
| 2,551 | 13,474 | 124,933 | Li et al. (2016a, 2017a), Su et al. (2019a) |
| 8,308 | 49,521 | 86,375 | Ozsoy et al. (2014, 2016) |
| 9,800 | 4,626 | 45,711 | Zhu et al. (2018a, 2019) |

this information, we observe that all of them evaluate based on some notion of ranking quality; while it is true that this evidences researchers are taking into account the guidelines provided in the recommendation field as a whole (McNee et al., 2006), a possible reason for this is that the data to be predicted is not ratings anymore, but binary feedback: whether the user visited the POI or not. In any case, this table emphasizes an even more important problem: most researchers are only focused on accuracy, disregarding additional dimensions such as novelty, diversity, or serendipity that are becoming prevalent in recent years in the evaluation of RS (Castells et al., 2015).

Table 3.7 also shows an interesting paradigm shift: those works prior to 2015 used a random split, and those more recent used a temporal split. We consider this a very important signal, since it indicates that (at least for the works that are later more cited by colleagues) a more realistic type of split is being used, which would indeed make the proposed approaches easier to put in context in a real scenario. It is also positive that most of these works (it is by no means the same in general) contrast their approaches against two data sources, which makes the results easier to generalize. On the other hand, what can be considered as a worrying sign is that there are not two articles sharing the number of check-ins or users, except Yuan et al. (2013) and Li et al. (2015a), but even in this case, each work performs a different data splitting; moreover, there are even cases where some of the statistics are not included (like the number of items or check-ins). This makes it almost impossible to compare two research works without implementing everything from scratch, hence hindering reproducibility and the advancement of the field (Said and Bellogín, 2014).

We were also surprised that in most cases the source code of the proposed algorithm is not provided. In particular, among the papers with more citations, Yang et al. (2017a) redirects to a repository with source code. With respect to the rest of the analyzed papers (that is, out of the 244 works, only Hu and Ester (2014), Li et al. (2015b), Zhao et al. (2017a), Ma et al. (2018), Manotumruksa et al. (2018), Li et al. (2018b), Christoforidis et al. (2018), Yu et al. (2019) provide a url to download the source code of their algorithm.

As a final analysis, we present in Table 3.8 different versions of datasets extracted from Foursquare, considering this is the most widely used LBSN in the articles included in our review. In this selection we show the datasets used by more than one article, since there are works using other variations not reported here but, for the sake of space, we focused only on those reported more than once among the papers considered in our analysis[15]. Nevertheless, it is remarkable to observe the large difference in the number of check-ins, ranging from 45k to 2M interactions; as a consequence, the experiments presented in the different works are probably not comparable at all – even if they belong to the same LBSN – since the inherent properties of the system are not

[15]For instance, a widely used version called Global-scale dataset presented in Yang et al. (2016) is not included in this table because only one paper reported the exact statistics as the original paper (which is actually not considered because it does not perform POI recommendation), whereas other works take subsets of it.

preserved: for instance, in some cases we have more users than items and in others the other way around, the levels of sparsity change dramatically, together with the number of cities/regions included in each dataset. Additionally, it is interesting to observe that most datasets are seldom used, and the few cases where the same dataset is used by many works, it is because they belong to the same authors, as in Zhu et al. (2018a, 2019), or Zhao et al. (2016a, 2017a).

## 3.9 Discussion

In this chapter we have dealt with RG1: *Review the state-of-the-art on POI Recommender Systems to identify and characterize the most important works in the area.* We have presented a summary of the POIs recommendation works between 2011 and 2019 classifying them according to the type of information and the type of algorithm used and the evaluation methodology (with special emphasis to this last aspect). Besides, we have also identified some open issues after performing an analysis on the state-of-the-art on POI recommendation. After that we have listed some potential future research lines we believe are in line with parallel developments in the Recommender System field.

Although several research efforts have been devoted to the problem of POI recommendation, it is still possible to find unresolved issues in the field, which opens up opportunities to improve the area as a whole, for instance, because they are more aligned with the necessities of the final users and, probably, with industry practitioners. By analyzing the current proposals in POI recommendation, we have observed some important open issues that need to be addressed. In the following, we group them according to the three main systematic reviews we performed: models or algorithms, evaluation methodologies, and datasets.

### 3.9.1 Open issues

Regarding algorithms, matrix factorization and, more recently, deep neural networks are very popular approaches in POI recommendation; however, it is often difficult to explain why the recommendations from these methods are made since they behave like a *black box* and this can be problematic in some domains, in particular in tourism. In addition, we have also observed that most researchers do not test their approaches against other classic recommendation algorithms like simple CF methods or non-personalized item popularity, comparing only with other POI recommendation approaches. Similarly, the sequential information, despite its relevance in this domain, is not usually exploited, which sometimes prompts incorrect or not realistic evaluation methodologies.

In fact, about evaluation methodologies we consider the comparisons between different algorithms must always be as transparent and as fair as possible in order to determine which proposals are superior to others. Therefore, although in the papers analyzed in this survey there seems to be consensus in evaluating the approaches using IR metrics like Precision or Recall, this is not the case about how to perform the splits, as there are both random and temporal partitions (each of them with different variations), even though the latter ones are the only strategies that could simulate real scenarios. At the same time, the sparsity of the datasets used, whether or not they have been pre-filtered, etc., also affects the performance of the models, which in particular may prevent from having research works that are comparable between each other.

Regarding the datasets, although most proposals extract data from well-known LBSNs such as Foursquare, Gowalla, or Yelp, these datasets are often not comparable among them due to different decisions considered when filtering users or items, or even how the data were captured, which produces a large number of versions for each LBSN. Comparing datasets is even more difficult when some researchers do not provide complete statistics about the actual dataset used in the experiments, leading to data with completely different characteristics and inherent properties (sparsity, granularity of temporal, geographical, and social information, POI attributes, and so on) even when they belong to the same LBSN.

### 3.9.2 Recommendations

As we have seen along the survey, the problem of POI recommendation is attractive to a growing number of researchers in the area of recommender systems. However, it may seem as if most of these issues have something in common: they make both the reproducibility and the generality of the proposed algorithms very difficult. Thus, in order to advance towards better systems and foster high-quality research, we recommend to:

- Explain in detail how the algorithms have been evaluated indicating the metrics used, the type of split and the rest of the models that have been used as baselines. In this regard, we suggest to test the proposed algorithm against specific POI recommendation models while also analyzing its performance against other baselines used in classical recommendation, such as neighbor-based algorithms, matrix factorization approaches, and the most-popular method. The evaluation methodology must be the same for all the algorithms and if it is necessary to make different experiments for choosing the parameters, this needs to be done for all the algorithms involved in the experiments and, if possible, with a validation subset independent of the test set.

- Clearly indicate the statistics of the used datasets, stating if any pre-processing step has been performed and showing the final details of the used data, including the number of users, POIs, and check-ins. This would help to detect the percentage of data that was removed to critically analyze if the filtered dataset is actually representative of the original dataset. We also strongly recommend researchers to use more than one dataset or, at least, to use different types of splits or more than one split from the same data if enough information is available.

- Finally, the easiest way to replicate a research work is by providing the code with a detailed description to achieve the same results mentioned in the paper; if this is not possible, the next best option is to, at least, provide the final datasets with which the algorithms were evaluated, so anyone interested in replicating it should not worry about that step of the evaluation pipeline.

In general, these recommendations aim to fix a lack of reproducible experimental settings that could hinder whether there is a significant improvement in the field, as already discussed in the RS and IR communities (Said and Bellogín, 2014, Armstrong et al., 2009).

# Part II

# Proposed solutions on temporal, sequential, and geographical contexts

# 4

# Novel approaches for evaluating Recommender Systems

As previously discussed in this thesis, Recommender Systems evaluation is a topic that still receives significant attention in the area. We have shown in Section 2.4 that measuring the recommenders quality by only considering their accuracy in terms of relevance is incomplete, and even though other researchers have proposed the study of alternative aspects such as novelty and diversity, it is still possible to analyze other important dimensions. Hence, in this chapter, we propose different models to evaluate the algorithms by integrating additional contexts. More specifically, in Section 4.1 we show how to incorporate temporal information into a novelty and diversity framework proposed in Vargas and Castells (2011), to consider as novel those items that have more recent interactions in the system. Then, in Section 4.2, we show complementary metrics to the classical accuracy ones to analyze those items recommended by the algorithms that have a very low rating. Later, in Section 4.3 we show how to make use of the attributes of the items and the users in the evaluation of the recommenders. On the one hand, item attributes allow us to define a relevance model considering as partially relevant those recommended items that share similarities with the ground truth of the user even if they do not appear in the test set. On the other hand, user attributes allow us to determine if the recommenders are biased towards specific groups of users or if otherwise the recommendations are of similar quality between the users. In Section 4.4 we show how to incorporate sequentiality in classical accuracy metrics to take into account not only relevance but also the order in which the recommendations are produced. In Section 4.5, we show an experimental analysis of the proposed metrics using a wide range of classical recommenders in well-known datasets, and finally, in Section 4.6 we discuss about other potential uses and the importance of these metrics.

The content of this chapter has been partially published in the following articles:

- **Pablo Sánchez** and Alejandro Bellogín. Time-aware novelty metrics for recommender systems. In Gabriella Pasi, Benjamin Piwowarski, Leif Azzopardi, and Allan Hanbury, editors, *Advances in Information Retrieval - 40th European Conference on IR Research*, ECIR 2018, Grenoble, France, March 26-29, 2018, Proceedings, volume 10772 of Lecture Notes in Computer Science, pages 357-370. Springer, 2018.

- **Pablo Sánchez** and Alejandro Bellogín. Measuring anti-relevance: a study on when recommendation algorithms produce bad suggestions. In Sole Pera, Michael D. Ekstrand, Xavier Amatriain, and John O'Donovan, editors, *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys 2018, Vancouver, BC, Canada, October. 2-7, 2018, pages 367-371. ACM, 2018.

- **Pablo Sánchez** and Alejandro Bellogín. Attribute-based evaluation for recommender systems: incorporating user and item attributes in evaluation metrics. In Toine Bogers, Alan Said, Peter Brusilovsky, and Domonkos Tikk, editors, *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019., pages 378-382. ACM, 2019.

## 4.1 Time-aware Novelty metrics

One of the most important contexts to be used in Recommender Systems is the temporal information. This type of information can be very diverse (for example, the time when the item was consumed by the user, the time when the item was included in the system, or the date of publication/premiere when the items are books or movies, etc.) and has generally proven useful in several domains of Recommender Systems, like the Point-of-Interest (POI) problem (see Chapter 3). However, temporal data has generally been modeled for generating recommendations, not for evaluating them. In order to explore its application in the evaluation step, we propose to include the temporal influence in the novelty and diversity framework defined in Vargas and Castells (2011) as follows:

$$m(R_u \mid \theta) = C \sum_{i_n \in R_u} \text{disc}(n) p(rel \mid i_n, u) \text{nov}(i_n \mid \theta) \tag{4.1}$$

where $\theta$ is a variable that represents the context on which the item discovery is applied (e.g., time intervals, group of users, etc.). The rest of the elements are the same as the ones explained for the novelty metric named Expected Popularity Complement (EPC), shown in Equation 2.37. In fact, the EPC metric is derived from this framework if we model the novelty component as follows: $\text{nov}(i_n \mid \theta) = 1 - p(seen|i, \theta)$, that is, that the more popular an item is, the less novel it is. Thus, our plan is to extend that framework to incorporate time and to observe when the recommended items are novel according to a temporal model.

Hence, in our time-aware novelty model, we define an item profile $\langle t_1(i), \cdots, t_n(i) \rangle$ for each item $i$, according to a specific time model $t$. This representation is flexible enough to allow us to operate with diverse data. For example, we may use different types of the metadata available in the system about the item (e.g., it could be its release time, its creation date, or inclusion in the system catalog, which may differ from the creation time, as in music or movies databases). In fact, our item profile could include all these different times in the same representation, since we will use an aggregation function to summarize it into a single value. An example of this item profiling is used in Chou et al. (2015), where the authors exploit the release dates of music songs.

Nonetheless, when dealing with collaborative datasets, another source of information becomes available. It is possible to define the temporal profile of an item as the instants when a user interacted with it in the system. Even though the most typical interaction type in the literature are ratings, this definition is easily extended to any other interaction between users and items, such as comments, reviews, clicks, purchases, or check-ins (in the case of Point-of-Interest recommendation).

Obviously, these time profiles will model the item in a different way. Whereas the metadata-based item representations are based on some objective, static information (such as the release date), an interaction-based representation produces dynamic profiles, which will change depending on when the profiling is performed. At the same time, these latter representations are probably more subjective in the sense that they depend on how users interact with the system, but, because of this, they allow for profiles more tailored to how the community is actually interacting with a specific item in the system.

We can draw a parallelism with how documents are treated when building temporal query profiles. According to Jones and Diaz (2007), documents in IR are annotated with a timestamp corresponding to the date the document was published; this would correspond to the first group of time models presented, those based on metadata. However, we could also annotate the documents according to when they are accessed – or retrieved – as a response to a query; this document profiling strategy would correspond to the second group of time models, those based on interactions.

As explained before, the core idea in the framework shown in Equation 4.1 is how to define the item novelty model $\text{nov}(i \mid \theta)$, where $\theta$ stands for a generic contextual variable. Thus, in order to model a time-aware novelty metric, we propose to encode the time model of the items in the $\theta$ variable as follows:

$$\theta_t = \{\theta_t(i)\} = \{(i, \langle t_1(i), \cdots, t_n(i)\rangle)\} \quad . \tag{4.2}$$

Here, $t_j(i)$ represents the $j$-th component of the temporal representation or profile of item $i$ by a specific time model $t$. Hence, we propose to compute the item novelty model based on different statistics of each item temporal profile.

In this context, the most basic model would account the item novelty based on the first appearance of that item in the system (**FIN**, from *First Item Novelty*), that is, $\text{nov}^F(i \mid \theta_t) \propto \min_{j \in \theta_t(i)} \theta_t(i)$ ; in this sense, an item is novel when looking at the system timeline (by default, starting with the first logged interaction, although this initial timestamp could be configured to have a later value) and onwards. On the other hand, we may also model the item novelty with respect to the point of view of the evaluation split, where the item would be considered more novel the closer it is with respect to the end of the training split (**LIN**, *Last Item Novelty*). Note that the LIN model does not necessarily simplify to the complement of FIN, since its actual value will depend on the time model being used. Additionally, we can also model the item novelty by computing the average (**AIN**) or median (**MIN**) of the item profile.

These item novelty models cannot be integrated like that in the framework shown in Equation 4.1, since it is a probabilistic model, and hence, these quantities need to be, at least, normalized to produce valid values of the novelty metric. At the moment, the range of the item novelty models is the same as the range of the item time models, which, in general, return timestamps. A simple normalization scheme would divide the output of the item novelty model by the maximum possible value of a specific time model, another possibility would be applying a min-max normalization. Therefore, we can formalize the item novelty model as follows:

$$\mathrm{nov}^{f,n}(i \mid \theta_t) = n(f(\theta_t(i)), \theta_t) \ , \tag{4.3}$$

where $f$ is one of the previous functions LIN, FIN, AIN, or MIN, and $n$ is a normalization function, either the simple normalization function defined as $n(x, s) = x / \max s$, or the min-max normalization formulated as $n(x, s) = (x - \min s)/(\max s - \min s)$. To help the reader better understand how the values of LIN, FIN, AIN, and MIN would work, we show in Figure 4.1 an example where we see the interactions of the users within the items in a system and also the different values of the novelty models.

In summary, we propose a family of time-aware item novelty models that depend on a specific item time model, an aggregation function that summarizes the item temporal profile into a single number, and a normalization function that allows a fair comparison among novelty values for different items. Then, these item novelty models would lead to time-aware novelty metrics when integrated within the framework presented in Equation 4.1.

## 4.2 Anti-Relevance metrics

In Section 2.4 we showed different metrics to measure the relevance of the recommendations and other complementary dimensions like novelty or diversity. In the previous section, we extended a novelty framework to provide a more complete picture of the temporal novelty of the items that are recommended to users. Even if all these metrics allow us to analyze different characteristics of the recommendations, an analysis of the recommended items that have been specifically marked by the user as “bad” (those items in the user profile explicitly rated as such – i.e., with low values in a bounded scale or “not liked” products) is still missing. Such an analysis would provide a complementary view of the accuracy of recommendations: while it is clear that a recommender suggesting more relevant items is preferred to another showing less relevant items, when the number of relevant items is comparable, techniques producing too many bad recommendations should be avoided, as the user confidence in the system may be undermined (Herlocker et al., 2004).

With this idea in mind, we aim to analyze the failures the algorithms make when recommending by accounting for these effects in the evaluation metrics. As far as we know, this is an issue not addressed in the field, at least from an analytical perspective and in terms of defining new metrics, as we present hereafter, although some researchers have started to consider this problem (Mena-Maldonado et al., 2020).

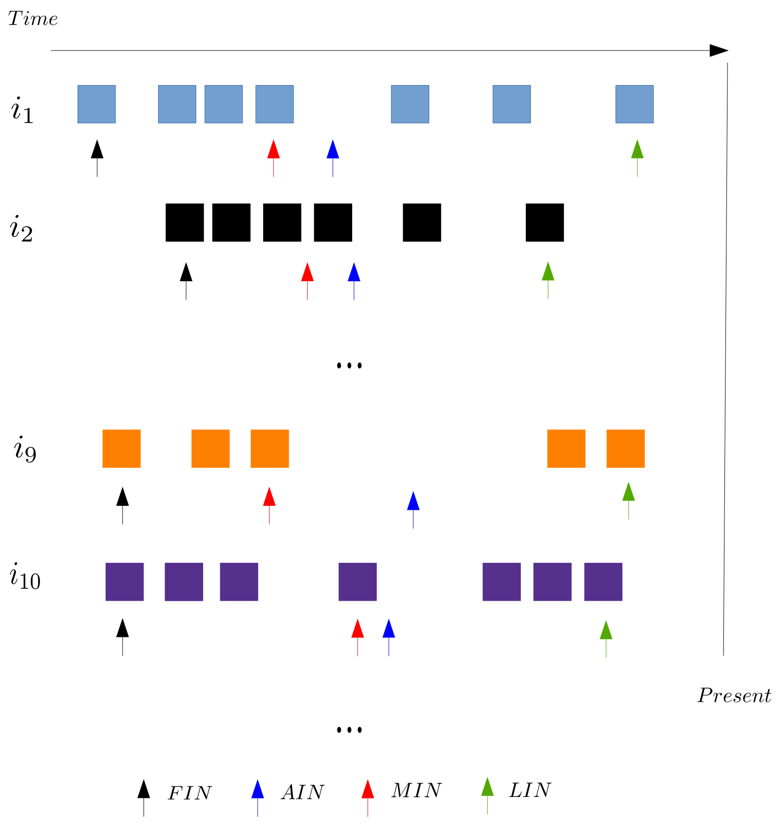


**Figure 4.1:** Example of different values of LIN, FIN, AIN, and MIN depending on the rating evolution of the items throughout the life of the system. Each color represents an item, whereas each square is the moment in which that item was consumed by a user.

One of the few examples we have identified in the area can be found in Ekstrand and Riedl (2012), where the authors measure how close different algorithms produce predictions with respect to the real ratings, and propose to use those findings to create a hybrid recommendation algorithm. In Gui et al. (2016), the authors proposed a method to detect bad recommendations using a residual model to capture users' utility. In IR, where there is a long tradition on evaluation, however, we do find authors that introduce alternative definitions for evaluation metrics, either to focus on the most difficult queries (Voorhees, 2004) or directly on the not relevant documents (frustration metric) (Myaeng and Korfhage, 1990), and even others that question relevance as a criteria to evaluate the performance of systems (Belkin et al., 2008).

Thus, drawing from IR and recent RS papers, we adapt the Probabilistic Ranking Principle (PRP) for recommendation (Cañamares and Castells, 2018) and interpret it as the starting point for the definition of an objective function that should be optimized by our algorithm (Clarke et al., 2008). The PRP states that *if a system's response to* *a query is a ranking of documents in order of decreasing probability of relevance, the overall effectiveness of the system to its users will be maximized* (Robertson, 1997).

Hence, to apply the PRP (or to estimate the PRP to evaluate a retrieval or recommender system) we must estimate the probability that a document or item $i$ is relevant for a user (need or profile) $u$, i.e., $p(\text{rel} = 1|u, i)$. This quantity is usually translated into RS evaluation as $p(r_{ui} \geq \tau_R|u, i)$, where $\tau_R$ is a relevance threshold, meaning that any item in the test set of user $u$ that was rated above (or equal) to such threshold will be considered relevant.

In our proposal, we study the *dual PRP problem*: estimating the probability of anti-relevance and ranking the documents according to the opposite of this probability, that is: $1-p(\text{rel} = 0|u, i)$. As before, this could be translated into RS as $1-p(r_{ui} \leq \tau_{AR}|u, i)$ for some anti-relevance threshold $\tau_{AR}$. It should be noted that these estimates can also be computed even when no ratings are available, as long as some measure of negative and positive interaction – e.g., products explicitly *liked* and *not liked* by the user – can be defined.

We argue that most evaluation metrics $m$ are formulated as estimating the classical PRP (note the similarities between this formulation and the one shown in Equation 4.1 if we incorporate the relevance model inside the metric):

$$m(R_u|\theta_{rel}) = C \sum_{i \in R_u} m(\theta_{rel}(r_{ui})|u, i) \tag{4.4}$$

since $\theta_{rel}$ encodes the dependency on relevance from the PRP $p(r_{ui} \geq \tau_R|u, i)$, where $C$ is a normalization constant and $R_u$ is the recommendation list computed for user $u$; that is, most evaluation metrics only account for the relevant items in the test set of a user. Now, to formulate the anti-metrics we follow the dual PRP problem as stated before:

$$\begin{aligned}\overline{m}(R_u|\theta_{arel}) = C \sum_{i \in R_u} (1 - \overline{m}(\theta_{arel}(r_{ui})|u, i)) \propto \\ \propto 1 - C' \sum_{i \in R_u} m(\theta_{arel}(r_{ui})|u, i) = 1 - m(R_u|\theta_{arel})\end{aligned} \tag{4.5}$$

Thus, our anti-metrics formulation is equivalent to computing any relevance-based metric using an anti-relevance model (where an item is relevant if $r_{ui} \leq \tau_{AR}$) and returning its complement. In Figure 4.2 we show an example of the intuition behind the anti-relevance metrics. In this figure we compare the test set of the user ($T_u$) against two recommendation lists ($R_u^1$ and $R_u^2$). In both lists, a relevant item is recommended (marked in green) but in the test set we can observe that the user specifically stated $I_7$ as anti-relevant. Therefore, $R_u^2$ should be preferred over $R_u^1$, as $R_u^1$ is returning that anti-relevant item.

Note that Equations 4.4 and 4.5 are computed for every user to whom we have made recommendations, so when reporting the value of the metrics in a set of users, we normally compute the arithmetic mean of all the results obtained.

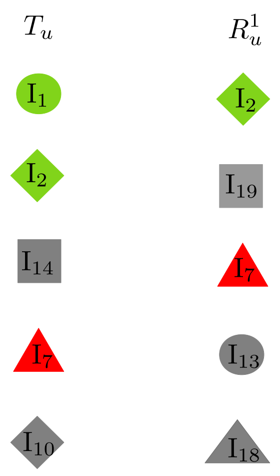


**Figure 4.2:** Comparison between the test set of a user ($T_u$) and two recommendation lists ($R_u^1$ and $R_u^2$) that return the same number of relevant items (1, marked in green), but in one case the list return an anti-relevant item (marked in red) whereas the second list only returns non-relevant items (marked in grey). As a consequence, the second list should be preferred as it does not provide any anti-relevant items.

We should note that unknown items (those whose ratings are not in the test set) are still considered as non-relevant by the classical metrics and by the anti-relevance model – i.e., they contribute with a 0 in the metric computation – however, since the anti-metric reports the complement value, the amount of unknown items is actually affecting the final result, and hence, it should be somehow considered. However, it is not easy to integrate this value in the final metric, since depending on the metric $m$ the final result could be normalized or transformed in some way (e.g., nDCG); nonetheless, for simple, binary metrics such as precision or recall, this issue could be addressed by subtracting the number of unknown items in $R_u$ (or $\text{unk}(R_u)$) as follows: $1 - (m(R_u|\theta_{arel}) + \text{unk}(R_u))$.

Finally, by optimizing the ranking obtained by the dual problem, relevant items are ignored (just as anti-relevant items were ignored when optimizing for the original PRP). Hence, in order to balance the information measured in each case, we should combine the metrics based on relevance and anti-relevance. Thus, if we have a measure $x$ computed by some metric $m$, and another measure $\overline{x}$ computed by its anti-metric $\overline{m}$, we can linearly combine those values with the average $\mu(x) = 1/2(x + \overline{x})$, the harmonic mean $H(x) = 2\frac{x\overline{x}}{x+\overline{x}}$, or taking the likelihood ratio $LH(x) = \frac{x}{1-\overline{x}}$ inspired by the probabilistic interpretation of $m$ and how this statistic is typically used to take decisions when comparing classifications based on two classes (in our case, $\text{rel} = 1$ and $\text{rel} = 0$).

## 4.3 Attribute-based evaluation

Despite all the advances done in Recommender Systems evaluation, and the novel formulations shown in the previous sections, there are still some issues that need to be addressed. Firstly, when analyzing the evaluation results we tend to treat all users equally, ignoring the specific underlying aspects of each user profile. This is something not entirely new, as it is easy to observe that in many datasets there are users that are more difficult to satisfy than others (e.g., they do not have a mature history of ratings, or they are simply different to the rest of the community, the so-called *gray sheep* effect). Furthermore, as described in the previous section, when evaluating the relevance of the recommendations, we normally only take into account those items the user has somehow interacted with, either by liking, rating, or clicking on them, ignoring the rest of recommended items, and, thus, considering them as non-relevant, which may have a strong impact on the evaluation hypotheses and obtained results (Bellogín et al., 2017). In some domains, like Point-of-Interest recommendation, this assumption may impose constraints too difficult to satisfy by the algorithms due to the high sparsity of the data. As a solution to this issue, some researchers decided to measure matchings between the item attributes (categories) in the test set and the recommended ones, instead of the actual items (He et al., 2017a, Brilhante et al., 2013, Palumbo et al., 2017, Lim et al., 2015).

With these ideas in mind, we aim to delve into some aspects of the evaluation of RS that are generally neglected in traditional evaluation. On the one hand, we define a way to generalize classic ranking-based accuracy metrics in order to consider as "partially relevant" those items that, although not specifically stated as relevant, are highly similar to the ones the user liked. On the other hand, we formalize the notion of aggregating the evaluation metric values at the user level by assigning users into different groups according to the available attributes (such as age, gender, or their consumption history), which would help us to detect if a recommendation algorithm makes more correct recommendations to users belonging to some specific groups.

As explained in the previous section, it is generally acknowledged that a common formulation for many evaluation metrics is the following (arithmetic) mean:

$$m(\theta) = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} m(R_u, \theta) \tag{4.6}$$

where $m(\theta)$ represents the value of a metric $m$ on the output of some recommendation algorithm, and $m(R_u, \theta)$ is the user-level metric value considering the list $R_u$ provided by such algorithm. For example, in agreement with the formulation presented in Section 2.4.3 for Precision or $P@k$, this function would be defined as $m(R_u, \theta) = \mathrm{Rel_u@k/k}$ , whereas for nDCG, it would take the form of $m(R_u, \theta) = \sum_{n=1}^{k}(2^{rel_n} - 1)/(\log_2(n+1))/\mathrm{IDCG}(u)@k$, where IDCG(u)@k represents the ideal ranking for each user.

In this context, we may now analyze whether all users in a recommender system have the same behavior, especially from the system perspective (Jannach and Adomavicius, 2016). In fact, some users may have more influence than others – either as power users (Herlocker et al., 2004) or influential in a social network context (Trusov et al., 2010) –, some spend more time in the system, or some could even be easier to satisfy than others (Said and Bellogín, 2018). In any case, it is reasonable to assume that, under various settings, the system designer might want to aggregate these user-level metric values according to different schemes (Deldjoo et al., 2021). For this, we propose to use a function $c$ that assigns a weight for each user, either based on her behavior in the system or according to her attributes, which is incorporated into Equation 4.6 as follows:

$$m(\theta) = C^{-1} \sum_{u \in \mathcal{U}} c(u) m(R_u, \theta) \tag{4.7}$$

where $C = \sum_u c(u)$. Even though this formulation is reduced to a simple weighted sum of the user-level metric values (note that Equation 4.6 is recovered by setting $c(u) = 1$), we argue that it consolidates many ad-hoc evaluations performed in the literature: for instance, cold-start evaluation and recommendation fairness can be recovered by setting binary weights on function $c$ such that only the cold-start users or those belonging to a specific group are aggregated; additionally, typical filters applied to the data, such as ignoring users with low ratings in training or test, could be modeled under this formulation by setting the appropriate function $c$. These examples will be considered later as use cases in the experiments.

Now that we have shown how to incorporate user attributes in any evaluation metric, we shall show how we can do the same with the item attributes by exploiting some concept of item similarity in the evaluation metrics. This idea is not completely new, since most of the works on diversity evaluation use some concept of similarity metric (Castells et al., 2015), the main difference is that in those cases such metric is computed within the recommendation list, to analyze how similar the recommended items are with respect to each other. Moreover, our proposal is inspired by the works of He et al. (2017a) and Brilhante et al. (2013), where the item categories instead of the actual items are compared as ground truth. It should be noted that in a recent published work, the authors performed a user study where they analyzed the recommended items that were not included in the ground truth, and found that those items highly similar to those selected by the user were perceived as *acceptable recommendations* (Frumerman et al., 2019), hence, validating our proposal.

We define the following three sets of items based on those items included in the recommendation list $R_u$, the items in the ground truth of user $u$ ($T_u$), and a given item similarity metric $\text{sim}_F(i, j)$ over the feature space F: $I^+(u)$ are those items explicitly interacted by the user and included in the test set, i.e., $I^+(u) = \{i \in R_u : i \in T_u\}$; $I^*(u)$ is formed by those items that show a non-zero similarity, $I^*(u) = \{i \in R_u \wedge i \notin I^+(u) : \exists j \in T_u^*, \text{sim}_F(i, j) > 0\}$, where $T_u^* = T_u \cap \overline{I^+(u)}$, that is, the subset of the user test that was not recommended; and, finally, the set $I^-(u)$ is formed by those items not included in the previous two sets. Now, taking the formulation previously introduced in Equation 4.6, we integrate and exploit these sets of items when computing the item-level metric value as follows:

$$m(R_u, \theta) \propto \sum_{i \in I^+(u)} w^+(u,i) + \sum_{i \in I^*(u)} w^*(u,i) + \sum_{i \in I^-(u)} w^-(u,i) \tag{4.8}$$

where each $w^+$, $w^*$, and $w^-$ are properly adjusted weights; typically, $w^+(u,i) = 1$ and $w^-(u,i) = 0$, we propose to define $w^*$ as a function $\tau$ that depends on the similarity with respect to the closest item not interacted by the user, that is: $w^*(u,i) = \tau(\text{sim}^*_F(i,j;\alpha))$, where $\text{sim}^*_F(i,j;\alpha) = \max_{j \in Te^*(u)} \alpha \cdot \text{sim}_F(i,j)$, where we use a penalization weight $\alpha$. Note that this item set could use as many similarity metrics as desired, associating each similarity with a different penalization; for instance, as we shall show in the experiments, we may consider two items as similar when they share the same director or the same genre, but obviously the penalization should be higher in the latter case. It is now straightforward to obtain those metrics defined in the literature where authors matched items at the category level (He et al., 2017a, Brilhante et al., 2013): by simply setting $w^+(u,i) = w^*(u,i) = 1, w^-(u,i) = 0$ and taking function $\text{sim}_F(i,j)$ as the binary mapping that outputs 1 if two items share the same category and 0 otherwise. We should take care, however, when we create item set $I^*(u)$ that each item $j$ found to have the highest similarity with respect to a recommended item $i$, is only considered once; this can be achieved by iterating through the recommendation list in order and removing the items from $T^*(u)$ once they are used. In Figure 4.3 we show a toy example explaining how the metrics incorporating the item categories would work. In this case, the test set of the user contains a circle, a diamond and a triangle. However, the first recommendation list only return a diamond (the relevant item), but no other item matching any attribute of the test items. On the other hand, the second recommendation list contains a relevant item (circle) but also two non-relevant items ($I_{20}$ and $I_{16}$) that matches the attributes of the items in the test set (hence, $R^2_u$ should be preferred over $R^1_u$).

We want to emphasize that our proposal does not extend the actual ground truth being exploited, as the number of relevant items is still limited by the items found in the test set of the user.

## 4.4 Sequential metrics

As we have shown in the previous sections, Recommender Systems can be evaluated in terms of a large number of dimensions (novelty, diversity, accuracy, freshness, etc.). However, regardless of the context being evaluated, the ordering of items is normally ignored. For example, traditional IR metrics (typically used in the RS area) like Precision, Recall, MAP, or nDCG do not consider the order in which the user consumed the items in the test set to measure the accuracy of the recommendations, they only focus on the relevance of the items. While this may be sufficient for most recommendation tasks, we believe that this type of sequential metrics might be very useful in domains where sequences have a great importance, such as music recommendation and POI and

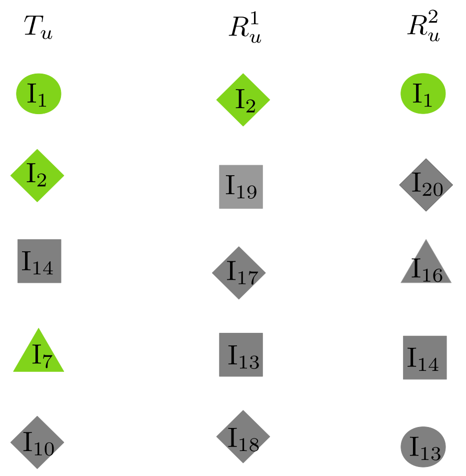


**Figure 4.3:** Comparison between the test set of a user ($T_u$) and two lists that return the same number of relevant items (1, marked in green), and non-relevant items (marked in grey). In this case, the shape of the item denotes its attribute/category. As we observe, there are more non-relevant items in the second list that matches the categories of the relevant items in the test set. As a consequence, the second list should be preferred since it provides more items matching the attributes of the relevant items of the test set.

route recommendation, since generally in these areas the proposed models are still evaluated with classical techniques such as MAE/RMSE or ranking-based accuracy metrics (explained in Section 2.4) (Liu et al., 2017, Schedl et al., 2018). In fact, as part of the thesis is focused on LBSNs data, we will pay special attention to the application of these metrics on trajectory recommendations as we will show in Chapter 6.

More specifically, except for the work presented in Chen et al. (2016a), where the authors propose a metric based on $F_1$ that takes into account the pairwise order between items, and the evaluation done in Menon et al. (2017) that used the same metric based on $F_1$ on pairs of items, we have not found other approaches where the evaluation metrics explicitly compare the order of the recommendations against the consumed items. Although sequential frameworks have been proposed in several domains, many of them do not include sequentiality in the evaluation step. For example, Monti et al. (2018) proposed a framework for sequence-based Recommender Systems, but the authors did not include any sequence-aware metric comparing against the order observed in the test set. We find a similar situation in a recent challenge on playlist continuation, where no sequence-aware metrics were used even though sequentiality was key for the solutions proposed (Zamani et al., 2019) and simple metrics based on the proportion of common elements in two top-n lists have been applied in the past to 2-song and 3-song sequences (Maillet et al., 2009); on the other hand, in the context of trajectory mining we do find measures that compare two trajectories with respect to all points in those trajectories while considering the order (Jeung et al., 2011).

One of those techniques that allows us to perform *alignments* between two sequences is the Longest Common Subsequence (LCS). The LCS algorithm is a technique used to

**Algorithm 1** Longest Common Subsequence

```
1: procedure LCS($x, y$)                                   ▷ The LCS of $x$ and $y$
2:     $L[0 \cdots m, 0 \cdots n] \leftarrow 0$
3:     for $i \leftarrow 1, m$ do
4:         for $j \leftarrow 1, n$ do
5:             if $x_i = y_j$ then
6:                 $L[i,j] \leftarrow L[i-1, j-1] + 1$         ▷ There is a matching
7:             else
8:                 $L[i,j] \leftarrow \max(L[i, j-1], L[i-1, j])$
9:             end if
10:        end for
11:    end for
12:    return $L[m, n]$    ▷ $L[i,j]$ contains the length of an LCS between $x_1 \ldots x_i$ and
   $i_1 \ldots y_j$
13: end procedure
```

find a subsequence of elements (not necessarily consecutive) whose length is the maximum possible between two sequences. Formally, the Longest Common Subsequence problem is defined like this: given a string $x$ over an alphabet $\Sigma = (\sigma_1, \cdots, \sigma_s)$, a *subsequence* of $x$ is any string $w$ that can be obtained from $x$ deleting zero or more (not necessarily consecutive) symbols. The LCS problem for input strings $x = x_1 \cdots x_m$ and $y = y_1 \cdots y_n$ (assuming $m \leq n$) consists of finding a third string $w = w_1 \cdots w_l$ such that $w$ is a subsequence of $x$ and also a subsequence of $y$, and $w$ is of maximum possible length. In general, such $w$ is not unique. This problem arises in a number of applications, from text editing to molecular sequence comparisons, and has been extensively studied (Apostolico, 1997). The standard dynamic programming solution to compute the LCS can be seen in Algorithm 1, which consists in filling a matrix $C$ of dimensions $m \times n$ following the Equation 4.9:

$$C[i,j] = \begin{cases} 0 & \text{if i = 0 or j = 0} \\ C[i-1, j-1] + 1 & \text{if } i,j > 0 \text{ and } x_i = y_j \\ \max(C[i, j-1], C[i-1, j]) & \text{if } i,j > 0 \text{ and } x_i \neq y_j \end{cases} \tag{4.9}$$

where $x_i$ and $y_j$ represent the characters at indexes $i$ and $j$ (starting in 1) of strings $X$ and $Y$. Hence, the final value in $C[l_x, l_y]$ will be the length of the LCS between the two input strings. This algorithm has a complexity of $O(mn)$ for both time and space, where $n$ and $m$ are the length of the two input sequences. In Table 4.1 the reader can see an example of a computation of the LCS for the strings "BACBAD" and "ABAZDC", whose LCS is "ABAD" with a length of 4.

**Table 4.1:** LCS example matrix. The circled number represents the length of the LCS found by the algorithm.

| | 0 | B | A | C | B | A | D |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| B | 0 | 1 | 1 | 1 | 2 | 2 | 2 |
| A | 0 | 1 | 2 | 2 | 2 | 3 | 3 |
| Z | 0 | 1 | 2 | 2 | 2 | 3 | 3 |
| D | 0 | 1 | 2 | 2 | 2 | 3 | 4 |
| C | 0 | 1 | 2 | 3 | 3 | 3 | $\textcircled{4}$ |

Our proposal, hence, is to incorporate the LCS technique into well-known ranking evaluation metrics such as Precision or nDCG, already described in Section 2.4.3. Although the original LCS problem is designed to work with strings, we can consider the sequence of items of a user as if it were a string in which each item consumed represents a character in the sequence. However, other possible transformations may also be applied, e.g. operating with the item categories (these transformations will be presented in more detail in Section 5.1).

Hence, we aim to measure how many items were recommended in the same order as the user visited them, while considering, at the same time, the inherent evaluation dimensions defined by each evaluation metric. As a consequence, if the relevance of the items is binary, then the sequence-aware evaluation metrics (based on LCS) will always achieve a lower or equal value than their non-sequential counterparts, since the LCS component would serve as a penalization factor every time a sequence is not followed in order.

More specifically, when adapting the LCS technique for evaluation, one of the sequences to be compared will be the recommendation list ($R_u$) and the other the actual consumed items that appear in the test set of the user ($T_u$, ordered by ascending timestamp). We propose to perform a slight modification on how the LCS is computed so that any classical ranking-based accuracy metric could be adapted in such a way that sequentiality is integrated seamlessly in their computation.

Based on this definition, a straightforward modification would be to create a new variable that will compute the value in a user basis for any evaluation metric. Such a variable would need to be updated whenever there is a match (second line in Equation 4.9), since that means that the subsequence found is now larger than before and, hence, the sequentiality will be considered. However, this process would not always work, since during the LCS computation there are matches that are not used because other subsequences are actually longer[1].

In any case, this observation helps us on finding where and how we should modify

[1]Consider, for example, a symbol that appears at the beginning of the first sequence and at the end of the second one (such as A in ABCDE and BDFA); even if this is a valid matching, the difference between the position of the symbol in both subsequences makes it to not be part of the longest common subsequence.

**Algorithm 2** Longest Common Subsequence: Backtracking

```
1: procedure LCS_BACKTRACKING($R_u, T_u, C$)    ▷ The recommendation list $R_u$ is
   limited to the desired cutoff; $T_u$ denotes the ordered test set of user $u$.
2:     $s \leftarrow 0$
3:     $(i, j) \leftarrow dim(C)$
4:     while $i > 1$ AND $j > 1$ do
5:         $x_i \leftarrow R_u[i]$
6:         $y_j \leftarrow T_u[j]$
7:         if $x_i = y_j$ then
8:             $s \leftarrow s + m(R_u, T_u, x_i, i)$    ▷ There is a matching
9:             $i \leftarrow i - 1$
10:            $j \leftarrow j - 1$
11:        else if $C[i-1][j] > C[i][j-1]$ then
12:            $i \leftarrow i - 1$
13:        else
14:            $j \leftarrow j - 1$
15:        end if
16:    end while
17:    return $s$
18: end procedure
```

the LCS algorithm to produce valid measurements. The correct place to integrate the evaluation metric component into the LCS algorithm is whenever the longest subsequence is being restored: at the end of the LCS algorithm, in the backtracking step (shown in Algorithm 2) that uses the computed matrix $C$ to find which elements belong to the common subsequence.

When performing the backtracking step as in Algorithm 2, we assume any evaluation metric can be divided in a user-item basis in such a way that $m(R_u, T_u, x_i, i)$ provides the contribution that item $x_i$ presented in the recommendation list $R_u$ at ranking position $i$ makes to user $u$ as evidenced in her test set. In the following, we provide these user-item components for some of the most well-known evaluation metrics: $m_P = 1/|R_u|$ for Precision, $m_R = 1/|T_u|$ for Recall, $m_{nDCG} = (2^{\text{rel}(x_i,u)} - 1)/(\log(i+1))$ for nDCG, $m_{ARHR} = 1/i$ for ARHR (Gunawardana and Shani, 2015). Note that, in the case of the metrics that need to be normalized by an ideal metric value (like nDCG), it would be enough to call the same procedure but with the test set of the user instead of the recommendation list. From now on, to denote the sequential variation of a specific metric $M$ (provided a user-item component of such metric $m_M$ is available) we shall use $M_s$.

In order to illustrate the differences between classical ranking-based accuracy metrics and the proposed adaptation (based on LCS) for sequence-aware metrics, we show in Figure 4.4 an example of how these metrics behave when using a user-item component based on Precision for two recommendation lists $R_u^1$ and $R_u^2$. Here we observe

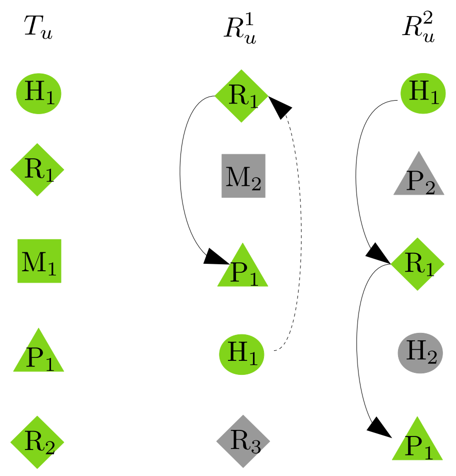


**Figure 4.4:** Comparison between the test set of a user ($T_u$) and two lists that return the same number of relevant items (3, marked in green), but in one case the list does not present all the items in the correct order ($R_u^1$), hence $LCS(T_u, R_u^1) = 2$, whereas in the other case, the order is the same as in the test set, and the value of LCS is maximum: $LCS(T_u, R_u^2) = 3$. As a consequence, $\mathrm{P_s}$ $(R_u^2) = \mathrm{P}(R_u^2) = 3/5$, whereas $\mathrm{P_s}$ $(R_u^1) = 2/5$ and $\mathrm{P}(R_u^1) = 3/5$.

that $\mathrm{P_s}$ is equivalent to P only when the relevant items are returned in the same order – that is, for list $R_u^2$ – and produces a lower value in other case, as explained before. Note that by using the LCS algorithm we admit sequences of symbols that are not necessarily consecutive. This differs from the problem of finding the Longest Common Substring, whose algorithm can also be used to compare sequences (Gusfield, 1997); however we believe such technique is less suitable for the task we address here because of its lack of flexibility, since a recommended sequence of items, despite having gaps with respect to the test set, should also be considered interesting for the user.

## 4.5 Experiments on novel approaches for evaluating Recommender Systems

In order to analyze the behavior of the metrics proposed in this chapter of the thesis, we have decided to experiment with them in isolated experiments that will be presented now. To do so, we decided to work with two representative datasets in the area. The first one is the Movielens1M dataset, i.e. a movie-based dataset provided by the GroupLens research lab. In the GroupLens website[2] there are different versions of these datasets, including the 100K dataset (formed by 100,000 interactions from 943 users on 1682 movies), the 1M dataset (formed by 1,000,209 interactions from 6,040 users on 3,900 movies and is the one used in these experiments), the 10M dataset (formed by 10,000,054 interactions from 71,567 users on 10,681 movies) and the 20M dataset

[2]Grouplens, `https://grouplens.org/`

(formed by 20,000,263 interactions from 138,493 users on 27,278 movies). Some of these datasets also contain additional information such as movie genres and user information such as age or gender. The second one will be the city of Tokyo from a large Foursquare dataset. This Foursquare dataset is a POI dataset formed by 33,278,683 check-ins by 266,909 users on 3,680,126 venues in 415 different cities around the world provided by Yang et al. (2016). The dataset is available in the authors website[3] and it also contains additional information like the gender, friends, and followers for the users who have checked-in in the cities of New York and Tokyo.

We have selected these datasets because they are representative from two recommendation domains, either a traditional one (Movielens1M) or another one based on Point-of-Interest (Foursquare). Besides, they also provide additional information of the users, like their gender (in both datasets) or their age in the Movielens1M dataset and also the timestamps of the interactions. We need all this information to test the aforementioned metrics, which need temporal information (for time-aware novelty models), user and item attributes (for the user and item attribute metrics), and ratings (for the anti-relevance framework). As in this chapter we will analyze the data from Tokyo of the Foursquare dataset, we will refer to this dataset as FS (Tokyo).

### 4.5.1 Evaluation settings

As in this chapter we want to analyze more in depth how Recommender Systems are evaluated, we will show the results of classic algorithms of the state-of-the-art (defined in Sections 2.2 and 2.3) using both standard evaluation metrics and our proposed metrics shown before, except for the sequential metrics, that will be used in the experiments of Chapter 6 (where we propose algorithms for generating trajectories). Hence, the reported recommenders are:

- Random (Rnd): the recommended items are selected randomly for each user.
- Popularity (Pop): this model recommends the items that have been rated by more users in the system.
- Collaborative-via content (UBCB): a hybrid user-based neighborhood recommender that uses a content-based similarity, based on the one proposed in Balabanovic and Shoham (1997), explained in Section 2.2.3.
- Item-based (IB): an item-based neighborhood recommender described in Section 2.2.2.
- User-based (UB): a user-based neighborhood recommender described in Section 2.2.2.
- Matrix factorization (HKV): a matrix factorization algorithm that uses Alternating Least Squares (ALS) for optimization. Shown in Section 2.2.2 and proposed in Hu et al. (2008).

[3]Foursquare dataset, `https://sites.google.com/site/yangdingqi/home/foursquare-dataset`

- Bayesian Personalized Ranking (BPRMF): matrix factorization algorithm that uses Bayesian Personalized Ranking as optimization method, as shown in Section 2.2.2 and proposed in Rendle et al. (2009).
- Temporal decay user-based (TD): a user-based neighborhood recommender that incorporates a time decay function so that items rated more recently have a higher score. Based on the algorithm shown in Equation 2.20, in Section 2.3, without mean-centering and normalization.
- Factorized Markov Chains (MC): first-order Markov Chains algorithms that factorized the item $\times$ item transition matrix. Implementation provided by He and McAuley (2016).
- Factorized Personalized Markov Chains (FPMC): combination of matrix factorization and First order Markov Chains explained in Section 2.3, proposed in Rendle et al. (2010). Implementation provided by He and McAuley (2016).
- Factorized Sequential Prediction with Item Similarity Models (Fossil): combination of Factorized Item Similarity Models and Markov Chains explained in Section 2.3 and proposed in He and McAuley (2016).
- Convolutional Sequence Embedding Recommendation Model (Caser): sequential deep learning model that learns the sequential patterns using convolutional filters. Explained in Section 2.3 and proposed in Tang and Wang (2018).
- Skyline (Skyline): recommender that reads the test set and recommends to the users the items they consider as relevant. This recommender serves to show the maximum value that can be obtained by the recommender in the ranking-based accuracy metrics.

For the experiments shown in this section and the rest of the thesis, unless stated otherwise, we have used the following frameworks: for most of the algorithms, we have used the models implemented in the RankSys library[4], as it is the library implemented in the Information Retrieval Group (IRG) from the Universidad Autónoma de Madrid (UAM). Nevertheless, for the MC, FPMC and Fossil recommenders, we have used the code provided by the authors[5], adapting it to generate top-n recommendations. For the BPRMF recommender we have used the MyMedialite library[6]. Finally, for the Caser recommender, we have used the code available in the following url[7]. The parameters tested for the recommenders can be shown in Table 4.2. Note that for the Caser recommender we tried more parameters to build the sequences, but the algorithm fails if every user do not have at least $L$ interactions. Finally, it should be mentioned

[4]RankSys, https://github.com/RankSys/RankSys

[5]Source code from He and McAuley (2016), https://drive.google.com/file/d/0B9Ck8jw-TZUEeEhSWXU2WWloc0k/view

[6]MyMedialite, http://www.mymedialite.net/

[7]Caser implementation, https://github.com/graytowne/caser_pytorch

**Table 4.2:** Parameters used with the evaluated recommenders. VecC and SetJ stand for VectorCosine (Equation 2.9) and SetJaccard (Equation 2.11).

| **Recommender** | **Parameters** |
|---|---|
| Rnd | None |
| Pop | None |
| UBCB | $k$={40, 60, 80, 100, 120} |
| IB | $k$={40, 60, 80, 100, 120}, sim={VecC, SetJ } |
| UB | $k$={40, 60, 80, 100, 120}, sim={VecC, SetJ } |
| HKV | Factors={10, 50, 100}, $\alpha$={0.1, 1, 10}, $\lambda$={0.1, 1, 10} |
| BPRMF | Factors={10, 50, 100}, Iter=50, LearnRate=0.05, RegJ=RegU/10, BiasReg={0, 0.5, 1}, RegU=RegI={0.0025, 0.001, 0.005, 0.01, 0.1} |
| TD | $k$={40, 60, 80, 100, 120}, sim={VecC, SetJ }, $\lambda$={0.05, 0.1} |
| MC | $K$={5, 10, 20, 50}, $\lambda$={0.1, 0.2} |
| FPMC | $K$={5, 10, 20, 50}, $\lambda$={0.1, 0.2} |
| Fossil | $K$={5, 10, 20, 50}, $\lambda$={0.1, 0.2}, $L$={1, 2, 3} |
| Caser | $L$=2, $T$={1, 2}, Iter=30, bath=512, LearnRate=0.003, negSamples=3, d={10, 50}, v=4, h={4, 16}, drop=0.5 |
| Skyline | None |

that throughout the different chapters and sections of this thesis, we will explain (if necessary) additional algorithms and their corresponding configurations used in the experiments.

In order to analyze different evaluation aspects, we create two splits at system level (community-centered base set with a proportion size condition) in each of the datasets. For the first one, we make a temporal split where 80% of the oldest ratings are sent to the training set and the rest to test. For the second split, we perform a random partition where 80% of the ratings go to training and the rest to test. In addition, in the case of FS (Tokyo), we have performed a 2-core, that is, we have filtered the dataset forcing all users and items to have at least 2 interactions. We show in Table 4.4 the statistics of the two datasets and the two splits that we are using in our experiments. At the same time, it should be noted that there may be repetitions in this dataset, i.e. a user may have visited the same POI more than once (and in the same way, it might be some interactions in the test set that have already appeared in training set). However, as the classic recommendation problem is intended to discover new items to users, we follow the TrainItems methodology (see Section 2.4.3), that is, for each user we consider as candidate item all those items that the user has not consumed before (that appear in the training set). On the other hand, since the behavior of the recommenders is not defined to deal with repetitions, in the case of FS (Tokyo), we remove the repetitions of the training set, making each user visit each POI only once, maintaining the timestamp of the last interaction. Finally, for the Movielens1M dataset, we consider as relevant all test items that have a rating of 4 or 5 (5 is the maximum value in a 1-5 rating scale), while in FS (Tokyo) we consider as relevant any POI that the user has visited in test. The final configuration of the recommenders are selected by the maximum

**Table 4.3:** Optimal parameters for Movielens1M and FS (Tokyo) datasets used in the experiments of this chapter. For each dataset, we report the best parameters of the recommenders for each split (Temporal or Random) according to nDCG@5.

| | **Movielens1M** | | **FS (Tokyo)** | |
|---|---|---|---|---|
| **Rec** | **Temporal** | **Random** | **Temporal** | **Random** |
| UBCB: k | 120 | 100 | 120 | 120 |
| IB: k, sim | 80, VecC | 40, VecC | 120, VecC | 120, VecC |
| UB: k, sim | 80, VecC | 100, VecC | 100, SetJ | 60, SetJ |
| HKV: k, $\alpha$, $\lambda$ | 10, 1, 10 | 50, 1, 10 | 10, 10, 0.1 | 50, 10, 10 |
| BPRMF: k, $\lambda_u = \lambda_i$, $\lambda_0$ | 10, 0.01, 0 | 100, 0.01, 0 | 100, 0.01, 0 | 100, 0.005, 0 |
| TD: k, sim, $\lambda$ | 120, VecC, 0.1 | - | 120, SetJ, 0.05 | - |
| MC: k, $\lambda$ | 50, 0.2 | - | 50, 0.1 | - |
| FPMC: k, $\lambda$ | 5, 0.2 | - | 10, 0.2 | - |
| Fossil: k, $\lambda$, L | 20, 0.1, 1 | - | 20, 0.1 | - |
| Caser: T, d, nh | 1, 10, 4 | - | 2, 10, 16 | - |

**Table 4.4:** Statistics of the Movielens1M and FS (Tokyo) datasets used in the experiments of this chapter. For each dataset, we report the statistics of the full dataset (denoted as Complete) and for each of the training and test splits. The density is computed as the number of interactions divided by the product of the number of users and items.

| **Dataset** | **Split** | **Training/Test** | **$\lvert\mathcal{U}\rvert$** | **$\lvert\mathcal{I}\rvert$** | **Interactions** | **Density** | **Period** |
|---|---|---|---|---|---|---|---|
| Movielens1M | Complete | - | 6k | 3.7k | 1M | 4.47% | Apr 2000 - Feb 2003 |
| | Temporal | Training | 5.4k | 3.7k | 800k | 4.05% | Apr 2000 - Dec 2000 |
| | Temporal | Test | 1.8k | 3.5k | 200k | 3.2% | Dec 2000 - Feb 2003 |
| | Random | Training | 6k | 3.7k | 800k | 3.6% | Apr 2000 - Feb 2003 |
| | Random | Test | 6k | 3.5k | 200k | 1% | Apr 2000 - Feb 2003 |
| FS (Tokyo) | Complete | - | 11.6k | 51.1k | 998k | 0.17% | Apr 2012 - Sep 2013 |
| | Temporal | Training | 10.9k | 49.7k | 798k | 0.15% | Apr 2012 - Apr 2013 |
| | Temporal | Test | 8.3k | 29.2k | 200k | 0.08% | Apr 2013 - Sep 2013 |
| | Random | Training | 11.6k | 50.8k | 798k | 0.14% | Apr 2012 - Sep 2013 |
| | Random | Test | 10.5k | 33.7k | 200k | 0.06% | Apr 2012 - Sep 2013 |

**Table 4.5:** Performance results on Movielens1M dataset. Temporal system split (80% ancient ratings to train, rest to test). In bold, it is represented the best performance without considering the Skyline recommender. We show in bold with a † the best performance considering the Skyline recommender. All metrics @5.

| Recommender | P | R | nDCG | EPC | Gini | IC | UC | UC-Rel |
|---|---|---|---|---|---|---|---|---|
| Rnd | 0.019 | 0.001 | 0.012 | 0.962 | **†0.651** | **†0.920** | **†1,783** | **†1,762** |
| $\text{Rnd}_{\text{CF}}$ | 0.015 | 0.001 | 0.008 | **†0.964** | 0.565 | 0.784 | 1,143 | 1,122 |
| Pop | **0.281** | 0.030 | **0.221** | 0.566 | 0.004 | 0.025 | **†1,783** | **†1,762** |
| $\text{Pop}_{\text{CF}}$ | 0.210 | 0.026 | 0.161 | 0.587 | 0.005 | 0.025 | 1,143 | 1,122 |
| UBCB | 0.254 | 0.039 | 0.195 | 0.669 | 0.018 | 0.070 | 1,143 | 1,122 |
| IB | 0.234 | 0.034 | 0.177 | 0.670 | 0.014 | 0.057 | 1,143 | 1,122 |
| UB | 0.248 | 0.039 | 0.195 | 0.684 | 0.024 | 0.091 | 1,143 | 1,122 |
| HKV | 0.257 | **0.043** | 0.202 | 0.725 | 0.038 | 0.122 | 1,143 | 1,122 |
| BPRMF | 0.231 | 0.034 | 0.172 | 0.700 | 0.036 | 0.146 | 1,143 | 1,122 |
| TD | 0.248 | 0.040 | 0.194 | 0.675 | 0.021 | 0.079 | 1,143 | 1,122 |
| MC | 0.177 | 0.029 | 0.134 | 0.708 | 0.010 | 0.039 | 1,143 | 1,122 |
| FPMC | 0.212 | 0.029 | 0.159 | 0.616 | 0.004 | 0.021 | 1,143 | 1,122 |
| Fossil | 0.227 | 0.036 | 0.170 | 0.650 | 0.009 | 0.048 | 1,143 | 1,122 |
| Caser | 0.192 | 0.029 | 0.136 | 0.788 | 0.089 | 0.238 | 1,143 | 1,122 |
| Skyline | **†0.943** | 0.280 | **†1.000** | 0.871 | 0.131 | 0.368 | 1,762 | **†1,762** |
| $\text{Skyline}_{\text{CF}}$ | 0.911 | **†0.363** | 0.999 | 0.883 | 0.133 | 0.322 | 1,122 | 1,122 |

value obtained according to nDCG@5. In Table 4.3 we show final parameters for both datasets under the two splits with this metric. Finally, we only report the results of the temporal/sequential recommenders (MC, FPMC, Fossil, TD, and Caser) under the temporal split as they are formulated to specifically predict future events.

### 4.5.2 Analyzing the effect of different splits on classical metrics

In Tables 4.5 and 4.6 we show the performance of the recommenders presented in Section 4.5.1 in terms of accuracy metrics (P, R, nDCG), novelty (EPC), diversity (Gini, IC), and user coverage (UC and UC-Rel) in the Movielens1M dataset under the temporal and random splits respectively. First, let us start by explaining the difference between UC and UC-Rel: UC represents the number of users to whom the recommenders are capable of recommending at least one item, while UC-Rel represents the number of users that have at least one relevant item in the test set to whom we are capable of making a recommendation. Hence, when computing the average of all the users in the metrics, we divide the cumulative result by UC-Rel, not UC. Thus, by comparing both values we can see that not all test users have a relevant item (it should be noted that we only consider as relevant those items that have been rated with a 4 or 5; items rated with a lower score are therefore not taken into account). When analyzing Table 4.5, we observe how the best recommender in terms of accuracy (ignoring the Skyline) is usually the Pop recommender. This is an interesting result and it is due to several reasons. First, the popularity bias, in which popular items tend to be more recommended than the rest of the items (Abdollahpouri et al., 2019a).

**Table 4.6:** Performance results on Movielens1M dataset. Random system split (80% ratings to train, rest to test, randomly selected). Same configuration as in Table 4.5.

| Recommender | P | R | nDCG | EPC | Gini | IC | UC | UC-Rel |
|---|---|---|---|---|---|---|---|---|
| Rnd | 0.006 | 0.002 | 0.004 | †**0.967** | 0.786 | †**1.000** | †**6,036** | †**5,978** |
| Rnd$_{\text{CF}}$ | 0.006 | 0.002 | 0.004 | 0.966 | †**0.789** | 1.000 | †**6,036** | †**5,978** |
| Pop | 0.168 | 0.052 | 0.141 | 0.626 | 0.003 | 0.012 | †**6,036** | †**5,978** |
| Pop$_{\text{CF}}$ | 0.168 | 0.052 | 0.141 | 0.626 | 0.003 | 0.012 | †**6,036** | †**5,978** |
| UBCB | 0.281 | 0.099 | 0.249 | 0.703 | 0.015 | 0.080 | †**6,036** | †**5,978** |
| IB | 0.287 | 0.105 | 0.241 | 0.712 | 0.015 | 0.108 | †**6,036** | †**5,978** |
| UB | 0.330 | 0.133 | 0.300 | 0.714 | 0.019 | 0.106 | †**6,036** | †**5,978** |
| HKV | **0.350** | **0.142** | **0.316** | 0.787 | 0.061 | 0.219 | †**6,036** | †**5,978** |
| BPRMF | 0.295 | 0.107 | 0.248 | 0.769 | 0.068 | 0.368 | †**6,036** | †**5,978** |
| Skyline | †**0.914** | †**0.512** | †**1.000** | 0.859 | 0.200 | 0.606 | 5,978 | †**5,978** |
| Skyline$_{\text{CF}}$ | †**0.914** | †**0.512** | †**1.000** | 0.859 | 0.200 | 0.606 | 5,978 | †**5,978** |

Besides, in this dataset, all users have consumed at least 20 items, so it is rare to find users who have seen unpopular items. In addition, we emphasize that it is a temporal split and as we can observe, there are a high number of users to whom most of the recommenders cannot provide any kind of recommendation (users who do not have any interaction in the training set). For this reason, we also show different versions for the recommenders with a total coverage (Pop, Rnd and Skyline). These new versions have the subscript "CF", from Collaborative Filtering, indicating that they only make recommendations to those users who have previously appeared in the training set. In that case, the performance of the Pop$_{\text{CF}}$ decreases substantially, showing us that the new users that this recommender cannot perform any recommendation tend to consume the most popular items in the test set, hence getting a high advantage from this method. Regarding the Skyline, it is important to mention that it does not obtain a value of 1 in the relevance metrics because not all users have a large number of relevant items (in particular, less than the cutoff used, which limits the upper bound of metrics like precision), besides, there might be new items in test that the recommender cannot return due to the TrainItems strategy.

For the rest of the recommenders, we can see that there are small differences between the IB and the UB, although the latter shows slightly better performance in the relevance metrics, perhaps because it learns user profiles better (we know that each user has rated at least 20 items but we do not know the equivalent value for the items). TD performance has almost no differences with the classic UB. This may be because the timestamps in Movielens1M are not as representative as in other datasets (Harper and Konstan, 2016). The matrix factorization approaches (HKV and BPRMF) obtain different results with respect to the neighbor-based approaches, obtaining a similar performance in the relevance metrics. It should be noted, however, that these types of algorithms have many more parameters than those based on neighbors, so it may be that the parameter tuning that has been done is not sufficient to find the best combinations. Finally, the three sequential recommenders (MC, FPMC, and Fossil) deserve

**Table 4.7:** Performance results on FS (Tokyo) dataset. Temporal global split (80% ancient ratings to train, rest to test). Same configuration as in Table 4.5.

| **Recommender** | **P** | **R** | **nDCG** | **EPC** | **Gini** | **IC** | **UC** | **UC-Rel** |
|---|---|---|---|---|---|---|---|---|
| Rnd | 0.000 | 0.000 | 0.000 | 0.999 | †**0.419** | †**0.564** | †**8,268** | †**8,268** |
| $\text{Rnd}_{\text{CF}}$ | 0.000 | 0.000 | 0.000 | †**0.999** | 0.404 | 0.536 | 7,615 | 7,615 |
| Pop | **0.065** | **0.035** | **0.078** | 0.767 | 0.000 | 0.000 | †**8,268** | †**8,268** |
| $\text{Pop}_{\text{CF}}$ | 0.059 | 0.030 | 0.070 | 0.771 | 0.000 | 0.000 | 7,615 | 7,615 |
| UBCB | 0.060 | 0.031 | 0.072 | 0.802 | 0.000 | 0.005 | 7,615 | 7,615 |
| IB | 0.054 | 0.026 | 0.066 | 0.854 | 0.026 | 0.132 | 7,611 | 7,611 |
| UB | 0.064 | 0.033 | 0.076 | 0.795 | 0.000 | 0.009 | 7,611 | 7,611 |
| HKV | 0.059 | 0.031 | 0.070 | 0.857 | 0.001 | 0.005 | 7,615 | 7,615 |
| BPRMF | 0.060 | 0.031 | 0.071 | 0.772 | 0.000 | 0.001 | 7,615 | 7,615 |
| TD | 0.064 | 0.033 | 0.077 | 0.796 | 0.000 | 0.011 | 7,611 | 7,611 |
| MC | 0.048 | 0.024 | 0.057 | 0.812 | 0.001 | 0.020 | 7,471 | 7,471 |
| FPMC | 0.038 | 0.021 | 0.047 | 0.853 | 0.000 | 0.003 | 7,457 | 7,457 |
| Fossil | 0.047 | 0.025 | 0.056 | 0.894 | 0.002 | 0.022 | 7,457 | 7,457 |
| Caser | 0.045 | 0.026 | 0.054 | 0.861 | 0.011 | 0.070 | 7,615 | 7,615 |
| Skyline | 0.775 | †**0.520** | †**0.933** | 0.984 | 0.164 | 0.301 | 7,900 | 7,900 |
| $\text{Skyline}_{\text{CF}}$ | †**0.777** | 0.500 | 0.928 | 0.985 | 0.163 | 0.289 | 7,250 | 7,250 |

special attention. It is important to note that despite being the most complex algorithms do not get the best results, this may be attributed to several reasons. First, in the original paper where these recommenders were proposed (He and McAuley, 2016) the authors optimized the recommenders using the Area Under Curve (AUC) metric. Besides, as they are sequential recommendation algorithms, they are optimized to predict the next item the user will consume. In this sense, the algorithms order all the interactions for each user and use the last item consumed in training by that user as validation. However, we have performed a temporal global split, so there might be users who have a large number of interactions in test and not so many in training, making it difficult to optimize the parameters under these circumstances. A similar effect can be the cause of the discrete performance of Caser that it is also oriented for exploiting sequences under a temporal per user split, even if the author of the repository claims that it is prepared to work with the Movielens1M dataset.

Finally, with respect to novelty and/or diversity performance, it should be noted that in general, in this type of metrics the Rnd recommender is always the best. This is because this algorithm does not introduce any artificial bias in the recommendations, so the recommendations are totally independent from the user's profile (except, for the fact that we do not recommend the users items they have consumed in the training set). Notwithstanding this result, it serves to confirm that in general the best recommendation is the one that achieves a balance between accuracy and non-accuracy metrics. In fact, if we analyze the Skyline (the ideal recommender), we can see that this recommender is superior in all metrics to the rest of the recommenders, if we ignore the Rnd algorithm, proving that the relevant items are also novel and diverse.

Let us now study Table 4.6 (random split). First, as already mentioned, we do not

**Table 4.8:** Performance results on FS (Tokyo) dataset. Random system split (80% ratings to train, rest to test, randomly selected). Same configuration as in Table 4.5.

| Recommender | P | R | nDCG | EPC | Gini | IC | UC | UC-Rel |
|---|---|---|---|---|---|---|---|---|
| Rnd | 0.000 | 0.000 | 0.000 | †**0.999** | 0.477 | 0.643 | †**10,463** | †**10,463** |
| $\text{Rnd}_{\text{CF}}$ | 0.000 | 0.000 | 0.000 | †**0.999** | †**0.479** | †**0.645** | 10,432 | 10,432 |
| Pop | 0.041 | 0.032 | 0.052 | 0.772 | 0.000 | 0.000 | †**10,463** | †**10,463** |
| $\text{Pop}_{\text{CF}}$ | 0.041 | 0.032 | 0.052 | 0.772 | 0.000 | 0.000 | 10,432 | 10,432 |
| UBCB | 0.044 | 0.035 | 0.057 | 0.806 | 0.000 | 0.007 | 10,432 | 10,432 |
| IB | 0.044 | 0.032 | 0.056 | 0.857 | 0.033 | 0.167 | 10,430 | 10,430 |
| UB | 0.050 | **0.040** | **0.064** | 0.807 | 0.001 | 0.015 | 10,430 | 10,430 |
| HKV | **0.051** | **0.040** | **0.064** | 0.894 | 0.002 | 0.012 | 10,432 | 10,432 |
| BPRMF | 0.042 | 0.032 | 0.053 | 0.775 | 0.000 | 0.002 | 10,432 | 10,432 |
| Skyline | 0.736 | †**0.540** | †**0.923** | 0.981 | 0.195 | 0.353 | 10,086 | 10,086 |
| $\text{Skyline}_{\text{CF}}$ | †**0.738** | 0.538 | †**0.923** | 0.981 | 0.195 | 0.352 | 10,055 | 10,055 |

include the temporal and sequential models as in the random split there may be interactions in the test that have occurred before others in training. The most significant change with respect to the previous table can be seen in the Pop recommender, which now becomes the second worst model (after Rnd), both in terms of relevance and in terms of novelty and diversity. We can also observe how the number of users in the test is much higher, showing that the activity of the users in a system change between the splits (there may be users who interact continuously and others who interact very sporadically or who abandon the system). Regarding the other recommenders, the performance of the matrix factorization algorithms deserves special attention, since besides exceeding in accuracy the rest of the models, they are also capable of obtaining a larger novelty and diversity. This may be due to the way of learning the latent factors. While neighbor-based models are very dependent on finding common users (therefore the recommended items depend considerably on the neighbors found), the factors learned in MFs do not have this problem. These results obtained using a widely known dataset in the area shows that the type of split is critical when analyzing the performance of the recommenders.

Let us now analyze what happens when we use the Foursquare dataset in the city of Tokyo. In Tables 4.7 and 4.8 we show the performance of the algorithms again using a temporal and a random split (respectively). If we look at the results of the recommenders in terms of accuracy, we can see how in this dataset the values obtained are much lower than in Movielens1M, this is due to the sparsity of the LBSNs: while in Movielens1M every user has rated at least 20 items, in this dataset the only restriction has been a 2-core. However, we can find equivalencies with respect to the results in the previous dataset. When we perform a temporal split, the Pop recommender is the best in terms of accuracy, although the differences between the algorithms are much smaller than in Movielens. Nevertheless, it is striking how sequential recommenders perform worse than classic recommenders in terms of accuracy, although they have a very good performance on novelty. This may be due to the fact that in this dataset

**Table 4.9:** Performance results on Movielens1M dataset. Temporal system split (80% ancient ratings to train, rest to test). Same configuration as in Table 4.5. Comparison between the best recommenders reported in previous tables and best recommenders obtained using a validation set for parameter tuning.

| Recommender | P | R | nDCG | EPC | Gini | IC | UC | UC-Rel |
|---|---|---|---|---|---|---|---|---|
| Rnd | 0.019 | 0.001 | 0.012 | 0.962 | †**0.651** | †**0.920** | †**1,783** | †**1,762** |
| $\text{Rnd}_{\text{CF}}$ | 0.015 | 0.001 | 0.008 | †**0.964** | 0.565 | 0.784 | 1,143 | 1,122 |
| Pop | **0.281** | 0.030 | **0.221** | 0.566 | 0.004 | 0.025 | †**1,783** | †**1,762** |
| $\text{Pop}_{\text{CF}}$ | 0.210 | 0.026 | 0.161 | 0.587 | 0.005 | 0.025 | 1,143 | 1,122 |
| UBCB | 0.254 | 0.039 | 0.195 | 0.669 | 0.018 | 0.070 | 1,143 | 1,122 |
| $\text{UBCB}_{\text{val}}$ | 0.254 | 0.039 | 0.195 | 0.669 | 0.018 | 0.070 | 1,143 | 1,122 |
| IB | 0.234 | 0.034 | 0.177 | 0.670 | 0.014 | 0.057 | 1,143 | 1,122 |
| $\text{IB}_{\text{val}}$ | 0.234 | 0.034 | 0.177 | 0.670 | 0.014 | 0.057 | 1,143 | 1,122 |
| UB | 0.248 | 0.039 | 0.195 | 0.684 | 0.024 | 0.091 | 1,143 | 1,122 |
| $\text{UB}_{\text{val}}$ | 0.247 | 0.039 | 0.194 | 0.674 | 0.021 | 0.081 | 1,143 | 1,122 |
| HKV | 0.257 | **0.043** | 0.202 | 0.725 | 0.038 | 0.122 | 1,143 | 1,122 |
| $\text{HKV}_{\text{val}}$ | 0.257 | **0.043** | 0.202 | 0.725 | 0.038 | 0.122 | 1,143 | 1,122 |
| BPRMF | 0.231 | 0.034 | 0.172 | 0.700 | 0.036 | 0.146 | 1,143 | 1,122 |
| $\text{BPRMF}_{\text{val}}$ | 0.213 | 0.027 | 0.155 | 0.595 | 0.005 | 0.025 | 1,143 | 1,122 |
| TD | 0.248 | 0.040 | 0.194 | 0.675 | 0.021 | 0.079 | 1,143 | 1,122 |
| $\text{TD}_{\text{val}}$ | 0.242 | 0.040 | 0.192 | 0.679 | 0.022 | 0.085 | 1,143 | 1,122 |
| MC | 0.177 | 0.029 | 0.134 | 0.708 | 0.010 | 0.039 | 1,143 | 1,122 |
| $\text{MC}_{\text{val}}$ | 0.165 | 0.027 | 0.124 | 0.714 | 0.009 | 0.037 | 1,143 | 1,122 |
| FPMC | 0.212 | 0.029 | 0.159 | 0.616 | 0.004 | 0.021 | 1,143 | 1,122 |
| $\text{FPMC}_{\text{val}}$ | 0.205 | 0.026 | 0.148 | 0.606 | 0.005 | 0.021 | 1,143 | 1,122 |
| Fossil | 0.227 | 0.036 | 0.170 | 0.650 | 0.009 | 0.048 | 1,143 | 1,122 |
| $\text{Fossil}_{\text{val}}$ | 0.215 | 0.029 | 0.169 | 0.624 | 0.004 | 0.022 | 1,143 | 1,122 |
| Caser | 0.192 | 0.029 | 0.136 | 0.788 | 0.089 | 0.238 | 1,143 | 1,122 |
| $\text{Caser}_{\text{val}}$ | 0.184 | 0.031 | 0.131 | 0.790 | 0.089 | 0.244 | 1,143 | 1,122 |
| Skyline | †**0.943** | 0.280 | †**1.000** | 0.871 | 0.131 | 0.368 | 1,762 | 1,762 |
| $\text{Skyline}_{\text{CF}}$ | 0.911 | †**0.363** | 0.999 | 0.883 | 0.133 | 0.322 | 1,122 | 1,122 |

there are many users with very few interactions, which means that the transitions between items do not provide much information. At the same time, in the random split, the Pop recommender is no longer the best one in terms of accuracy, as the UB and HKV (as in Movielens1M) are now the recommenders that perform the best. Besides, it is important to note how the Skyline has much less coverage than the rest of the algorithms. This is due to two reasons: first, there is no restriction on the minimum number of interactions in the training and test subsets, so there may be test users without interactions in training; secondly, this dataset has repeated interactions, so it may happen that when making recommendations, there are test users who have not visited any different POI with respect to the ones already rated in the training set.

As explained in Section 2.4.2, when performing offline experiments, in related areas like Machine Learning (ML) an additional subset is used for parameter tuning, the validation set. Although this is not so common in Recommender Systems, we now want to show here the different behavior of the recommenders when tuning the parameters using such a validation subset. Hence, we now show in Tables 4.9 and 4.10

**Table 4.10:** Performance results on Movielens1M dataset. Random system split (80% ratings to train, rest to test, randomly selected). Same configuration as in Table 4.9.

| Recommender | P | R | nDCG | EPC | Gini | IC | UC | UC-Rel |
|---|---|---|---|---|---|---|---|---|
| Rnd | 0.006 | 0.002 | 0.004 | †**0.967** | 0.786 | †**1.000** | †**6,036** | †**5,978** |
| $\mathrm{Rnd}_{\mathrm{CF}}$ | 0.006 | 0.002 | 0.004 | 0.966 | †**0.789** | 1.000 | †**6,036** | †**5,978** |
| Pop | 0.168 | 0.052 | 0.141 | 0.626 | 0.003 | 0.012 | †**6,036** | †**5,978** |
| $\mathrm{Pop}_{\mathrm{CF}}$ | 0.168 | 0.052 | 0.141 | 0.626 | 0.003 | 0.012 | †**6,036** | †**5,978** |
| UBCB | 0.281 | 0.099 | 0.249 | 0.703 | 0.015 | 0.080 | †**6,036** | †**5,978** |
| $\mathrm{UBCB}_{\mathrm{val}}$ | 0.281 | 0.099 | 0.249 | 0.703 | 0.015 | 0.080 | †**6,036** | †**5,978** |
| IB | 0.287 | 0.105 | 0.241 | 0.712 | 0.015 | 0.108 | †**6,036** | †**5,978** |
| $\mathrm{IB}_{\mathrm{val}}$ | 0.287 | 0.105 | 0.241 | 0.712 | 0.015 | 0.108 | †**6,036** | †**5,978** |
| UB | 0.330 | 0.133 | 0.300 | 0.714 | 0.019 | 0.106 | †**6,036** | †**5,978** |
| $\mathrm{UB}_{\mathrm{val}}$ | 0.329 | 0.131 | 0.300 | 0.710 | 0.018 | 0.099 | †**6,036** | †**5,978** |
| HKV | **0.350** | **0.142** | **0.316** | 0.787 | 0.061 | 0.219 | †**6,036** | †**5,978** |
| $\mathrm{HKV}_{\mathrm{val}}$ | **0.350** | **0.142** | **0.316** | 0.787 | 0.061 | 0.219 | †**6,036** | †**5,978** |
| BPRMF | 0.295 | 0.107 | 0.248 | 0.769 | 0.068 | 0.368 | †**6,036** | †**5,978** |
| $\mathrm{BPRMF}_{\mathrm{val}}$ | 0.295 | 0.107 | 0.248 | 0.769 | 0.068 | 0.368 | †**6,036** | †**5,978** |
| Skyline | †**0.914** | †**0.512** | †**1.000** | 0.859 | 0.200 | 0.606 | 5,978 | †**5,978** |
| $\mathrm{Skyline}_{\mathrm{CF}}$ | †**0.914** | †**0.512** | †**1.000** | 0.859 | 0.200 | 0.606 | 5,978 | †**5,978** |

the performance of the best recommenders according to nDCG@5 for Movielens1M as in the previous tables along with the corresponding best algorithm obtained with the same metric using a validation subset. The validation subset was created following the same split mechanism but splitting the training set into two subsets, the validation training and the validation test subsets. Hence, for the random split, we selected the training subset of the random split and maintained a random subset of 80% of the interactions for training and the rest for validation. Similarly, in the temporal split we obtain two subsets from the training split but applying a temporal split in this case. The recommenders optimized using the validation subset are denoted with the subscript "val".

Based on the results in these tables, we can observe that in most cases we achieve the same result using the training and the validation subsets, except for the models that have more parameters (like the matrix factorization and sequential approaches). This may be due to the fact that these algorithms optimize a model that starts from a high number of random factors and even if these factors are optimized according to a loss function, they are more sensitive to the data used for training them than other algorithms based on similarities. Hence, in Recommender Systems, where the user $\times$ item matrix is so sparse, the parameters of model-based recommenders tend to have a higher variance than other approaches. However, this effect may also be due to parameter tuning conducted in algorithms. As we can see, in Table 4.2, we have performed a relatively small parameter tuning, so if we were to increase the number of parameters to analyze (e.g. by increments of 5 by 5 in the case of neighborhoods), we would possibly find more differences between the proposals optimized with the test and validation subsets. Likewise, this opens an interesting debate in the Recommender Systems domain (and probably in other related areas such as Machine Learning) since sometimes

**Table 4.11:** Performance results on FS (Tokyo) dataset. Temporal system split (80% ancient ratings to train, rest to test). Same configuration as in Table 4.5. Allowing previous consumed items in train.

| Recommender | P | R | nDCG | EPC | Gini | IC | UC | UC-Rel |
|---|---|---|---|---|---|---|---|---|
| Rnd | 0.000 | 0.000 | 0.000 | †**0.999** | †**0.418** | †**0.562** | †**8,268** | †**8,268** |
| $\text{Rnd}_{\text{CF}}$ | 0.000 | 0.000 | 0.000 | †**0.999** | 0.409 | 0.540 | 7,615 | 7,615 |
| Pop | 0.160 | 0.080 | 0.185 | 0.723 | 0.000 | 0.000 | †**8,268** | †**8,268** |
| $\text{Pop}_{\text{CF}}$ | 0.162 | 0.079 | 0.187 | 0.723 | 0.000 | 0.000 | 7,615 | 7,615 |
| UBCB | 0.169 | 0.083 | 0.196 | 0.754 | 0.000 | 0.004 | 7,615 | 7,615 |
| IB | 0.170 | 0.080 | 0.191 | 0.885 | 0.157 | 0.346 | 7,611 | 7,611 |
| UB | **0.204** | **0.105** | **0.242** | 0.801 | 0.002 | 0.024 | 7,611 | 7,611 |
| HKV | 0.195 | 0.096 | 0.234 | 0.868 | 0.001 | 0.007 | 7,615 | 7,615 |
| BPRMF | 0.162 | 0.079 | 0.188 | 0.723 | 0.000 | 0.000 | 7,615 | 7,615 |
| TD | **0.204** | **0.105** | **0.242** | 0.793 | 0.001 | 0.020 | 7,611 | 7,611 |
| MC | 0.141 | 0.068 | 0.161 | 0.767 | 0.001 | 0.022 | 7,471 | 7,471 |
| FPMC | 0.125 | 0.066 | 0.151 | 0.899 | 0.003 | 0.038 | 7,471 | 7,471 |
| Fossil | 0.087 | 0.039 | 0.102 | 0.883 | 0.002 | 0.022 | 7,457 | 7,457 |
| Caser | 0.131 | 0.071 | 0.153 | 0.821 | 0.009 | 0.062 | 7,615 | 7,615 |
| Training | 0.201 | 0.104 | 0.237 | 0.872 | 0.007 | 0.058 | 7,615 | 7,615 |
| Skyline | 0.822 | †**0.609** | †**0.994** | 0.978 | 0.185 | 0.340 | 8,247 | 8,247 |
| $\text{Skyline}_{\text{CF}}$ | †**0.828** | 0.598 | †**0.994** | 0.979 | 0.184 | 0.328 | 7,597 | 7,597 |

when authors propose new algorithms, they perform comparisons against weak or not well-tuned baselines (Dacrema et al., 2019). As a conclusion, and based on these minor differences, in the rest of the thesis we will show results where recommenders are optimized according to the test set.

Finally, we want to analyze the effect of repeated interactions on the recommendation algorithms (a user consuming the same item several times). This analysis can only be performance on FS (Tokyo) since on Movielens1M users consume a movie only once. Hence, in Tables 4.11 and 4.12 we we use another recommendation strategy that we have named “ItemsInTraining”, which consists of recommending to users those items that appear in the training set without any additional restriction (i.e., all items in training are ordered from highest to lowest score, including those that have been previously consumed by the user). Our objective is to examine the difference in performance both in relevance and in novelty and diversity if we consider these repeated items when recommending them. In these tables, we also include a new algorithm, the training recommender, which only recommends for each user the same items that she has previously visited in the training set (denoted as Training). In this case we can observe how in both splits the recommender that only returns the training items obtains a very competitive result, being only closely outperformed by UB and TD. In addition, if we analyze the novelty and diversity performance, we can also observe that this recommender obtains significantly higher results than the rest of the algorithms. This implies two important considerations. First, this behavior reinforces the idea that in this type of datasets, previous rated items have a great impact on recommendations.

**Table 4.12:** Performance results on FS (Tokyo) dataset. Random system split (80% ratings to train, rest to test, randomly selected). Same configuration as in Table 4.11.

| Recommender | P | R | nDCG | EPC | Gini | IC | UC | UC-Rel |
|---|---|---|---|---|---|---|---|---|
| Rnd | 0.000 | 0.000 | 0.000 | †**0.999** | 0.476 | 0.642 | †**10,463** | †**10,463** |
| $\text{Rnd}_{\text{CF}}$ | 0.000 | 0.000 | 0.000 | †**0.999** | †**0.478** | †**0.643** | 10,432 | 10,432 |
| Pop | 0.145 | 0.080 | 0.169 | 0.727 | 0.000 | 0.000 | †**10,463** | †**10,463** |
| $\text{Pop}_{\text{CF}}$ | 0.145 | 0.080 | 0.169 | 0.727 | 0.000 | 0.000 | 10,432 | 10,432 |
| UBCB | 0.153 | 0.088 | 0.182 | 0.763 | 0.000 | 0.006 | 10,432 | 10,432 |
| IB | 0.172 | 0.087 | 0.191 | 0.883 | 0.174 | 0.411 | 10,430 | 10,430 |
| UB | **0.204** | **0.124** | **0.243** | 0.827 | 0.002 | 0.029 | 10,430 | 10,430 |
| HKV | 0.197 | 0.110 | 0.236 | 0.877 | 0.002 | 0.007 | 10,432 | 10,432 |
| BPRMF | 0.147 | 0.081 | 0.171 | 0.732 | 0.000 | 0.008 | 10,432 | 10,432 |
| Training | 0.203 | 0.121 | 0.240 | 0.882 | 0.009 | 0.078 | 10,432 | 10,432 |
| Skyline | 0.806 | †**0.641** | 0.998 | 0.977 | 0.238 | 0.421 | 10,456 | 10,456 |
| $\text{Skyline}_{\text{CF}}$ | †**0.807** | 0.640 | †**0.999** | 0.978 | 0.238 | 0.421 | 10,425 | 10,425 |

Second, despite the fact that the recommended items from the training recommender are very different from each other (in terms of novelty and diversity), the other recommenders are suffering from a severe popularity bias. Hence, these algorithms are learning the patterns derived from the most popular items, as we can infer from the lower values obtained in novelty and diversity. In the Appendix A.2.3 we explore again the effect of recommendations of already consumed items.

### 4.5.3 Analyzing Time-Aware Novelty metrics

In this section we analyze the behavior of the recommenders on our proposed Time-Aware Novelty metrics from Section 4.1. As these metrics are oriented to measure how novel the items are according to a temporal profile, we will show the results only on the temporal split. For the sake of simplicity, we will ignore in our time-aware novelty model both the discount and relevance (i.e., in Equation 4.1 we only take into account the temporal novelty). Thus, in Tables 4.13 and 4.14 we show the results of the best recommenders previously shown in Movielens1M and FS (Tokyo) datasets respectively. In both tables we show the four different metrics of our proposed models, **FIN**, that analyze the first timestamp of the items, **AIN**, that takes into account the average, **MIN**, that uses the median, and finally **LIN** that only works with the last timestamp associated with each item (all these metrics were explained in Section 4.1). If we analyze the results, we can observe some interesting and illustrative behaviors. Firstly, the Pop recommender has the lowest value of FIN but at the same time the highest value of LIN, indicating that the most popular items are consumed throughout all the life of the system in both datasets. On the other hand, in Movielens1M we can see that the Skyline recommenders have a slightly higher value in our time-aware metrics than the rest of the recommenders, showing that sometimes analyzing the temporal novelty should also be taken into account for producing relevant recommendations. In the case of FS (Tokyo), we can see that in the Skyline the value of FIN is much higher

**Table 4.13:** Performance results on Movielens1M dataset. Temporal system split (80% ancient ratings to train, rest to test) for Time-Aware Novelty metrics. Same configuration as in Table 4.5.

| Recommender | nDCG | EPC | Gini | IC | FIN | AIN | MIN | LIN | UC | UC-Rel |
|---|---|---|---|---|---|---|---|---|---|---|
| Rnd | 0.012 | 0.962 | †**0.651** | †**0.920** | **0.118** | **0.630** | **0.616** | 0.971 | †**1,783** | †**1,762** |
| $\text{Rnd}_{\text{CF}}$ | 0.008 | †**0.964** | 0.565 | 0.784 | 0.112 | 0.626 | 0.611 | 0.972 | 1,143 | 1,122 |
| Pop | **0.221** | 0.566 | 0.004 | 0.025 | 0.000 | 0.614 | 0.592 | †**1.000** | †**1,783** | †**1,762** |
| $\text{Pop}_{\text{CF}}$ | 0.161 | 0.587 | 0.005 | 0.025 | 0.000 | 0.613 | 0.591 | †**1.000** | 1,143 | 1,122 |
| UBCB | 0.195 | 0.669 | 0.018 | 0.070 | 0.001 | 0.608 | 0.579 | 0.999 | 1,143 | 1,122 |
| IB | 0.177 | 0.670 | 0.014 | 0.057 | 0.001 | 0.605 | 0.570 | 0.999 | 1,143 | 1,122 |
| UB | 0.195 | 0.684 | 0.024 | 0.091 | 0.004 | 0.610 | 0.581 | 0.999 | 1,143 | 1,122 |
| HKV | 0.202 | 0.725 | 0.038 | 0.122 | 0.005 | 0.611 | 0.585 | 0.999 | 1,143 | 1,122 |
| BPRMF | 0.172 | 0.700 | 0.036 | 0.146 | 0.003 | 0.614 | 0.591 | 0.999 | 1,143 | 1,122 |
| TD | 0.194 | 0.675 | 0.021 | 0.079 | 0.004 | 0.612 | 0.587 | 0.999 | 1,143 | 1,122 |
| MC | 0.134 | 0.708 | 0.010 | 0.039 | 0.028 | 0.629 | 0.614 | 0.999 | 1,143 | 1,122 |
| FPMC | 0.159 | 0.616 | 0.004 | 0.021 | 0.001 | 0.606 | 0.577 | 0.999 | 1,143 | 1,122 |
| Fossil | 0.170 | 0.650 | 0.009 | 0.048 | 0.004 | 0.613 | 0.591 | 0.999 | 1,143 | 1,122 |
| Caser | 0.136 | 0.788 | 0.089 | 0.238 | 0.025 | 0.626 | 0.609 | 0.999 | 1,143 | 1,122 |
| Skyline | †**1.000** | 0.871 | 0.131 | 0.368 | 0.136 | 0.666 | 0.661 | 0.998 | 1,762 | †**1,762** |
| $\text{Skyline}_{\text{CF}}$ | 0.999 | 0.883 | 0.133 | 0.322 | †**0.145** | †**0.671** | †**0.670** | 0.997 | 1,122 | 1,122 |

than the rest of the algorithms (except for Rnd). This may indicate that there is a significant popularity bias in this dataset, since most of the items being returned by the recommenders have a very low FIN (i.e., they are the items that have low timestamps) and therefore have existed for a longer period of time in the system and hence they are more likely to have been visited more.

Now, if we analyze the personalized recommendation algorithms and, specifically, the time-aware and sequential ones (i.e., TD, MC, FPMC, Fossil, and Caser), we find the following. First, it is important to note that, except TD, the rest of the sequential recommenders, will try to predict items that fit the sequence of the user. This means that sometimes they can obtain a worse result in this type of metric than other proposals that do not exploit the temporal context (such as the case of FPMC that obtains a worse result in Movielens1M than the BPRMF). Nevertheless, we can observe that the performance of TD is always slightly superior with respect to UB as it will prioritize the items recommended if their timestamps are higher. However, in Movielens1M, the behavior of the rest of sequential recommenders needs a further analysis. As we can observe, the MC and Caser algorithms have a higher temporal novelty than the FPMC and Fossil models. To analyze this effect, it is desirable to analyze well the difference of the temporal splits of both Movielens1M and FS (Tokyo). As shown in Table 4.4, in the Movielens1M temporal split there are very few users in the test set compared to the total number of users in the full dataset, while in the FS (Tokyo) temporal split this effect is not so critical. This affects the distribution of ratings, because it means that there is a significant number of users in the Movielens1M dataset that are not active in the test set. Besides, Fossil and FPMC are personalized, that is, that these models are optimizing the sequences for each user while MC not. On the other hand, Caser, as we can see in Figure 2.1, uses the sequences of the user but only uses the latent representation of the user in the final step, so the convolutional layers are applied to

**Table 4.14:** Performance results on FS (Tokyo) dataset. Temporal system split (80% ancient ratings to train, rest to test). Same configuration as in Table 4.13.

| Recommender | nDCG | EPC | Gini | IC | FIN | AIN | MIN | LIN | UC | UC-Rel |
|---|---|---|---|---|---|---|---|---|---|---|
| Rnd | 0.000 | †**0.999** | †**0.419** | †**0.564** | †**0.232** | 0.474 | 0.468 | 0.733 | †**8,268** | †**8,268** |
| $\text{Rnd}_{\text{CF}}$ | 0.000 | †**0.999** | 0.404 | 0.536 | †**0.232** | 0.475 | 0.470 | 0.736 | 7,615 | 7,615 |
| Pop | **0.078** | 0.767 | 0.000 | 0.000 | 0.001 | 0.483 | 0.535 | †**1.000** | †**8,268** | †**8,268** |
| $\text{Pop}_{\text{CF}}$ | 0.070 | 0.771 | 0.000 | 0.000 | 0.001 | 0.483 | 0.534 | †**1.000** | 7,615 | 7,615 |
| UBCB | 0.072 | 0.802 | 0.000 | 0.005 | 0.002 | 0.482 | 0.527 | 0.999 | 7,615 | 7,615 |
| IB | 0.066 | 0.854 | 0.026 | 0.132 | 0.131 | **0.509** | **0.544** | 0.918 | 7,611 | 7,611 |
| UB | 0.076 | 0.795 | 0.000 | 0.009 | 0.003 | 0.480 | 0.523 | 0.998 | 7,611 | 7,611 |
| HKV | 0.070 | 0.857 | 0.001 | 0.005 | 0.003 | 0.481 | 0.516 | 0.999 | 7,615 | 7,615 |
| BPRMF | 0.071 | 0.772 | 0.000 | 0.001 | 0.001 | 0.483 | 0.531 | 0.999 | 7,615 | 7,615 |
| TD | 0.077 | 0.796 | 0.000 | 0.011 | 0.003 | 0.482 | 0.527 | 0.998 | 7,611 | 7,611 |
| MC | 0.057 | 0.812 | 0.001 | 0.020 | 0.005 | 0.473 | 0.496 | 0.997 | 7,471 | 7,471 |
| FPMC | 0.047 | 0.853 | 0.000 | 0.003 | 0.005 | 0.481 | 0.517 | 0.997 | 7,457 | 7,457 |
| Fossil | 0.056 | 0.894 | 0.002 | 0.022 | 0.011 | 0.483 | 0.526 | 0.989 | 7,457 | 7,457 |
| Caser | 0.054 | 0.861 | 0.011 | 0.070 | 0.014 | 0.481 | 0.513 | 0.986 | 7,615 | 7,615 |
| Skyline | †**0.933** | 0.984 | 0.164 | 0.301 | 0.201 | 0.540 | †**0.550** | 0.886 | 7,900 | 7,900 |
| $\text{Skyline}_{\text{CF}}$ | 0.928 | 0.985 | 0.163 | 0.289 | 0.205 | †**0.541** | †**0.550** | 0.883 | 7,250 | 7,250 |

the sequences ignoring the users to which they belong. This may indicate that MC and Caser learn more about the patterns globally while Fossil and FPMC learn at the user level and if there are many users who have not been in the system for a long time, they will produce less novel recommendations at the temporal level. This effect is not seen in the FS (Tokyo) dataset, probably because in this case, the number of users being tested is closer to the number of users in the entire set (they are more active throughout the life of the system).

### 4.5.4 Analyzing Anti-relevance metrics

Let us analyze now the results of the recommenders with the anti-relevance metrics shown in Section 4.2. In this case we will only show the results on the Movielens1M dataset as the anti-relevance model only works on datasets with explicit data interactions, that is, ratings. Thus, in Tables 4.15 and 4.16 we show the performance of the recommenders in some classic ranking accuracy metrics with their anti-relevance counterparts while keeping the user coverage. Recall that this type of metrics is intended to measure the number of anti-relevant items that have been recommended to the user (in this case, we have decided to consider as anti-relevant the items scored with a 2 or less). In the same way, these metrics measure the complementary, so the closer to 1, the fewer anti-relevant items we will be recommending. In those tables we also include the $\overline{\text{Skyline}}$ recommender that represents the complement of the Skyline model. That is, the $\overline{\text{Skyline}}$ will recommend to the users the items rated by the user with a 2 or less. This recommender serves to check the minimum value that can be obtained with the anti-relevance metrics, and, as we can see, it does not obtain values of 0 in the anti-relevance metrics because there may be users who have not rated a high number of anti-relevant items.

Based on these results, we can observe an interesting behavior: the Rnd recom-

**Table 4.15:** Performance results on Movielens1M dataset. Temporal system split (80% ancient ratings to train, rest to test). Anti-relevance metrics. Same configuration as in Table 4.5.

| **Recommender** | **P** | $\overline{\textbf{P}}$ | **R** | $\overline{\textbf{R}}$ | **nDCG** | $\overline{\textbf{nDCG}}$ | **UC** | **UC-Rel** |
|---|---|---|---|---|---|---|---|---|
| Rnd | 0.019 | 0.993 | 0.001 | 0.999 | 0.012 | 0.996 | †**1,783** | †**1,762** |
| $\text{Rnd}_{\text{CF}}$ | 0.015 | **0.994** | 0.001 | **0.999** | 0.008 | **0.997** | 1,143 | 1,122 |
| Pop | **0.281** | 0.977 | 0.030 | 0.990 | **0.221** | 0.981 | †**1,783** | †**1,762** |
| $\text{Pop}_{\text{CF}}$ | 0.210 | 0.979 | 0.026 | 0.990 | 0.161 | 0.983 | 1,143 | 1,122 |
| UBCB | 0.254 | 0.979 | 0.039 | 0.991 | 0.195 | 0.985 | 1,143 | 1,122 |
| IB | 0.234 | 0.979 | 0.034 | 0.990 | 0.177 | 0.987 | 1,143 | 1,122 |
| UB | 0.248 | 0.985 | 0.039 | 0.992 | 0.195 | 0.990 | 1,143 | 1,122 |
| HKV | 0.257 | 0.985 | **0.043** | 0.991 | 0.202 | 0.990 | 1,143 | 1,122 |
| BPRMF | 0.231 | 0.975 | 0.034 | 0.988 | 0.172 | 0.983 | 1,143 | 1,122 |
| TD | 0.248 | 0.987 | 0.040 | 0.993 | 0.194 | 0.990 | 1,143 | 1,122 |
| MC | 0.177 | 0.972 | 0.029 | 0.988 | 0.134 | 0.978 | 1,143 | 1,122 |
| FPMC | 0.212 | 0.979 | 0.029 | 0.991 | 0.159 | 0.985 | 1,143 | 1,122 |
| Fossil | 0.227 | 0.974 | 0.036 | 0.989 | 0.170 | 0.984 | 1,143 | 1,122 |
| Caser | 0.192 | 0.969 | 0.029 | 0.990 | 0.136 | 0.977 | 1,143 | 1,122 |
| Skyline | †**0.943** | †**1.000** | 0.280 | †**1.000** | †**1.000** | †**1.000** | 1,762 | 1,762 |
| $\text{Skyline}_{\text{CF}}$ | 0.911 | †**1.000** | †**0.363** | †**1.000** | 0.999 | †**1.000** | 1,122 | 1,122 |
| $\overline{\text{Skyline}}$ | 0.000 | 0.189 | 0.000 | 0.397 | 0.000 | 0.001 | 1,519 | 1,507 |
| $\overline{\text{Skyline}_{\text{CF}}}$ | 0.000 | 0.221 | 0.000 | 0.376 | 0.000 | 0.001 | 914 | 902 |

mender is the best one according to the anti-relevance metrics. Although this may be contradictory, this effect can be explained as follows: as we have mentioned, these metrics measure the number of anti-relevant items we are recommending to the users, but in the same way that the Rnd recommender has a low probability of recovering a relevant item, it is also difficult for it to recover an anti-relevant item. This leads towards an interesting dichotomy. The fact of learning the tastes of users and recommending relevant items sometimes also causes the algorithms to recommend items that the users do not like, and that is something we should pay attention to, because a failure in a recommendation (understanding as a failure the recommendation of an item that the user specifically does not like) affects the user more than a recommendation that is neither relevant nor anti-relevant.

With respect to the recommenders, we can observe how in the temporal split the recommenders which work with implicit data (MC, FPMC, Fossil, and BPRMF) obtain the worst results in terms of relevance and anti-relevance. This may be due to the fact that these algorithms do not work with ratings, so in this case it may be detrimental to this type of metrics. In fact, those models consider every item as relevant, and in the case of the sequential recommenders, they try to predict the item that the user will consume next, without considering the rating. Another special case can be seen in the Pop recommender, which in the temporal split obtains worse results in the anti-relevance metrics than other personalized algorithms such as UB or HKV, whereas in the random split the Pop recommender is one of the best ones in this dimension. This may be due to the variation in the number of users in the test set (see the UC

**Table 4.16:** Performance results on Movielens1M dataset. Random split (80% ratings to train, rest to test, randomly selected). Same configuration as in Table 4.15.

| **Recommender** | **P** | $\overline{\textbf{P}}$ | **R** | $\overline{\textbf{R}}$ | **nDCG** | $\overline{\textbf{nDCG}}$ | **UC** | **UC-Rel** |
|---|---|---|---|---|---|---|---|---|
| Rnd | 0.006 | 0.998 | 0.002 | 0.999 | 0.004 | 0.998 | †**6,036** | †**5,978** |
| Rnd$_{\text{CF}}$ | 0.006 | **0.998** | 0.002 | **0.999** | 0.004 | **0.998** | †**6,036** | †**5,978** |
| Pop | 0.168 | 0.987 | 0.052 | 0.984 | 0.141 | 0.986 | †**6,036** | †**5,978** |
| Pop$_{\text{CF}}$ | 0.168 | 0.987 | 0.052 | 0.984 | 0.141 | 0.986 | †**6,036** | †**5,978** |
| UBCB | 0.281 | 0.986 | 0.099 | 0.982 | 0.249 | 0.986 | †**6,036** | †**5,978** |
| IB | 0.287 | 0.980 | 0.105 | 0.979 | 0.241 | 0.982 | †**6,036** | †**5,978** |
| UB | 0.330 | 0.982 | 0.133 | 0.975 | 0.300 | 0.980 | †**6,036** | †**5,978** |
| HKV | **0.350** | 0.986 | **0.142** | 0.981 | **0.316** | 0.985 | †**6,036** | †**5,978** |
| BPRMF | 0.295 | 0.972 | 0.107 | 0.970 | 0.248 | 0.972 | †**6,036** | †**5,978** |
| Skyline | †**0.914** | †**1.000** | †**0.512** | †**1.000** | †**1.000** | †**1.000** | 5,978 | †**5,978** |
| Skyline$_{\text{CF}}$ | †**0.914** | †**1.000** | †**0.512** | †**1.000** | †**1.000** | †**1.000** | 5,978 | †**5,978** |
| $\overline{\text{Skyline}}$ | 0.000 | 0.367 | 0.000 | 0.183 | 0.000 | 0.000 | 4,556 | 4,509 |
| $\overline{\text{Skyline}_{\text{CF}}}$ | 0.000 | 0.367 | 0.000 | 0.183 | 0.000 | 0.000 | 4,556 | 4,509 |

metric), since in the temporal split there are far fewer users, and therefore, failing the recommendations affects more the average in the metrics. Finally, it should be noted that the Skyline always gets 1 in all anti-relevant metrics because it only recommends test items with a score higher than 4.

### 4.5.5 Analyzing user attributes

Tables 4.18 and 4.19 show the results in the Movielens1M dataset comparing the performance of nDCG@5 (denoted as Std) against different aggregations based on user attributes. In Table 4.17 we show the percentage of users in the test subset for both datasets under the two splits. Regarding the user attributes, we decided to group the user ages in four intervals for Movielens1M: [1, 18), [18, 35), [35, 56) and [56, $+\infty$). Besides, for both datasets we have also created 4 user groups based on the quartiles obtained from the number of items consumed by each user, either in the training or in the test set. Hence, users are grouped in quartiles Q1-Q4 according to the number of preferences of each user in the corresponding set (ordered from lowest to highest number of preferences). Finally, again, for both datasets, we also exploited the gender of the users. Note that in that table there might be users that do not match any attribute. For example, when computing the quartiles for training, the new users that appear in the test set will not belong to any quartile. At the same time, we do not include information about the ages of the users in FS (Tokyo) as this information is not available.

Returning to the aforementioned tables, by analyzing the first attribute (gender), we observe the performance is generally much lower for females than for males in every recommender except Rnd. This can be explained if we take into account that there is a large difference between males and females in both the number of users and their activity in the system (see Table 4.17). Regarding the age, we can see an interesting

**Table 4.17:** User features of the Movielens1M and the FS (Tokyo) datasets in the temporal and random splits. The columns indicate the percentage of the users in the test set matching the user feature.

| | | **Movielens1M** | | **FS (Tokyo)** | |
|---|---|---|---|---|---|
| **Feature** | **Value** | **Temporal** | **Random** | **Temporal** | **Random** |
| Genre | Female | 27% | 28% | 11% | 11% |
| | Male | 73% | 72% | 80% | 80% |
| Age | 1 | 4% | 4% | - | - |
| | 18 | 57% | 53% | - | - |
| | 35 | 34% | 37% | - | - |
| | 56 | 5% | 6% | - | - |
| Training Quartiles | Q1 | 11% | 26% | 14% | 20% |
| | Q2 | 11% | 24% | 22% | 26% |
| | Q3 | 26% | 25% | 26% | 27% |
| | Q4 | 15% | 25% | 29% | 27% |
| Test Quartiles | Q1 | 26% | 28% | 26% | 28% |
| | Q2 | 24% | 23% | 25% | 25% |
| | Q3 | 25% | 24% | 24% | 22% |
| | Q4 | 25% | 25% | 25% | 25% |

behavior. While people who are over 56 years old have a worse result (they represent a very small percentage of users), in the case of 18 years old, also representing a minority group, they obtain relatively high results. Perhaps the explanation may be that users under 18 are more like people between 18 and 35 than those over 56. Finally, about the quartiles, in general the results are higher in the upper quartiles. This is an expected behavior because the higher the number of the quartile, the larger the user profile, and therefore the recommenders are able to have more information about the user. However, it is interesting to observe how the split type again affects the performance of the recommender depending on the characteristics of the users. Thus we can see how in the temporal split, the best recommender varies for each feature while in the case of a random split the best recommender is always the HKV.

Now, let us analyze the behavior on the FS (Tokyo). We now show the results under the temporal split in Table 4.20 and the results in the random split in Table 4.21. In this dataset we can observe again that in the two splits, the users identified as women obtain a lower performance than the rest of the users who identified as men. However, although the difference between the percentage of users is larger than in the Movielens1M dataset, the difference in the result of the recommenders is lower than the one reported in the movies domain. This may be due to the type of items that we are recommending in each dataset. While in the case of Movielens1M we are recommending movies that may have a very significant bias in the gender of the users, in the case of POIs this gender bias may not be as important as, for example, the type of "traveler" the user might be. More specifically, the check-in behavior of a user who visits a city on a business trip might be different than another user who visited the same city for leisure. However,

**Table 4.18:** Performance results on Movielens1M dataset with different user attributes. Temporal system split (80% ancient ratings to train, rest to test). nDCG@5. Same configuration as in Table 4.5.

| | | **Gender** | | **Age** | | | | **Training Quartile** | | | | **Test Quartile** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Family** | **Std** | **F** | **M** | **1** | **18** | **35** | **56** | **Q1** | **Q2** | **Q3** | **Q4** | **Q1** | **Q2** | **Q3** | **Q4** |
| Rnd | 0.012 | 0.011 | 0.012 | 0.011 | 0.014 | 0.009 | 0.005 | 0.007 | 0.014 | 0.008 | 0.009 | 0.003 | 0.004 | 0.016 | 0.023 |
| $\text{Rnd}_{\text{CF}}$ | 0.008 | 0.010 | 0.008 | 0.003 | 0.009 | 0.009 | 0.000 | 0.011 | 0.009 | 0.005 | 0.009 | 0.002 | 0.003 | 0.006 | 0.027 |
| Pop | **0.221** | 0.177 | **0.238** | 0.192 | **0.250** | **0.190** | 0.132 | 0.191 | 0.163 | 0.163 | 0.147 | 0.055 | 0.160 | 0.260 | **0.406** |
| $\text{Pop}_{\text{CF}}$ | 0.161 | 0.131 | 0.171 | 0.185 | 0.178 | 0.135 | 0.101 | 0.191 | 0.163 | 0.163 | 0.147 | 0.043 | 0.114 | 0.219 | 0.344 |
| UBCB | 0.195 | 0.177 | 0.202 | 0.195 | 0.206 | 0.180 | 0.164 | 0.225 | **0.193** | 0.183 | 0.191 | 0.057 | **0.178** | 0.264 | 0.368 |
| IB | 0.177 | 0.153 | 0.185 | 0.168 | 0.187 | 0.161 | 0.166 | 0.213 | 0.172 | 0.186 | 0.158 | 0.052 | 0.144 | 0.239 | 0.351 |
| UB | 0.195 | 0.173 | 0.202 | 0.194 | 0.208 | 0.176 | 0.160 | 0.207 | 0.170 | 0.199 | **0.198** | 0.067 | 0.165 | 0.274 | 0.352 |
| HKV | 0.202 | **0.184** | 0.209 | **0.207** | 0.213 | 0.185 | **0.191** | **0.235** | 0.185 | **0.210** | 0.192 | **0.074** | 0.166 | **0.284** | 0.366 |
| BPRMF | 0.172 | 0.166 | 0.175 | 0.180 | 0.179 | 0.164 | 0.144 | 0.201 | 0.172 | 0.184 | 0.154 | 0.056 | 0.144 | 0.232 | 0.330 |
| TD | 0.194 | 0.176 | 0.200 | 0.188 | 0.205 | 0.178 | 0.171 | 0.202 | 0.173 | 0.203 | 0.194 | 0.066 | 0.162 | 0.270 | 0.358 |
| MC | 0.134 | 0.127 | 0.137 | 0.127 | 0.142 | 0.123 | 0.122 | 0.169 | 0.130 | 0.144 | 0.115 | 0.052 | 0.109 | 0.170 | 0.257 |
| FPMC | 0.159 | 0.139 | 0.166 | 0.196 | 0.176 | 0.134 | 0.085 | 0.192 | 0.162 | 0.145 | 0.152 | 0.044 | 0.124 | 0.215 | 0.327 |
| Fossil | 0.170 | 0.178 | 0.168 | 0.160 | 0.177 | 0.160 | 0.172 | 0.211 | 0.170 | 0.183 | 0.146 | 0.062 | 0.134 | 0.221 | 0.333 |
| Caser | 0.136 | 0.141 | 0.135 | 0.114 | 0.143 | 0.128 | 0.129 | 0.171 | 0.144 | 0.137 | 0.118 | 0.044 | 0.109 | 0.202 | 0.248 |
| Skyline | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†0.999** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†0.999** | **†1.000** | **†1.000** | **†0.999** | **†1.000** |
| $\text{Skyline}_{\text{CF}}$ | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†0.999** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†0.999** | **†1.000** | **†1.000** | **†0.999** | **†1.000** |

**Table 4.19:** Performance results on Movielens1M dataset with different user attributes. Random system split (80% ratings to train, rest to test, randomly selected). Same configuration as in Table 4.18.

| | | **Gender** | | **Age** | | | | **Training Quartile** | | | | **Test Quartile** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Family** | **Std** | **F** | **M** | **1** | **18** | **35** | **56** | **Q1** | **Q2** | **Q3** | **Q4** | **Q1** | **Q2** | **Q3** | **Q4** |
| Rnd | 0.004 | 0.003 | 0.004 | 0.001 | 0.004 | 0.004 | 0.001 | 0.002 | 0.002 | 0.003 | 0.008 | 0.002 | 0.001 | 0.004 | 0.008 |
| $\text{Rnd}_{\text{CF}}$ | 0.004 | 0.003 | 0.004 | 0.002 | 0.004 | 0.004 | 0.002 | 0.001 | 0.002 | 0.003 | 0.009 | 0.001 | 0.002 | 0.003 | 0.009 |
| Pop | 0.141 | 0.093 | 0.159 | 0.108 | 0.168 | 0.116 | 0.072 | 0.066 | 0.082 | 0.136 | 0.278 | 0.064 | 0.084 | 0.141 | 0.275 |
| $\text{Pop}_{\text{CF}}$ | 0.141 | 0.093 | 0.159 | 0.108 | 0.168 | 0.116 | 0.072 | 0.066 | 0.082 | 0.136 | 0.278 | 0.064 | 0.084 | 0.141 | 0.275 |
| UBCB | 0.249 | 0.203 | 0.267 | 0.202 | 0.276 | 0.226 | 0.182 | 0.134 | 0.186 | 0.262 | 0.413 | 0.138 | 0.186 | 0.266 | 0.410 |
| IB | 0.241 | 0.196 | 0.259 | 0.220 | 0.259 | 0.227 | 0.186 | 0.142 | 0.192 | 0.250 | 0.381 | 0.142 | 0.194 | 0.254 | 0.380 |
| UB | 0.300 | 0.249 | 0.320 | 0.249 | 0.325 | 0.279 | 0.242 | 0.204 | 0.243 | 0.310 | 0.442 | 0.200 | 0.248 | 0.315 | 0.441 |
| HKV | **0.316** | **0.265** | **0.337** | **0.277** | **0.337** | **0.300** | **0.259** | **0.219** | **0.260** | **0.329** | **0.457** | **0.213** | **0.263** | **0.335** | **0.459** |
| BPRMF | 0.248 | 0.210 | 0.262 | 0.199 | 0.266 | 0.236 | 0.187 | 0.147 | 0.190 | 0.262 | 0.391 | 0.143 | 0.193 | 0.266 | 0.394 |
| Skyline | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** |
| $\text{Skyline}_{\text{CF}}$ | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** | **†1.000** |

users that are “locals” from a city do not visit the same types of POIs as other users that are “tourists”, so this attribute might be more important than the gender in this domain.

However, when we analyze the quartiles we do find some interesting results. Although in the test quartiles, the higher the quartile, the better the results are obtained in both types of split (which makes sense, since the higher the number of items in the test, the more likely it is that the recommendations will match at least one item), this is not the case in the training quartiles for the temporal split, since the lower the quartile, the better the performance of the recommendation. This can be due to the well-known effect of the popularity bias. If a user has few items in training it may be normal to recommend the most popular items, especially in areas with as much sparsity as in the POI recommendation domain. This effect can be combined with the fact that there is a set of users belonging to the first quartile that might be becoming more active in the moment of the split (e.g., tourists that have just arrive in the city). Besides, in this type of split there may be bursts of activities depending on the time context (events such as concerts, sports activities, exhibitions, etc.) and for that reason there may be users who have been very active in training and not in test. On the other hand, in the

**Table 4.20:** Performance results on FS (Tokyo) dataset. Temporal system split (80% ancient ratings to train, rest to test). Same configuration as in Table 4.18.

| | | Gender | | Training Quartile | | | | Test Quartile | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Family** | **Std** | **F** | **M** | **Q1** | **Q2** | **Q3** | **Q4** | **Q1** | **Q2** | **Q3** | **Q4** |
| Rnd | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| $\text{Rnd}_{\text{CF}}$ | 0.000 | 0.001 | 0.000 | 0.000 | 0.000 | 0.001 | 0.001 | 0.000 | 0.000 | 0.000 | 0.001 |
| Pop | **0.078** | 0.066 | **0.076** | 0.089 | 0.077 | 0.057 | 0.069 | 0.037 | **0.053** | **0.080** | **0.141** |
| $\text{Pop}_{\text{CF}}$ | 0.070 | 0.062 | 0.069 | 0.089 | 0.077 | 0.057 | 0.069 | 0.033 | 0.046 | 0.072 | 0.128 |
| UBCB | 0.072 | 0.065 | 0.071 | 0.083 | 0.078 | 0.061 | 0.073 | 0.033 | 0.046 | 0.073 | 0.132 |
| IB | 0.066 | 0.053 | 0.066 | 0.061 | 0.073 | 0.058 | 0.070 | 0.029 | 0.040 | 0.068 | 0.124 |
| UB | 0.076 | 0.066 | 0.075 | **0.093** | 0.078 | **0.065** | **0.077** | 0.037 | 0.049 | 0.077 | 0.138 |
| HKV | 0.070 | 0.059 | 0.069 | 0.081 | 0.076 | 0.061 | 0.067 | 0.035 | 0.046 | 0.070 | 0.125 |
| BPRMF | 0.071 | 0.062 | 0.070 | 0.090 | 0.077 | 0.058 | 0.070 | 0.033 | 0.046 | 0.074 | 0.129 |
| TD | 0.077 | **0.068** | 0.075 | 0.093 | **0.081** | 0.064 | 0.077 | **0.038** | 0.047 | 0.078 | 0.140 |
| MC | 0.057 | 0.051 | 0.057 | 0.074 | 0.063 | 0.047 | 0.054 | 0.027 | 0.037 | 0.059 | 0.101 |
| FPMC | 0.047 | 0.044 | 0.046 | 0.079 | 0.055 | 0.037 | 0.035 | 0.027 | 0.035 | 0.049 | 0.073 |
| Fossil | 0.056 | 0.043 | 0.056 | 0.078 | 0.068 | 0.045 | 0.048 | 0.030 | 0.037 | 0.059 | 0.095 |
| Caser | 0.054 | 0.046 | 0.054 | 0.083 | 0.076 | 0.046 | 0.032 | 0.033 | 0.039 | 0.060 | 0.084 |
| Skyline | †**0.933** | †**0.950** | †**0.931** | †**0.944** | †**0.902** | †**0.910** | †**0.954** | †**0.903** | †**0.851** | †**0.975** | †**0.999** |
| $\text{Skyline}_{\text{CF}}$ | 0.928 | 0.944 | 0.926 | †**0.944** | †**0.902** | †**0.910** | †**0.954** | 0.889 | 0.836 | 0.973 | †**0.999** |

**Table 4.21:** Performance results on FS (Tokyo) dataset. Random system split (80% ratings to train, rest to test, randomly selected). Same configuration as in Table 4.18.

| | | Gender | | Training Quartile | | | | Test Quartile | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Family** | **Std** | **F** | **M** | **Q1** | **Q2** | **Q3** | **Q4** | **Q1** | **Q2** | **Q3** | **Q4** |
| Rnd | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.001 | 0.000 | 0.000 | 0.000 | 0.001 |
| $\text{Rnd}_{\text{CF}}$ | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| Pop | 0.052 | 0.047 | 0.051 | 0.051 | 0.047 | 0.047 | 0.064 | 0.046 | 0.047 | 0.049 | 0.068 |
| $\text{Pop}_{\text{CF}}$ | 0.052 | 0.047 | 0.051 | 0.051 | 0.047 | 0.047 | 0.064 | 0.046 | 0.047 | 0.049 | 0.068 |
| UBCB | 0.057 | 0.052 | 0.057 | 0.057 | 0.051 | 0.053 | 0.069 | 0.051 | 0.050 | 0.055 | 0.075 |
| IB | 0.056 | 0.049 | 0.055 | 0.046 | 0.050 | 0.053 | 0.072 | 0.043 | 0.050 | 0.055 | 0.077 |
| UB | **0.064** | **0.061** | **0.063** | **0.067** | **0.056** | 0.057 | **0.078** | **0.059** | 0.055 | 0.061 | **0.083** |
| HKV | **0.064** | **0.061** | 0.062 | 0.061 | **0.056** | **0.063** | 0.073 | 0.055 | **0.056** | **0.065** | 0.080 |
| BPRMF | 0.053 | 0.047 | 0.052 | 0.050 | 0.047 | 0.048 | 0.066 | 0.045 | 0.047 | 0.051 | 0.070 |
| Skyline | †**0.923** | †**0.938** | †**0.923** | †**0.932** | †**0.844** | †**0.917** | †**0.993** | †**0.912** | †**0.825** | †**0.961** | †**0.996** |
| $\text{Skyline}_{\text{CF}}$ | †**0.923** | †**0.938** | †**0.923** | †**0.932** | †**0.844** | †**0.917** | †**0.993** | †**0.912** | †**0.825** | †**0.961** | †**0.996** |

random split, the number of training and test users are more balanced so we do not observe this behavior in such a pronounced way, as in this case the users belonging to the Q4 group obtain better results.

### 4.5.6 Analyzing item attributes

For the item attributes, we work with a main feature and a secondary one: directors and genres in Movielens1M and level 3 and level 1 categories in FS (Tokyo). We use as similarity function $\text{sim}_F$ the Jaccard coefficient between the features of each pair of items; we set the penalization weight $\alpha$ as 0.8 and 0.6 for the main and secondary feature. When using both features at the same time, we first compute the similarity using the main feature, if there is no matching we use the secondary feature, and if there is no matching then, we assume the items have a zero similarity. Additionally,

**Table 4.22:** Performance results on Movielens1M dataset with different item attributes. Temporal system split (80% ancient ratings to train, rest to test). $\tau = 0$, represents the pure metric and $\tau_m$, $\tau_s$ and $\tau_{ms}$ represent the metric with the main, the secondary and both the main and the secondary features respectively. Same configuration as in Table 4.5.

| | **nDCG** | | | | **P** | | | |
|---|---|---|---|---|---|---|---|---|
| **Family** | $\tau = 0$ | $\tau_m$ | $\tau_s$ | $\tau_{ms}$ | $\tau = 0$ | $\tau_m$ | $\tau_s$ | $\tau_{ms}$ |
| Rnd | 0.012 | 0.034 | 0.269 | 0.276 | 0.019 | 0.053 | 0.329 | 0.344 |
| $\text{Rnd}_{\text{CF}}$ | 0.008 | 0.023 | 0.251 | 0.255 | 0.015 | 0.040 | 0.308 | 0.320 |
| Pop | **0.221** | **0.244** | 0.361 | 0.372 | **0.281** | **0.325** | **0.461** | **0.484** |
| $\text{Pop}_{\text{CF}}$ | 0.161 | 0.189 | 0.308 | 0.322 | 0.210 | 0.259 | 0.396 | 0.421 |
| UBCB | 0.195 | 0.221 | 0.356 | 0.366 | 0.254 | 0.297 | 0.450 | 0.471 |
| IB | 0.177 | 0.206 | 0.322 | 0.337 | 0.234 | 0.281 | 0.417 | 0.442 |
| UB | 0.195 | 0.224 | 0.347 | 0.360 | 0.248 | 0.297 | 0.438 | 0.463 |
| HKV | 0.202 | 0.230 | **0.364** | **0.375** | 0.257 | 0.302 | 0.454 | 0.476 |
| BPRMF | 0.172 | 0.201 | 0.334 | 0.347 | 0.231 | 0.279 | 0.429 | 0.454 |
| TD | 0.194 | 0.223 | 0.347 | 0.361 | 0.248 | 0.296 | 0.439 | 0.463 |
| MC | 0.134 | 0.170 | 0.312 | 0.327 | 0.177 | 0.234 | 0.397 | 0.424 |
| FPMC | 0.159 | 0.181 | 0.314 | 0.325 | 0.212 | 0.248 | 0.407 | 0.426 |
| Fossil | 0.170 | 0.195 | 0.331 | 0.342 | 0.227 | 0.270 | 0.428 | 0.450 |
| Caser | 0.136 | 0.166 | 0.309 | 0.321 | 0.192 | 0.239 | 0.401 | 0.426 |
| Skyline | †**1.000** | †**1.000** | †**1.000** | †**1.000** | †**0.943** | †**0.943** | †**0.943** | †**0.943** |
| $\text{Skyline}_{\text{CF}}$ | 0.999 | 0.999 | 0.999 | 0.999 | 0.911 | 0.911 | 0.911 | 0.911 |

we need to specify the $\tau$ function used to map the maximum similarity values into relevance weights; for this work we use the following simple mapping, in the future we aim to study how to better define such correspondence: $\tau(s) = 0.25$ if $0 < s \leq 0.5$, $\tau(s) = 0.5$ if $0.5 < s \leq 0.75$, $\tau(s) = 0.75$ if $s > 0.75$, and $\tau(s) = 0$ otherwise. Abusing the notation, we shall use $\tau_m$, $\tau_s$, or $\tau_{m,s}$ when this function is applied to the output of the Jaccard similarity as explained before to the main, secondary, or combined features. Hence, in Tables 4.22 and 4.23 we show the results of the recommenders in the Movielens1M dataset for the system and the random split respectively. Analyzing the results we can observe that with the first feature the recommenders obtain results more similar to the pure metric (and at the same time lower than the ones reported with the second feature). This makes sense since in this dataset we have a total of 3,565 different directors while we only have 16 film genres. This causes the metrics using the second feature to obtain such high results. Still, it is interesting that in most cases the best recommender is the same in all configurations (it only changes in the temporal split with nDCG), showing us the potential of these types of metrics. However, we would like to warn about the behavior of the Rnd recommender, since when we use the secondary feature it performs much better than we would expect, being very close to some other personalized recommenders. This shows that we must be specially careful when selecting the features to consider relevant items since although somewhat more specific features such as the directors in this case, could help us to better analyze the differences of the recommenders, if we use more generic features, the results can be counterproductive.

**Table 4.23:** Performance results on Movielens1M dataset with different item attributes. Random system split (80% ratings to train, rest to test, randomdly). Same configuration as in Table 4.22.

| | **nDCG** | | | | **P** | | | |
|---|---|---|---|---|---|---|---|---|
| **Family** | $\tau = 0$ | $\tau_m$ | $\tau_s$ | $\tau_{ms}$ | $\tau = 0$ | $\tau_m$ | $\tau_s$ | $\tau_{ms}$ |
| Rnd | 0.004 | 0.013 | 0.217 | 0.220 | 0.006 | 0.019 | 0.257 | 0.263 |
| $\text{Rnd}_{\text{CF}}$ | 0.004 | 0.013 | 0.218 | 0.222 | 0.006 | 0.020 | 0.258 | 0.265 |
| Pop | 0.141 | 0.165 | 0.288 | 0.298 | 0.168 | 0.206 | 0.342 | 0.359 |
| $\text{Pop}_{\text{CF}}$ | 0.141 | 0.165 | 0.288 | 0.298 | 0.168 | 0.206 | 0.342 | 0.359 |
| UBCB | 0.249 | 0.267 | 0.406 | 0.414 | 0.281 | 0.309 | 0.465 | 0.479 |
| IB | 0.241 | 0.261 | 0.381 | 0.390 | 0.287 | 0.315 | 0.454 | 0.468 |
| UB | 0.300 | 0.319 | 0.435 | 0.444 | 0.330 | 0.359 | 0.491 | 0.505 |
| HKV | **0.316** | **0.336** | **0.457** | **0.466** | **0.350** | **0.379** | **0.516** | **0.530** |
| BPRMF | 0.248 | 0.268 | 0.401 | 0.411 | 0.295 | 0.323 | 0.476 | 0.490 |
| Skyline | †**1.000** | †**1.000** | †**1.000** | †**1.000** | †**0.914** | †**0.914** | †**0.914** | †**0.914** |
| $\text{Skyline}_{\text{CF}}$ | †**1.000** | †**1.000** | †**1.000** | †**1.000** | †**0.914** | †**0.914** | †**0.914** | †**0.914** |

Let us analyze the effect of these metrics in the FS (Tokyo) dataset. In Tables 4.24 and 4.25 we show the results on this dataset for the temporal system and random system split respectively. Regarding these results, we can observe how the behavior of the Rnd recommender is similar to the reported in the Movielens1M dataset (with the main feature improving the performance slightly but with the secondary feature improving the performance substantially). However, there is a fundamental difference between the features. In this case, the results obtained using the main feature are normally superior to the ones obtained using the second feature. The reasons for this may be in the number of different features, since although there are not many differences with respect to the size of the secondary features (9 in this case versus 15 in Movielens1M), but 429 in the case of the main feature (versus 3,565 in Movielens1M). This makes the main feature of FS (Tokyo) more generic than in the case of Movielens1M, and thus explaining the greater increase in the results in this dataset. In addition, it should be noted that in this case, the main features are included in the secondary features. That is, if a primary feature is for example “Italian Restaurant”, its associated secondary feature would be “Food”. On the other hand, in this dataset we can see more changes in the ranking of the recommenders when we use the item attributes, while in Movielens1M they were much more stable. This may be due to two main reasons. First, it is more challenging to make relevant recommendations in this dataset because of its sparsity and secondly, when evaluating using the item features we may end up having greater discrepancies because there are a fewer number of different features than in the case of Movielens1M, specially in the main feature. Although we want to emphasize again to be careful with this type of metrics since we believe that a deeper analysis on the penalties to apply when using these features should be done, we also believe that this type of metrics can be especially useful in some circumstances, e.g., in very sparse scenarios or as tie-breaker when we have recommenders obtaining a very similar performance in ranking-based accuracy metrics.

**Table 4.24:** Performance results on FS (Tokyo) dataset with different item attributes. Temporal system split (80% ancient ratings to train, rest to test). Same configuration as in Table 4.22.

| | **nDCG** | | | | **P** | | | |
|---|---|---|---|---|---|---|---|---|
| **Family** | $\tau = 0$ | $\tau_m$ | $\tau_s$ | $\tau_{ms}$ | $\tau = 0$ | $\tau_m$ | $\tau_s$ | $\tau_{ms}$ |
| Rnd | 0.000 | 0.090 | 0.279 | 0.306 | 0.000 | 0.085 | 0.246 | 0.271 |
| $\text{Rnd}_{\text{CF}}$ | 0.000 | 0.094 | 0.282 | 0.310 | 0.000 | 0.088 | 0.250 | 0.276 |
| Pop | **0.078** | 0.386 | 0.366 | 0.469 | **0.065** | 0.307 | 0.302 | 0.383 |
| $\text{Pop}_{\text{CF}}$ | 0.070 | 0.386 | 0.362 | 0.468 | 0.059 | 0.307 | 0.301 | 0.384 |
| UBCB | 0.072 | **0.415** | 0.380 | **0.494** | 0.060 | **0.342** | 0.320 | 0.413 |
| IB | 0.066 | 0.373 | **0.388** | 0.489 | 0.054 | 0.310 | **0.330** | **0.414** |
| UB | 0.076 | 0.401 | 0.366 | 0.473 | 0.064 | 0.325 | 0.305 | 0.390 |
| HKV | 0.070 | 0.381 | 0.375 | 0.476 | 0.059 | 0.318 | 0.317 | 0.401 |
| BPRMF | 0.071 | 0.382 | 0.363 | 0.467 | 0.060 | 0.304 | 0.302 | 0.384 |
| TD | 0.077 | 0.400 | 0.368 | 0.475 | 0.064 | 0.324 | 0.306 | 0.392 |
| MC | 0.057 | 0.381 | 0.365 | 0.473 | 0.048 | 0.315 | 0.307 | 0.395 |
| FPMC | 0.047 | 0.368 | 0.366 | 0.473 | 0.038 | 0.294 | 0.309 | 0.394 |
| Fossil | 0.056 | 0.338 | 0.367 | 0.460 | 0.047 | 0.290 | 0.318 | 0.398 |
| Caser | 0.054 | 0.338 | 0.370 | 0.463 | 0.045 | 0.276 | 0.313 | 0.388 |
| Skyline | †**0.933** | †**0.933** | †**0.933** | †**0.933** | 0.775 | 0.775 | 0.775 | 0.775 |
| $\text{Skyline}_{\text{CF}}$ | 0.928 | 0.928 | 0.928 | 0.928 | †**0.777** | †**0.777** | †**0.777** | †**0.777** |

## 4.6 Discussion

In this chapter we have addressed RG2: *Study evaluation metrics for classical recommendation to adapt and integrate additional dimensions beyond relevance.* We have presented several metrics that incorporate both temporal and sequential information to evaluate the recommendations produced by the model. In addition, we have also defined other metrics by exploring other sources of information, such as low ratings (usually ignored in recommendation) and attributes of both users and items.

With the metrics developed, we have been able to analyze in more detail the recommendations produced by the different algorithms, detecting some interesting behaviors. Thus, with the time-aware novelty metrics we have found a correlation between relevant and temporal novel items. Using the negative ratings of the items, we have observed how some traditional personalized models, while producing relevant recommendations for users, they sometimes also produce recommendations that displease the user. Finally, with the user and item attributes we detected some biases in the recommendations produced, since for the users belonging to a less numerous group, it is more likely to obtain worse recommendations. In the case of item attributes, we have shown that they might be useful in domains where the sparsity is too high, although the selection of these attributes is critical. If we apply relatively low penalties on the features similarities or reduced number of attributes are used (favoring more hits between the attributes of the relevant and the recommended items), the results might be too optimistic.

However, we have also observed that it is very difficult for an algorithm to obtain the best results in all metrics. While we would argue that the accuracy in terms of relevance

**Table 4.25:** Performance results on FS (Tokyo) dataset with different item attributes. Random system split (80% ratings to train, rest to test, randomdly). Same configuration as in Table 4.22.

| | nDCG | | | | P | | | |
|---|---|---|---|---|---|---|---|---|
| **Family** | $\tau = 0$ | $\tau_m$ | $\tau_s$ | $\tau_{ms}$ | $\tau = 0$ | $\tau_m$ | $\tau_s$ | $\tau_{ms}$ |
| Rnd | 0.000 | 0.086 | 0.269 | 0.295 | 0.000 | 0.079 | 0.234 | 0.257 |
| $\text{Rnd}_{\text{CF}}$ | 0.000 | 0.087 | 0.270 | 0.297 | 0.000 | 0.081 | 0.236 | 0.260 |
| Pop | 0.052 | 0.389 | 0.359 | 0.472 | 0.041 | 0.302 | 0.289 | 0.375 |
| $\text{Pop}_{\text{CF}}$ | 0.052 | 0.390 | 0.360 | 0.472 | 0.041 | 0.302 | 0.289 | 0.376 |
| UBCB | 0.057 | **0.428** | 0.383 | **0.505** | 0.044 | **0.341** | 0.310 | 0.408 |
| IB | 0.056 | 0.383 | **0.389** | 0.497 | 0.044 | 0.312 | **0.323** | **0.411** |
| UB | **0.064** | 0.410 | 0.372 | 0.485 | 0.050 | 0.323 | 0.299 | 0.388 |
| HKV | **0.064** | 0.399 | 0.387 | 0.496 | **0.051** | 0.326 | 0.319 | 0.407 |
| BPRMF | 0.053 | 0.385 | 0.367 | 0.477 | 0.042 | 0.300 | 0.297 | 0.383 |
| Skyline | †**0.923** | †**0.923** | †**0.923** | †**0.923** | 0.736 | 0.736 | 0.736 | 0.736 |
| $\text{Skyline}_{\text{CF}}$ | †**0.923** | †**0.923** | †**0.923** | †**0.923** | †**0.738** | †**0.738** | †**0.738** | †**0.738** |

should be one of the most important criteria for choosing the best recommenders, it is also necessary to analyze other dimensions (novelty, diversity, etc.) as in many cases these recommendations might be very similar to each other (sometimes users are interested in expanding their tastes and finding new items). We argue that with the developed metrics, we can further analyze what kind of recommendations each algorithm produces and detect possible biases in the datasets.

On the other hand, we have observed that the results of the recommenders change significantly depending on the evaluation methodology followed to make the recommendations (parameter tuning and type of split performed). Thus, we noticed that the results obtained with a system random split tend to be higher in terms of accuracy than those obtained using a system temporal partition. Moreover, we have also observed that in the case of temporal splits we might be losing user coverage, showing that user interactions with the system are not homogeneous (with a large number of users who are active for only a short time in the system). However, we would like to point out once again that when evaluating algorithms in an offline environment, they should be performed in the most realistic configuration possible. Hence, we argue that the evaluation should be conducted using temporal splits, since past and future interactions should not be mixed when making recommendations.

# 5

# Sequence integration in $k$-NN Recommender Systems

In the previous chapters, we have shown how to incorporate the temporal and sequential contexts in recommendation metrics. We have also explained some state-of-the-art proposals based on Neural Networks and Markov Chains (usually combined with other models such as matrix factorization) that capture this type of information. In this chapter, as a novel contribution, we will show how these contexts can be incorporated into neighborhood-based algorithms, since they are techniques that are easier to implement efficiently and to understand than those mentioned above. First, in Section 5.1 we define a sequential similarity metric based on the Longest Common Subsequence (LCS) algorithm; then, in Section 5.2 we show a reformulation of a classic user-based neighborhood algorithm by bringing ideas from ranking fusion techniques. In Section 5.3 we show the experiments of our proposals in two different datasets under two temporal evaluation methodologies and finally, in Section 5.4 we make a discussion about our sequential $k$-NN proposed algorithms.

The content of this chapter has been partially published in the following articles:

- **Pablo Sánchez** and Alejandro Bellogín. Time and sequence awareness in similarity metrics for recommendation. *Information Processing Management*, 57(3):102228, 2020.

- **Pablo Sánchez** and Alejandro Bellogín. Building user profiles based on sequences for content and collaborative filtering. *Information Processing Management*, 56(1):192-211, 2019.

- Alejandro Bellogín and **Pablo Sánchez**. Collaborative filtering based on subsequence matching: A new approach. *Information Sciences*, 418:432-446, 2017.

**Algorithm 3** Longest Common Subsequence for Recommender Systems

```
1: procedure LCS_RecSys($u, v, f, \delta$)    ▷ The LCS of users $u$ and $v$ applying transformation $f$
2:     $(x, y) \leftarrow (f(u), f(v))$    ▷ String $x$ contains $m$ symbols
3:     $L[0 \cdots m, 0 \cdots n] \leftarrow 0$
4:     for $i \leftarrow 1, m$ do
5:         for $j \leftarrow 1, n$ do
6:             if match($x_i, y_j, \delta$) then
7:                 $L[i, j] \leftarrow L[i-1, j-1] + 1$    ▷ There is a $\delta$-matching
8:             else
9:                 $L[i, j] \leftarrow \max(L[i, j-1], L[i-1, j])$
10:            end if
11:        end for
12:    end for
13:    return $L[m, n]$
14: end procedure
```

## 5.1 Sequential similarity metric

In Section 2.2.2 we presented some of the most popular similarities that are applied to the neighborhood-based Recommender Systems: cosine similarity (Equation 2.9), Pearson correlation (Equation 2.10), and Jaccard index (Equation 2.11). However, as we observe in those equations, these similarities work with the interactions between users and items as if they were time-agnostic sets. Even though in Section 2.3 we showed some examples of similarities incorporating temporal information (Campos et al., 2014, Ding and Li, 2005), we believe that more grounded functions that exploit this type of information could be defined, in particular, sequential similarities to be used in neighborhood-based algorithms. In fact, there are several sequential metrics that can be adapted to be used for this purpose (e.g. Levenshtein, Jaro-Winkler, Hamming, etc.). We believe the Longest Common Subsequence (LCS), already introduced in Section 4.4 to define a sequential evaluation metric, has a great potential to be used in a neighborhood-based algorithm. To do so, it is necessary to define a method to represent the consumed items into sequences and also a function that identifies when two characters are the same (in order to detect a match in those sequences).

In Algorithm 3 we present the adaptation of the LCS algorithm (shown in Algorithm 1) to be used as a similarity metric between two users. As the reader may observe, there are two major changes with respect to the original formulation shown in Algorithm 1, where we used this algorithm to compare the recommended items to the target user with respect to those items consumed in the test set. Firstly, the algorithm receives a function $f$ that transforms both users into sequences and a $\delta$-matching that allows us to define when two symbols of the alphabet are considered equal. This second modification lets us to be more flexible when comparing the elements of the sequences (if the difference between the elements is lower that $\delta$ then the elements are considered equal).

The transformation function $f$ deserves special attention as depending on the information we use to represent the users, we may obtain different sequences and hence different value in the LCS metric. Assuming that a user can be modeled as a set of interactions (items and ratings), we describe the following steps to generate a sequence in a generic way:

1. **Extend the associated information about the items interacted by the user.** Formally, we need a function that returns a set of elements associated to every item. That is, a function of the form: $e : \mathcal{I} \times \mathcal{R} \to \mathcal{I} \times \mathcal{T}^k$, where $\mathcal{R}$ denotes the set of ratings, $k > 0$ denotes the number of those elements that function $e$ is able to associate with every item, and $\mathcal{T}$ represents those elements, modeled in general as tuples (item and rating). Note that a pure CF approach is derived from this formulation if we use the identity function in this step: $e_{ir}(i, r) = (i, \{i, r\})$. Nevertheless, content-based methods would exploit the feature space so that every item is linked to their corresponding features: $e_{Ar}(i, r) = (i, \{A_j(i), r\}_j)$, where the feature space $A$ could be genres, directors, or actors in the movie domain or text features in news recommendation.

2. **Represent the tuples created as interpretable symbols by the LCS algorithm.** Here, we propose to use $t : \mathcal{I} \times \mathcal{T}^k \to \mathcal{I} \times \mathbb{Z}^k$, where a proper transformation between $\mathcal{T}$ and $\mathbb{Z}$ (the set of integer numbers) is required. The reason why we use the set of integer numbers instead of strings or other space is that they are computationally more affordable. As a simple example, associated to the function $e_{ir}$ we define the function $t_{ir}(i, r) = 10 \cdot \text{id}(i) + r$ in such a way that it is also possible to recover the original elements of the tuple (the item identifier and the corresponding rating of the interaction) given its output (bijective function). The factor of 10 that multiplies the id allows us to separate the item id and the rating while combining them into a single symbol; obviously, this factor depends on the rating interval. Thus, if ratings are, for example, between 1 and 20, the transformation function should be $t'_{ir}(i, r) = 100 \cdot \text{id}(i) + r$ in order to make that recovery possible – i.e., to have an actual bijection.

3. **Arrange the symbols into a sequence.** In string matching, the ordering of the sequence is important, and it is an aspect that the LCS algorithm is able to exploit. As a first approach, this step can be simplified to just sort the items in the sequence according to their item id, although it is worth noting that any other global ordering of the items would be equivalent to this one, for example, by item popularity. However, in order to fully exploit the temporal information, it would be more useful to sort the user sequences following a temporal order. This allows the LCS-based similarity to be even more flexible, as we could give more importance to the matchings that have occurred more recently. The LCS-based similarities used in this chapter will order the sequences in a temporal manner. In particular, the sequence arranging function proposed will take several pairs

of items and tuples generated as described before and will output a sequence of symbols, prepared to be processed by the LCS algorithm.

Thus, the sequence generation function $f$ could be seen as a composition of the three functions presented at each step: $f = s \circ t \circ e$.

To clarify the process of sequence generation explained before, let us present an example considering a dataset based on movies, which usually have some content information associated like actors, directors, or genres. We have the movie *Star Wars IV*, with id 1, and a user $u$ who has rated it with a 5 as rating value. If we use function $e_{gr}$ to extend this information based on genres – i.e., $A = G$ and then $e_{gr}(i, r) = (i, \{G_j(i), r\}_j)$ – we could find that item 1 has two genres: Sci-Fi (id 7) and Adventure (id 10). According to the definition of $e_{gr}$, this function leads to the tuple $(1, \{\{\text{Sci-Fi}, 5\}, \{\text{Adventure}, 5\}\})$. After that, we would represent these tuples as useful symbols for the sequence similarity function (LCS, in our case) using a reasonable $t_{gr}$ function. By taking a similar one to $t_{ir}$, we could transform each genre into its id and use that value in combination with its associated rating, creating the tuple $(1, \{75, 105\})$. Finally, to generate the sequence corresponding to this user, we simply take the tuples associated to the only item this user has rated: $(75, 105)$. However, if the user had also rated *The Godfather* after *Star Wars IV* (with a rating value of 4 and whose id is 15), then the output would be slightly different. This movie has Drama (id 2) and Crime (id 6) as genres. The tuple related to this second movie would be (following the same steps as before, i.e., using $t_{gr} \circ e_{gr}$): $(15, \{24, 64\})$. The final step would produce the sequence $(75, 105, 24, 64)$, since the timestamp of *Star Wars* is lower than the one for *The Godfather*. Note that if $e_{ir}$ and $t_{ir}$ functions are used, that is, pure collaborative filtering information is being exploited, then each item will only generate one tuple and the final generated sequence will be shorter: $(15, 154)$.

Nevertheless, as we can derive from Algorithm 3, the LCS algorithm obtains values in the $[0, \min(|f(u)|, |f(v)|)]$ interval. However, classic similarity metrics in recommendation are usually normalized in the range $[-1, 1]$ or $[0, 1]$. In line with the same rationale, we propose to use the following repetition normalizations for our LCS-based similarity metric:

$$\mathrm{sim}_1^{f,\delta}(u, v) = \text{LCS\_Recsys}(u, v, f, \delta) \tag{5.1}$$

$$\mathrm{sim}_2^{f,\delta}(u, v) = \frac{\mathrm{sim}_1^{f,\delta}(u, v)^2}{|f(u)| \cdot |f(v)|} \tag{5.2}$$

$$\mathrm{sim}_3^{f,\delta}(u, v) = \frac{2 \cdot \mathrm{sim}_1^{f,\delta}(u, v)}{|f(u)| + |f(v)|} \tag{5.3}$$

$$\mathrm{sim}_4^{f,\delta}(u, v) = \frac{\mathrm{sim}_1^{f,\delta}(u, v)}{\max(|f(u)|, |f(v)|)} \tag{5.4}$$

$$\mathrm{sim}_5^{f,\delta}(u, v) = \frac{\mathrm{sim}_1^{f,\delta}(u, v)}{\min(|f(u)|, |f(v)|)} \tag{5.5}$$

As these normalizations include the lengths of the sequences in the denominator as penalization, they tend to favor longer subsequences found inside short sequences, while producing values in the $[0, 1]$. These normalization functions were also used in de la Rosa et al. (2005) in the context of data cleaning and processing by comparing groups of string identifiers.

## 5.2 Sequential user-based $k$-NN recommenders

### 5.2.1 Background

As mentioned in the previous section, the traditional definition of user-based $k$-NN recommenders does not incorporate nor temporal nor sequential information. Both types of information have proven to be of vital importance to model and understand the evolution of the user behavior and have been applied in a large number of algorithms. Although in the previous section we mentioned how to incorporate sequentiality in a similarity metric, we have not yet discussed in depth how to add these dimensions in the formulation of neighbor-based algorithms.

Let us now focus on the TD approaches shown in Equations 2.19 and 2.20 since they have demonstrated good performance in the past. Even if these methods incorporate temporal information in an intuitive way, they also have some limitations. Firstly, it is not clear where the timestamp used in that formulation is captured, which affects the reproducibility of the model and, secondly, both formulations are oriented to a rating-prediction task where the objective is to minimize the error of the recommender. However, as we have explained in Section 2.2, Recommender Systems community is currently focused in proposing approaches oriented to the ranking task, that is, providing a list of items to the user, not a rating prediction to a specific item (Steck, 2013).

In order to address these concerns, we propose a reformulation of this time-aware similarity metric to be used in a user-based $k$-NN recommender. Thus, bringing ideas from Equations 2.19 and 2.20, we propose the following formulation of a $k$-NN using a time-aware user similarity:

$$\hat{r}_{ui} = \sum_{v \in N_u} sim(u, v) \cdot r_{vi} \cdot f_c(t_u^L, t_{v,i}) \tag{5.6}$$

where $t_u^L$ is the timestamp of the last rating of user $u$ in the training set, $t_{v,i}$ is the time when user $v$ rated item $i$, and $f_c(\cdot, \cdot)$ is our *soften* time decay function. This function, as the previous formulations, is based on an exponential function, which, by default, has the following form: $f_c(t_1, t_2) = e^{-\lambda \cdot \text{diff}(t_1, t_2)}$, where $\text{diff}(t_1, t_2)$ indicates the difference in the same time units as $\lambda$ (recall that $\lambda$ is the decay rate and is usually defined in days by default) between two timestamps (i.e., $t_1 - t_2$).

However, we decided to apply a soften value when $\text{diff}(t_1, t_2) < 0$ and $abs(\text{diff}(t_1, t_2)) > \frac{1}{\lambda} \cdot c$, where we force $\text{diff}(t_1, t_2) = -\frac{1}{\lambda} \cdot c$. This bounds the time decay factor to a value of $e^c$, to avoid large values when the neighbor rated the item further away in the future with respect to the last interaction of the target user. Hence, factor

$c$, allows us to control up to which point in the future ($c$ times the period $T_0 = \lambda^{-1}$) all the neighbor interactions will have the same weight. It is worth remembering that the similarity used between users is configurable and we could use both cosine, Pearson, or even an LCS-based similarity in which the sequences are ordered temporarily.

### 5.2.2 Our approach

Our main contribution in this chapter regarding the neighborhood-based recommender systems is to combine ranking fusion techniques used in the Information Retrieval area with neighborhood-based algorithms. Thus, bringing ideas from Aggregated Search (Dwork et al., 2001, Renda and Straccia, 2003), we will denote each neighbor as a *judge* (in Information Retrieval, these judges normally refer to different search engines) and will give a complete ordering of all the items to be ranked. We will refer as $\tau$ to the candidate list of each neighbor and the final fused ranking will be denoted as $\hat{\tau}$. This ranking aggregation process can be divided in two different steps: 1) normalization, where the scores of the ranks of $\tau$ are normalized into a common scale $w^{\tau}(i)$ for each item $i$, and 2) combination, where the normalized weights $w^{\tau}(i)$ are combined into a fused score for each item.

There are several methods for each of these stages. In this sense, Renda and Straccia (2003) conducted an in-depth review of the most prominent ones. Interestingly, by taking the identity normalizer for the scores ($w^{\tau}(i) = \tau(i)$) and the so-called CombSUM combiner (where the normalized weights are simply added for each item) with a preference weight for each ranking equals to the similarity between the neighbor and the target user, we obtain a linear combination of the normalized weights, which is equivalent to the user $k$-NN recommender formulation. In fact, when we take into account the ratings of the neighbors, the "score" of user $u$ to item $i$ using CombSUM and the identity normalizer produces the formulation in Equation 2.8. In this situation, each ranking $\tau$ is composed of the item-rating pairs rated by a particular neighbor, excluding, as a standard practice in the community, those items already rated by the target user in training. Further extensions and *ad-hoc* modifications could be made to these normalizers and combiners so that other formulations of this problem – such as mean-centering or Z-score normalization (Ning et al., 2015) – are obtained.

Once we have reformulated the problem of neighborhood-based recommendation as a ranking fusion technique, we now describe how we can incorporate temporal and/or sequential information in the process. Our main idea is that each neighbor will find her last common interaction with respect to the target user and will create a ranking of her candidate alternatives iterating around that item, taking into account the order (sequence) in which she rated each of those alternatives. We are aware that a major drawback of this approach is that it does not take into account the time at which the interactions occurred, so it could be that you end up recommending items from a neighbor who has not been active in the system for a long time. Nevertheless, we believe that this disadvantage can be easily mitigated by incorporating additional filters to eliminate this type of users. Note, however, that the sequential aspect is considered twice in this model: it is used to involve the target user (through the last common interaction) in setting the actual moment (context) of the recommendation and, at the same time, to exploit the temporal sequence (order) in which the neighbor interacted with the items.

Hence we now define our model called backward-forward (BF), as it first computes the last common interaction between the users and then makes use of different strategies to compute the order of the neighbors rankings. After that, a single ranking is generated by normalizing and combining all the rankings generated by the neighbors. For normalizing the scores, the most common strategies are the default normalization (Def), where the item value is the same as its score, the standard normalization (Std), that applies the min-max normalization to every item in the recommended list and the rank-sim normalization (Rks), in which the item score is inversely proportional to its position in the ranking. On the other hand, for the combination step the most extended approach, as mentioned earlier, is the CombSUM combiner that computes a weighted average of the normalized weights of each item using an additional preference weight for each ranking. Other approaches such as CombMIN, CombMAX, and CombMNZ (where the minimum, maximum, and the number of nonzero values is used in the combination step) have been proposed, but we will not use them in our approach.

Thus, the first step of the BF approach is computing the last common interaction between two users, that can be represented as follows:

$$n^*(u;v) = \max_k \left(i_k \in \mathcal{I}_u^t : i_k \in \mathcal{I}_v^t\right) \tag{5.7}$$

where $\mathcal{I}_u^t$ are the items rated by user $u$ ordered by timestamp in ascending order:

$$\mathcal{I}_u^t = \text{sort}\left(\mathcal{I}_u, t\right) = \left(i_k^t\right)_{k=1}^{|\mathcal{I}_u|}, \text{ with } t\left(i_k^t\right) < t\left(i_{k+1}^t\right) \tag{5.8}$$

It is important to note that the last common interaction represented in Equation 5.7 is not symmetrical, that is, $n^*(u;v) \neq n^*(v;u)$, since it looks for the preferences of the first user in those of the second user. Thus, we propose four different strategies to generate a list of candidate items from each neighbor using the last common interaction with respect to the target user: 1) take the $m$ items that have been rated after that common interaction ($L_m^+(v;u)$) (b) take the $m$ items that have been rated before that common interaction ($L_m^-(v;u)$) (c) take the $m$ items that have been rated before and after that last common interaction alternating them ($L_m^a(v;u)$) (d) take the $m$ items that have been rated after the last common interaction first and concatenate them the $m$ that have been rated before ($L_m^\pm(v;u)$). Formally:

$$\text{Let } \mathcal{I}^t(v;u) = \text{sort}\left(\mathcal{I}_v - \mathcal{I}_u, t\right)$$

$$L_m^+(v;u) = \left(i_k^t\right)_{n^*(v;u)}^{n^*(v;u)+m}, i_k^t \in \mathcal{I}^t(v;u) \tag{5.9}$$

$$L_m^-(v;u) = \left(i_k^t\right)_{n^*(v;u)-m}^{n^*(v;u)}, i_k^t \in \mathcal{I}^t(v;u) \tag{5.10}$$

$$L_m^a(v;u) = \left(a_k^t, b_k^t\right)_{k=1}^{\max(|L_m^+(v;u)|,|L_m^-(v;u)|)} \quad a_k^t \in L_m^+(v;u), b_k^t \in L_m^-(v;u) \tag{5.11}$$

$$L_m^\pm(v;u) = \left(L_m^+(v;u), L_m^-(v;u)\right) \tag{5.12}$$

Therefore, we generate a list $L(v;u)$ for each neighbor with all the candidate items from that neighbor. This candidate list will be later normalized and combined, to produce a single ranking, containing the recommendations for the target user $u$. Since these items are related to the last interactions between users $u$ and $v$, they can be interpreted as "the recent difference" in interaction history between these users. It is important to note that any similarity can be used in such scheme when obtaining the last common interaction between the users or generating the lists with candidate items, since these two steps do not depend on the user similarity, although the neighbors might be different or the weights used at the combination step if the similarity changes.

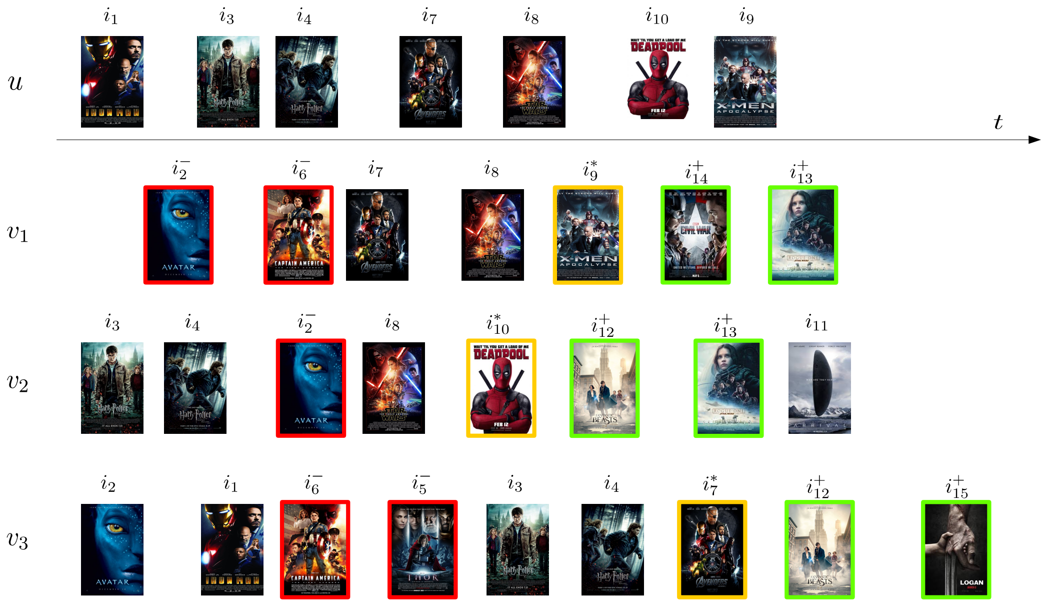
In summary, when using this formalization, we obtain a neighborhood-based recommendation model equivalent to classical formulations that can further incorporate the temporal information under different models.

### 5.2.3 Toy example

Finally, let us illustrate the whole process with an example shown in Figure 5.1 using the movie domain. For the sake of simplicity, we do not include the user's rating, so the reader should assume that all sequences correspond to items that the user has equally liked. In the movie domain, the temporal component is usually determining, since newer movies tend to be consumed more often than older ones. In our example, user $u$ is the target user, and we represent three neighbors $v_1$, $v_2$, and $v_3$, where $v_1$ and $v_3$ have three items in common whereas $v_2$ shares four items with the target user. According to these interactions, the candidate items generated with respect to the different strategies presented before (limited to size 2) will be (considering that $n^*(v_1;u) = i_9, n^*(v_2;u) = i_{10}, n^*(v_3;u) = i_7$):

$$L_2^+(v_1;u) = (i_{14}, i_{13}), L_2^-(v_1;u) = (i_6, i_2), L_{1,1}^{\pm}(v_1;u) = (i_{14}, i_6), L_{1,1}^a(v_1;u) = (i_{14}, i_6)$$
$$L_2^+(v_2;u) = (i_{12}, i_{13}), L_2^-(v_2;u) = (i_2), L_{1,1}^{\pm}(v_2;u) = (i_{12}, i_2), L_{1,1}^a(v_2;u) = (i_{12}, i_2)$$
$$L_2^+(v_3;u) = (i_{12}, i_{15}), L_2^-(v_3;u) = (i_5, i_6), L_{1,1}^{\pm}(v_3;u) = (i_{12}, i_5), L_{1,1}^a(v_3;u) = (i_{12}, i_5)$$

Note that in this case, as we are selecting only two items, $L_m^a$ and $L_m^{\pm}$ are equivalent. Suppose now that items $i_{12}$, $i_{13}$, and $i_{14}$ are in the test set (as mentioned before, newer items are more likely to be chosen by user $u$). A standard neighborhood-based recommender (that does not consider any temporal aspect of the data) would probably recommend item $i_2$ whereas in our approach, we are favoring more recent items, like the movie $i_{12}$. In fact, $i_2$ only appears in our method once for strategy $L^{\pm}$ and twice for $L^-$. Moreover, we believe that moving forward from the last common interaction is more useful in terms of recommendation performance – especially for novelty purposes – than moving backwards, this is evidenced by the strategies $L^+$ and $L^{\pm}$ that recommend $i_{13}$ and $i_{14}$; because of this, we ignore the $L^{\pm}$ strategy, since in some preliminary experiments we found it was equivalent to $L^+$, due to very sparse data.



$u$
$i_1$ $i_3$ $i_4$ $i_7$ $i_8$ $i_{10}$ $i_9$
$t$
$v_1$
$i_2^-$ $i_6^-$ $i_7$ $i_8$ $i_9^*$ $i_{14}^+$ $i_{13}^+$
$v_2$
$i_3$ $i_4$ $i_2^-$ $i_8$ $i_{10}^*$ $i_{12}^+$ $i_{13}^+$ $i_{11}$
$v_3$
$i_2$ $i_1$ $i_6^-$ $i_5^-$ $i_3$ $i_4$ $i_7^*$ $i_{12}^+$ $i_{15}^+$

**Figure 5.1:** Example of user interactions in the movie domain. Items denoted with an asterisk ($*$) and a yellow border correspond to the last common interaction between $u$ and each neighbor, those with a $-$ symbol as superscript and a red border are included in $L_2^-(v;u)$, whereas those with a $+$ as superscript and a green border in $L_2^+(v;u)$.

## 5.3 Experiments

### 5.3.1 Datasets

In this chapter, we experiment with two datasets from different domains where all the provided ratings have been timestamped: Foursquare and MovieTweetings. The Foursquare dataset, as we have explained in previous chapters, comes from the tourism domain (the items in this case are touristic venues or Point-of-Interest, as they are usually denoted). This dataset was obtained from He and McAuley (2016), the authors that provided the source code for MC, FPMC and Fossil. We decided to use this dataset here because we wanted to test both our models and the state-of-the-art algorithms using the dataset where the Fossil algorithm was proposed, as the author claimed that their model performed very well on this dataset. The Foursquare dataset, as explained before, contains implicit feedback and we only have information about whether a user visited a specific venue. MovieTweetings, on the contrary is a movie dataset where IMDb ratings have been collected from Twitter (Dooms et al., 2016). For MovieTweetings, the users have indicated their tastes with different values (0 to 10), low values representing bad opinions about the items. Besides, MovieTweetings is a dataset that is constantly updated, we decided to take a specific snapshot of approximately 600K ratings.

For both datasets, we have filtered the original data by removing all the repeated preferences, taking into account only the newest preference when a user rated the same item more than once. This processing only had an effect on Foursquare, since MovieTweetings does not include repetitions. Additionally, we performed a $k$-core in both datasets, this means that we removed every user or item that did not have at least $k$ preferences; in this case, we used $k = 5$ in MovieTweetings and Foursquare datasets. The final statistics of the processed datasets are shown in Table 5.1.

We would like to draw the attention on the difficulty of obtaining datasets with realistic temporal information. In some preliminary analyses, we discarded Movielens (one of the most popular datasets in the area but where some researchers alerted about its non-realistic timestamps (Harper and Konstan, 2016)) and Amazon reviews datasets[1], because too many items were consumed at the same time (in the exact same second) by a user, which does not make sense for either performing a temporal evaluation or running a time-aware recommendation algorithm. The datasets used in this chapter are not perfect either (see column 'Unique times' in Table 5.1 which, ideally, should be close to the number of interactions in every case), but are the best ones we could find with a large number of users and items and a decent number of interactions.

### 5.3.2 Evaluation methodology

Since we deal with temporal information and its effect on Recommender Systems, we should use time-aware evaluation methodologies such as the ones introduced in

[1] Available here, `http://jmcauley.ucsd.edu/data/amazon/`.

**Table 5.1:** Statistics from the datasets used in the experiments.

| Dataset | Users | Items | Ratings | Density | Scale | Unique times | Time interval |
|---|---|---|---|---|---|---|---|
| Foursquare | 16k | 3k | 105k | 0.205% | 1 | 102k | Dec 2011 - Apr 2012 |
| MovieTweetings | 15k | 8k | 519k | 0.399% | 0-10 | 517k | Feb 2013 - Apr 2017 |

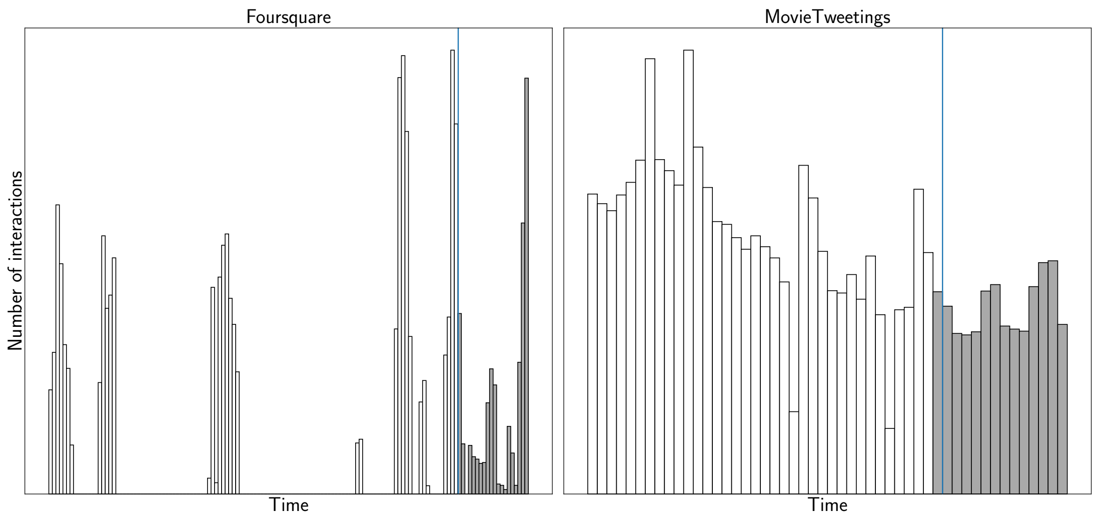


**Figure 5.2:** Distribution of ratings in the datasets for the system split. Foursquare (left), and MovieTweetings (right). The blue line represents the splits of the training and test sets

Section 4.1. In this chapter, we use a time-dependent rating order (the timestamps of the test split for each user occur after those of the training split) in two evaluation methodologies: one with a user-centered base set and a fixed size condition (the last 2 actions of each user with at least 6 actions are included in the test split) and another with a community-centered base set and a proportion-based size condition (the same timestamp is used for all the users, in such a way that we retain the data corresponding to the 20% of the most recent rating times for testing, and the rest for training). Using the notation introduced in Chapter 2 (see Section 2.4.2), we refer to the first configuration as *temporal per user* and the second as *temporal system*. There are obvious differences between these two evaluation methodologies: whereas in *temporal system* the test set is always (for every user) after the training set, in *temporal per user* this may not be the case; besides, (almost) every user is included in the test set of *temporal per user* but this is not the case for *temporal system*. Because of these features, the *temporal system* methodology represents better a real environment, where there are some users that may not be active at some point, whereas with the *temporal per user* evaluation we can analyze the recommendations for all the users, not only the most active (or active in the last period when the dataset was collected) ones.

Furthermore, as in Chapter 4, we report the results obtained following the

**Table 5.2:** Parameters used with the evaluated recommenders. VecC and SetJ stand for VectorCosine and SetJaccard.

| **Recommender** | **Parameters** |
|---|---|
| Rnd | None |
| Pop | None |
| IB | $k$=$\{40, 60, 80, 100, 120\}$, sim=$\{$VecC, SetJ $\}$ |
| UB | $k$=$\{40, 60, 80, 100, 120\}$, sim=$\{$VecC, SetJ $\}$ |
| HKV | Factors=$\{10, 50, 100\}$, $\alpha$=$\{0.1, 1, 10\}$, $\lambda$=$\{0.1, 1, 10\}$ |
| BPRMF | Factors=$\{10, 50, 100\}$, Iter=50, LearnRate=0.05, RegJ=RegU/10, BiasReg=$\{0, 0.5, 1\}$, RegU=RegI=$\{0.0025, 0.001, 0.005, 0.01, 0.1\}$ |
| TD | $k$=$\{40, 60, 80, 100, 120\}$, sim=$\{$VecC, SetJ $\}$, $\lambda$=$\{0.05, 0.1\}$ |
| BFUB | $k$=$\{40, 60, 80, 100, 120\}$, sim=$\{$VecC, SetJ $\}$, $L_m^- = L_m^+$=$\{5, 10\}$, norm=$\{$ Def, Std, Rks $\}$, Wgt = $\{T, F\}$ |
| BFsUB | $k$=$\{40, 60, 80, 100, 120\}$, sim=$\{\text{sim}_1, \text{sim}_2, \text{sim}_3\}$, $L_m^- = L_m^+$=$\{5, 10\}$, norm=$\{$ Def, Std, Rks $\}$, Wgt = $\{T, F\}$ |
| MC | $K$=$\{5, 10, 20, 50\}$, $\lambda$=$\{0.1, 0.2\}$ |
| FPMC | $K$=$\{5, 10, 20, 50\}$, $\lambda$=$\{0.1, 0.2\}$ |
| Fossil | $K$=$\{5, 10, 20, 50\}$, $\lambda$=$\{0.1, 0.2\}$, $L$=$\{1, 2, 3\}$ |
| Caser | $L$=2, $T$=$\{1, 2\}$, Iter=30, bath=512, LearnRate=0.003, negSamples=3, d=$\{10, 50\}$, v=4, h=$\{4, 16\}$, drop=0.5 |
| Skyline | None |

*TrainItems* strategy to select the candidate items to be ranked by each algorithm; that is, a ranking is generated for each user by predicting a score for every item that has at least one interaction in the training set; as it is standard, we remove those items already rated in training by the user from each candidate list (in a user basis). We then compute classic ranking-based accuracy metrics (see Chapter 2) considering as relevant every item rated at least with a 9 in the test split for MovieTweetings, whereas for Foursquare every item in the user's test set is considered as relevant. We present Precision (P), Recall (R) and nDCG for measuring the accuracy of the recommenders, EPC for novelty, MIN for our time-aware novelty metric presented in Section 4.5.3, Gini and IC for measuring diversity and finally the user coverage of the recommenders with UC and UC-Rel. All metrics are reported using a cutoff of 5. In every case, higher values means more accurate/fresh recommendations.

Unless stated otherwise, we report tuned versions of the recommendation algorithms. This tuning has been performed by selecting the best configuration of each algorithm by nDCG@5. Additionally, we show in Figure 5.2 the temporal evolution of the rating distribution in both datasets for the temporal system methodology.

### 5.3.3 Recommenders

In our experiments, we include a set of Recommender Systems that combine implementations provided in different libraries with our own code, as the experiments shown in Chapter 4. These algorithms can be classified into non-personalized baselines, standard

**Table 5.3:** Optimal parameters for MovieTweetings and Foursquare datasets used in the experiments of this chapter. For each dataset, we report the best parameters of the recommenders for each split (temporal system or temporal per user) according to nDCG@5.

| | **MovieTweetings** | | **Foursquare** | |
|---|---|---|---|---|
| **Rec** | **System** | **Per user** | **System** | **Per user** |
| IB: k, sim | 40, VecC | 100, VecC | 120, VecC | 120, VecC |
| UB: k, sim | 100, VecC | 120, SetJ | 120, SetJ | 120, SetJ |
| HKV: k, $\alpha$, $\lambda$ | 10, 10, 1 | 50, 1, 1 | 10, 10, 10 | 10, 10, 0.1 |
| BPRMF: k, $\lambda_u = \lambda_i$, $\lambda_0$ | 10, 0.0025, 1 | 100, 0.001, 1 | 50, 0.001, 0 | 100, 0.01, 1 |
| TD: k, sim, $\lambda$ | 100, SetJ, 0.05 | 120, SetJ, 0.05 | 120, SetJ, 0.05 | 120, VecC, 0.05 |
| BFUB: k, sim, $L_m^- = L_m^+$, norm, Wgt | 120, SetJ, 5, Std, F | 120, SetJ, 5, Std, T | 120, SetJ, 10, Def, T | 120, SetJ, 5, Def, T |
| BFsUB: k, sim, $L_m^- = L_m^+$, norm, Wgt | 120, sim$_1$, 5, Std, T | 120, sim$_1$, 10, Std, T | 120, sim$_1$, 10, Def, T | 120, sim$_3$, 5, Def, T |
| MC: k, $\lambda$ | 20, 0.1 | 50, 0.1 | 20, 0.1 | 10, 0.1 |
| FPMC: k, $\lambda$ | 5, 0.1 | 5, 0.1 | 5, 0.1 | 5, 0.1 |
| Fossil: k, $\lambda$, L | 20, 0.1, 3 | 20, 0.1, 3 | 50, 0.1, 3 | 10, 0.1, 1 |
| Caser: T, d, nh | 2, 10, 16 | 2, 10, 4 | 2, 50, 4 | 2, 50, 4 |

collaborative filtering approaches, time-aware/sequential baselines, and our proposed approaches. Most of the selected recommenders have already been shown in Section 4.5.

First, as non-personalized baselines, we report the random recommender (Rnd) and the popularity recommender (Pop). Second, as standard CF methods, we use the classical item-based (IB) and user-based (UB) $k$-NN recommenders, and the MF recommender using ALS optimization (denoted as HKV (Hu et al., 2008)), as explained in Section 2.2.2. We also include the BPRMF algorithm proposed in Rendle et al. (2009). Third, we include as sequential recommenders the Fossil approach from He and McAuley (2016), two algorithms based on Markov Chains (denoted as MC and FPMC) from the same authors. We also consider Caser as a sequential recommender based on Neural Networks proposed in Tang and Wang (2018).

Finally, we test the following combinations of our approaches (all of them implemented on top of the RankSys framework): our adaptation of the temporal decay method (TD, Equation 5.6) as an example of a time-aware similarity metric, and our backward-forward approach using a standard $k$-NN similarity (BFUB) and a sequence-aware similarity based on LCS (BFsUB). Despite the wide variety of parameters that can be applied in our LCS-based similarity, we have decided to use a simple collaborative similarity without taking ratings into account, ordering the sequences in ascending temporal order (e.g., the function $e$ will be $e_i$). At the same time, for the BFsUB we did not use any $\delta$-matching. The parameters of the tested recommenders are shown in Table 5.2 and the final parameters optimized at nDCG@5 are shown in Table 5.3.

### 5.3.4 Performance analysis: system temporal split

In Tables 5.4 and 5.5 we show the results obtained under the temporal system evaluation methodology in the MovieTweetings and Foursquare datasets respectively.

Let us focus first on the relevance metrics, in particular, in the low results achieved in

**Table 5.4:** Performance results on MovieTweetings dataset. Temporal system split (80% ancient ratings to train, rest to test). In bold, the best recommender in each metric. Best recommenders in nDCG@5.

| Recommender | P | R | nDCG | EPC | MIN | Gini | IC | UC | UC-Rel |
|---|---|---|---|---|---|---|---|---|---|
| Rnd | 0.000 | 0.000 | 0.001 | †**0.996** | 0.410 | †**0.645** | †**0.949** | †**6,518** | †**4,962** |
| Rnd$_{CF}$ | 0.000 | 0.000 | 0.000 | †**0.996** | 0.411 | 0.600 | 0.900 | 5,118 | 3,704 |
| Pop | 0.004 | 0.003 | 0.003 | 0.853 | 0.207 | 0.001 | 0.006 | †**6,518** | †**4,962** |
| Pop$_{CF}$ | 0.003 | 0.003 | 0.003 | 0.854 | 0.210 | 0.001 | 0.006 | 5,118 | 3,704 |
| IB | 0.009 | 0.010 | 0.010 | 0.914 | 0.585 | 0.012 | 0.126 | 5,117 | 3,703 |
| UB | 0.014 | 0.017 | 0.016 | 0.907 | 0.585 | 0.006 | 0.030 | 5,117 | 3,703 |
| HKV | 0.019 | 0.025 | 0.024 | 0.934 | 0.573 | 0.015 | 0.081 | 5,118 | 3,704 |
| BPRMF | 0.014 | 0.015 | 0.016 | 0.923 | 0.579 | 0.017 | 0.125 | 5,118 | 3,704 |
| TD | 0.019 | 0.024 | 0.023 | 0.916 | 0.697 | 0.006 | 0.053 | 5,117 | 3,703 |
| BFUB | 0.026 | 0.031 | 0.031 | 0.927 | 0.728 | 0.009 | 0.077 | 5,117 | 3,703 |
| BFsUB | **0.029** | **0.036** | **0.034** | 0.936 | †**0.828** | 0.007 | 0.076 | 5,117 | 3,703 |
| MC | 0.021 | 0.030 | 0.031 | 0.919 | 0.707 | 0.004 | 0.043 | 4,900 | 3,516 |
| FPMC | 0.015 | 0.018 | 0.020 | 0.913 | 0.634 | 0.004 | 0.040 | 4,900 | 3,516 |
| Fossil | 0.020 | 0.027 | 0.025 | 0.915 | 0.647 | 0.003 | 0.028 | 4,900 | 3,516 |
| Caser | 0.020 | 0.028 | 0.026 | 0.939 | 0.771 | 0.015 | 0.129 | 5,118 | 3,704 |
| Skyline | †**0.520** | 0.630 | 0.806 | 0.977 | 0.588 | 0.108 | 0.295 | 3,545 | 3,545 |
| Skyline$_{CF}$ | 0.494 | †**0.644** | †**0.812** | 0.977 | 0.616 | 0.103 | 0.251 | 2,585 | 2,585 |

this kind of metrics, especially in MovieTweetings. The main reason why this behavior occurs is because we are using a high relevance threshold (we only consider as relevant items those rated with a 9 or 10 in MovieTweetings). However, for Foursquare this is not so critical since in that case we deal with implicit information, hence, all items in the test set are relevant for the user, which is why these metrics obtain higher values, i.e., around 0.2 and 0.18 in case of R and nDCG, respectively. Besides, since we have performed a temporal system split, we may end up having users and items in the test set that do not appear in the training data. Because of that, modeling user profiles under these circumstances becomes a difficult task (as already discussed in the experiments of Chapter 4).

Regarding the results, we observe the difference in performance of the Pop algorithm in both datasets. While in Foursquare it is very competitive with respect to the rest of the models, in MovieTweetings it is one of the worsts (after the Rnd recommender). This, as mentioned throughout the thesis, may be due to the popularity bias, which may be more pronounced in Foursquare as it has a higher sparsity than MovieTweetings. In addition, it should be noted that in MovieTweetings not all items in the test set are relevant (only those scored with a rating higher or equal 9), so this may be somewhat detrimental to this recommender.

With respect to the rest of the recommenders, we observe some interesting results. First, we can see that the IB algorithm is the worst in MovieTweetings after Pop and Rnd, while in the Foursquare dataset it outperforms other recommenders like MC, FPMC, or BPRMF. However, in both datasets, IB always obtains a worse performance than UB. This behavior was observed in the previous chapter, where algorithms based on user-based similarity obtained better results than those based on item-based simi-

**Table 5.5:** Performance results on Foursquare dataset. Temporal system split (80% ancient ratings to train, rest to test). In bold, the best recommender in each metric. Best recommenders in nDCG@5.

| Recommender | P | R | nDCG | EPC | MIN | Gini | IC | UC | UC-Rel |
|---|---|---|---|---|---|---|---|---|---|
| Rnd | 0.001 | 0.002 | 0.001 | †**0.998** | 0.615 | 0.847 | †**1.000** | †**9,217** | †**9,217** |
| $\text{Rnd}_{\text{CF}}$ | 0.001 | 0.001 | 0.001 | †**0.998** | 0.612 | †**0.854** | †**1.000** | 9,130 | 9,130 |
| Pop | 0.070 | 0.181 | 0.130 | 0.879 | 0.515 | 0.001 | 0.004 | †**9,217** | †**9,217** |
| $\text{Pop}_{\text{CF}}$ | 0.069 | 0.182 | 0.130 | 0.879 | 0.515 | 0.001 | 0.004 | 9,130 | 9,130 |
| IB | 0.071 | 0.200 | 0.155 | 0.952 | 0.613 | 0.177 | 0.828 | 9,130 | 9,130 |
| UB | 0.081 | 0.214 | 0.173 | 0.929 | 0.573 | 0.022 | 0.293 | 9,130 | 9,130 |
| HKV | 0.071 | 0.185 | 0.154 | 0.949 | 0.585 | 0.012 | 0.029 | 9,130 | 9,130 |
| BPRMF | 0.070 | 0.187 | 0.146 | 0.886 | 0.511 | 0.003 | 0.071 | 9,130 | 9,130 |
| TD | 0.081 | 0.211 | 0.170 | 0.929 | 0.582 | 0.023 | 0.307 | 9,130 | 9,130 |
| BFUB | 0.081 | 0.214 | 0.173 | 0.929 | 0.573 | 0.022 | 0.293 | 9,130 | 9,130 |
| BFsUB | **0.083** | **0.219** | **0.174** | 0.921 | 0.569 | 0.018 | 0.281 | 9,130 | 9,130 |
| MC | 0.064 | 0.172 | 0.133 | 0.945 | **0.624** | 0.027 | 0.269 | 8,909 | 8,909 |
| FPMC | 0.066 | 0.180 | 0.133 | 0.935 | 0.608 | 0.016 | 0.196 | 8,909 | 8,909 |
| Fossil | 0.078 | 0.206 | 0.163 | 0.938 | **0.624** | 0.012 | 0.131 | 8,909 | 8,909 |
| Caser | 0.079 | 0.208 | 0.170 | 0.929 | 0.610 | 0.028 | 0.301 | 9,130 | 9,130 |
| Skyline | †**0.441** | 0.988 | †**0.998** | 0.960 | †**0.671** | 0.163 | 0.577 | 9,184 | 9,184 |
| $\text{Skyline}_{\text{CF}}$ | 0.436 | †**0.989** | †**0.998** | 0.960 | 0.670 | 0.161 | 0.573 | 9,097 | 9,097 |

larity. However, in the case of the pure user-based algorithm, we can also see how TD outperforms it in MovieTweetings, suggesting that in temporal contexts with realistic timestamps (if they have enough data), it might be crucial to incorporate the temporal information in the recommendations. On the other hand, in the Foursquare the performance of both TD and UB is practically the same.

It is surprising that some of the sequential recommenders (MC, FPMC, and Fossil) do not produce good results in any of the datasets. In fact, the MC recommender outperforms Fossil in MovieTweetings, and although in Foursquare the performance of Fossil is clearly the best one when compared against MC and FPMC (as reported in the original paper), its performance is lower than the UB. One possible reason of such performance from Fossil is that, in the original paper (He and McAuley, 2016), no ranking-based accuracy metrics were tested, only the AUC metric was used, the same metric these algorithms aim to optimize; moreover, they did not use any evaluation threshold in their datasets, so that every item in the test set was considered relevant. Besides, they focused on predicting the next item to consume, as they left for testing the recommender only the last item rated by the user in the dataset. At the same time, state-of-the-art recommenders such as nearest-neighbors were ignored in the experimental comparison in that paper. Similarly, Caser does not perform as well as in the original paper (Tang and Wang, 2018). We hypothesize that we have not replicated those positive results mostly because the authors performed a heavy pruning of the dataset– removing all the cold-start users (those with less than 15 or 10 interactions, depending on the dataset) – and they did not include more traditional recommenders (such as $k$-NN or MF) which are very competitive in our experiments. In any case, sequential recommenders remain competitive, even beating other baselines such as IB, UB, and BPRMF in MovieTweetings; and HKV and BPRMF in Foursquare. In fact, Caser and MC show the best performance among the sequential baselines in MovieTweetings, whereas Caser and Fossil obtain the best results in Foursquare.

Let us analyze the performance of our sequential-based $k$-NN. As aforementioned, BFsUB refers to our backward-forward proposal with a sequential similarity (based on LCS) and BFUB refers to our backward-forward proposal with a classic neighbor similarity (VecC or SetJ). As we can see in the two datasets with this configuration, they are the best ones in terms of accuracy, being the BFsUB the best among them. This is an interesting conclusion because it demonstrates the usefulness of incorporating temporal and/or sequential information in classical models, being able to overcome other more complex algorithms in the area. Furthermore, although there are many parameters that can be configured in proposals such as MFs or neural networks, it should be noted that these parameters are sometimes difficult to justify, that is to say, that sometimes it is not clear why modifying some parameters of regularization or latent factors can improve or worsen the performance of some of the algorithms, whereas in our proposal it is relatively simple to explain the effect that each one of the parameters has in the final model. Hence, we can conclude that using temporal information in a recommender when performing a system temporal split improves the ranking quality.

In terms of temporal novelty (freshness) we notice some interesting patterns. First, we see that in MovieTweetings dataset, Caser, BFsUB, and BFUB obtain better results than the rest of the recommenders. This is an expected result since these methods give more importance to the items that are closer to the test split or to the last interacted items by the neighbors, hence, it is more likely that these recommendations are more temporally novel than those from time-agnostic approaches. Nevertheless it is also relevant that this effect does not happen in Foursquare, where most of the recommenders achieve a comparable level of the freshness metric MIN. This could be attributed to the fact that the timespan in this dataset is rather short (less than a year), whereas in the other dataset many years of interactions are available, so the temporal patterns that we may be learning and evaluating upon in Foursquare may not be discriminating enough. Besides, as we can see in Figure 5.2, the rating distribution in Foursquare is less standard than the one in MovieTweetings.

Finally, regarding the user coverage of the recommenders, we find interesting results. First of all, as always, the Pop and Rnd recommenders have full coverage. In this case, we can also observe in the MovieTweetings dataset that there is a large number of users who do not have any relevant item in test (as shown by the Skyline coverage). This is due to the sum of two factors, the relevance threshold (9 for MovieTweetings) and the fact that since it is a system split, there may be users who do not appear in the training set. However, in both datasets it can be seen how our sequential proposals do not lose coverage in the recommendations produced.

**Table 5.6:** Performance results on MovieTweetings dataset. Temporal per user split (last 2 actions for every user with at least 6 interacions to the test set). In bold, the best recommender in each metric. Best recommenders in nDCG@5.

| Recommender | P | R | nDCG | EPC | MIN | Gini | IC | UC | UC-Rel |
|---|---|---|---|---|---|---|---|---|---|
| Rnd | 0.000 | 0.001 | 0.000 | †**0.996** | 0.383 | †**0.722** | †**0.980** | †**13,860** | †**6,873** |
| Rnd$_{\text{CF}}$ | 0.000 | 0.000 | 0.000 | †**0.996** | 0.383 | 0.721 | †**0.980** | †**13,860** | †**6,873** |
| Pop | 0.008 | 0.034 | 0.024 | 0.870 | 0.159 | 0.001 | 0.005 | †**13,860** | †**6,873** |
| Pop$_{\text{CF}}$ | 0.008 | 0.034 | 0.024 | 0.870 | 0.159 | 0.001 | 0.005 | †**13,860** | †**6,873** |
| IB | 0.019 | 0.073 | 0.050 | 0.919 | 0.402 | 0.018 | 0.185 | †**13,860** | †**6,873** |
| UB | 0.020 | 0.072 | 0.049 | 0.910 | 0.360 | 0.007 | 0.038 | †**13,860** | †**6,873** |
| HKV | 0.020 | 0.076 | 0.050 | 0.934 | 0.367 | 0.019 | 0.075 | †**13,860** | †**6,873** |
| BPRMF | 0.014 | 0.054 | 0.037 | 0.933 | 0.363 | 0.039 | 0.218 | †**13,860** | †**6,873** |
| TD | 0.030 | 0.116 | 0.081 | 0.916 | 0.451 | 0.009 | 0.077 | †**13,860** | †**6,873** |
| BFUB | 0.027 | 0.102 | 0.070 | 0.918 | 0.424 | 0.010 | 0.054 | †**13,860** | †**6,873** |
| BFsUB | **0.040** | **0.155** | **0.111** | 0.928 | **0.518** | 0.012 | 0.086 | †**13,860** | †**6,873** |
| MC | 0.025 | 0.094 | 0.062 | 0.905 | 0.436 | 0.004 | 0.073 | †**13,860** | †**6,873** |
| FPMC | 0.015 | 0.057 | 0.038 | 0.913 | 0.365 | 0.006 | 0.065 | †**13,860** | †**6,873** |
| Fossil | 0.019 | 0.073 | 0.050 | 0.909 | 0.386 | 0.004 | 0.045 | †**13,860** | †**6,873** |
| Caser | 0.031 | 0.120 | 0.083 | 0.928 | 0.483 | 0.019 | 0.158 | †**13,860** | †**6,873** |
| Skyline | †**0.269** | †**1.000** | †**1.000** | 0.962 | †**0.525** | 0.092 | 0.260 | 6,873 | †**6,873** |
| Skyline$_{\text{CF}}$ | †**0.269** | †**1.000** | †**1.000** | 0.962 | †**0.525** | 0.092 | 0.260 | 6,873 | †**6,873** |

### 5.3.5 Performance analysis: per user temporal split

Let us now analyze the results obtained when using the temporal per user split. Tables 5.6 and 5.7 show the performance of the MovieTweetings and the Foursquare datasets. As we can observe, the performance of the recommenders in general is higher than the previous results. This is because under this configuration the test set is only formed by the last 2 interactions of the users, so the training set has more information than in the temporal system configuration.

Regarding the behavior of the recommendation approaches, we observe some situations that are very different to what we found using the temporal system split. IB does not perform as bad as before, in fact, they are better or at the same level than sequential recommenders. Interestingly, the baselines based on sequences (MC, FPMC, and Fossil) do not achieve as good results as it should be expected, even though this evaluation methodology is closer to the one presented in the original paper (He and McAuley, 2016); however, as discussed before, we should consider that in that paper the authors reported results based on the AUC metric whereas we are evaluating with other ranking-based accuracy metrics and against other baselines. In the Foursquare dataset, the performance of Fossil is better than other state-of-the-art algorithms, although MC, again, is the best performing sequential baseline in MovieTweetings, being able to beat Fossil.

We now note that one aspect where both evaluation methodologies agree on is that the proposed time-aware and sequential-aware recommenders are very competitive in terms of ranking-based accuracy metrics. Here, we observe this is especially clear in MovieTweetings, where time-agnostic recommenders obtain very bad results; in fact, our proposed approaches (BFUB, BFsUB) achieve the highest values, although some-

**Table 5.7:** Performance results on Foursquare dataset. Temporal per user split (last 2 actions for every user with at least 6 interacions to the test set). In bold, the best recommender in each metric. Best recommenders in nDCG@5.

| Recommender | P | R | nDCG | EPC | MIN | Gini | IC | UC | UC-Rel |
|---|---|---|---|---|---|---|---|---|---|
| Rnd | 0.001 | 0.002 | 0.001 | †**0.998** | 0.540 | 0.849 | †**1.000** | †**8,986** | †**8,986** |
| $\text{Rnd}_{\text{CF}}$ | 0.001 | 0.002 | 0.002 | †**0.998** | 0.538 | †**0.858** | †**1.000** | †**8,986** | †**8,986** |
| Pop | 0.069 | 0.172 | 0.133 | 0.878 | 0.501 | 0.002 | 0.004 | †**8,986** | †**8,986** |
| $\text{Pop}_{\text{CF}}$ | 0.069 | 0.172 | 0.133 | 0.878 | 0.501 | 0.002 | 0.004 | †**8,986** | †**8,986** |
| IB | 0.089 | 0.222 | 0.186 | 0.950 | 0.535 | 0.147 | 0.829 | †**8,986** | †**8,986** |
| UB | **0.091** | **0.228** | 0.191 | 0.926 | 0.516 | 0.014 | 0.169 | †**8,986** | †**8,986** |
| HKV | 0.081 | 0.202 | 0.174 | 0.948 | 0.503 | 0.014 | 0.032 | †**8,986** | †**8,986** |
| BPRMF | 0.075 | 0.188 | 0.157 | 0.947 | 0.515 | 0.056 | 0.489 | †**8,986** | †**8,986** |
| TD | 0.088 | 0.220 | 0.185 | 0.929 | 0.536 | 0.018 | 0.232 | †**8,986** | †**8,986** |
| BFUB | **0.091** | **0.228** | **0.192** | 0.927 | 0.515 | 0.014 | 0.176 | †**8,986** | †**8,986** |
| BFsUB | **0.091** | 0.227 | 0.190 | 0.925 | 0.515 | 0.018 | 0.259 | †**8,986** | †**8,986** |
| MC | 0.077 | 0.192 | 0.159 | 0.940 | 0.558 | 0.016 | 0.185 | †**8,986** | †**8,986** |
| FPMC | 0.074 | 0.185 | 0.145 | 0.933 | 0.554 | 0.009 | 0.149 | †**8,986** | †**8,986** |
| Fossil | 0.087 | 0.217 | 0.177 | 0.939 | **0.563** | 0.010 | 0.080 | †**8,986** | †**8,986** |
| Caser | 0.087 | 0.217 | 0.182 | 0.932 | 0.559 | 0.029 | 0.307 | †**8,986** | †**8,986** |
| Skyline | †**0.400** | †**1.000** | †**1.000** | 0.960 | †**0.568** | 0.174 | 0.687 | †**8,986** | †**8,986** |
| $\text{Skyline}_{\text{CF}}$ | †**0.400** | †**1.000** | †**1.000** | 0.960 | †**0.568** | 0.174 | 0.687 | †**8,986** | †**8,986** |

times BFUB obtains a worse performance than TD. Hence, we conclude that using temporal information in a recommender when performing a temporal per user split does not necessarily improves the ranking quality but it helps achieving high performance, mostly due to inconsistencies in the way this evaluation methodology deals with the preference data.

When analyzing the temporal novelty of the recommendations, we observe an strange behavior in the datasets. The BF approaches achieve lower values than in the previous tables; this effect, despite being counter-intuitive, can be explained if we bear in mind that this evaluation methodology is not considering a temporal system for all users, so there can be users that stop being active at the beginning of the system lifetime and their recommendations may be less fresh than those from other users. The TD recommender is not affected by this, as in that case the method considers the timestamp, not the specific interaction sequence of the user. Nevertheless, we note that for the MovieTweetings dataset it is clear that the time-aware neighborhood recommenders are able to improve the performance of their corresponding time-agnostic approaches in terms of both relevance and freshness.

Finally, regarding the user coverage in this split, we observe a similar behavior to the one reported in the temporal system split. Nevertheless, in this case, the difference between full coverage and coverage of relevant users in MovieTweetings dataset might be more striking. This effect in the temporal per user split can only be due to the relevance threshold, as all test users appear in the training set, showing us that sometimes applying high relevance thresholds produces significant effects on the performance of recommendations. However, as we observe, our BF proposals again have a coverage equal to that reported by other collaborative algorithms, proving that, in fact, our

**Table 5.8:** Performance results on MovieTweetings and Foursquare dataset. Temporal system split (80% ancient ratings to train, rest to test). In bold, the best recommender in each metric. Best recommenders in nDCG@5 in every combination of the BF approaches.

| **Recommender** | | | **MovieTweetings** | | | | **Foursquare** | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Sim** | **Norm** | **Wgt** | **nDCG** | **EPC** | **MIN** | **IC** | **nDCG** | **EPC** | **MIN** | **IC** |
| BFUB | Rks | F | 0.029 | 0.929 | 0.745 | 0.055 | 0.169 | 0.930 | 0.578 | 0.292 |
| | Rks | T | 0.029 | 0.929 | 0.744 | 0.058 | 0.170 | **0.931** | **0.580** | **0.354** |
| | Std | F | 0.031 | 0.927 | 0.728 | 0.077 | 0.169 | 0.929 | 0.572 | 0.285 |
| | Std | T | 0.030 | 0.930 | 0.721 | **0.104** | 0.173 | 0.929 | 0.573 | 0.293 |
| | Def | F | 0.030 | 0.924 | 0.739 | 0.042 | 0.169 | 0.929 | 0.572 | 0.285 |
| | Def | T | 0.030 | 0.924 | 0.738 | 0.043 | 0.173 | 0.929 | 0.573 | 0.293 |
| BFsUB | Rks | F | 0.030 | 0.936 | 0.830 | 0.046 | 0.171 | 0.922 | 0.575 | 0.282 |
| | Rks | T | 0.030 | 0.936 | 0.832 | 0.050 | 0.173 | 0.925 | 0.577 | 0.307 |
| | Std | F | **0.034** | 0.936 | 0.824 | 0.076 | 0.171 | 0.921 | 0.568 | 0.272 |
| | Std | T | **0.034** | 0.936 | 0.828 | 0.076 | **0.174** | 0.921 | 0.569 | 0.281 |
| | Def | F | 0.032 | **0.937** | **0.835** | 0.052 | 0.171 | 0.921 | 0.568 | 0.272 |
| | Def | T | 0.032 | 0.936 | 0.833 | 0.050 | 0.174 | 0.921 | 0.569 | 0.281 |

proposals obtain superior performance in both accuracy and freshness metrics, since the coverage is comparable.

### 5.3.6 Sensitivity of backward-forward components

To better understand the behavior of the different components in the proposed neighborhood BF approach, we show in Tables 5.8 (for the temporal system strategy) and 5.9 (for the temporal per user strategy) the results for the best configurations using a sequence-aware similarity metric (LCS) against a classic neighborhood similarity (BFsUB vs BFUB) using our backward-forward approach. In these tables, we compare these similarities when using the normalization strategies (Nrm) defined in Section 5.2.2 and testing whether to consider the weight of the neighbors or not (column Wgt).

Based on these results, we observe that the performance achieved by the sequential similarity is always the best one in terms of accuracy except in the case of Foursquare with the temporal per user split, however, the differences are, in general, small. We hypothesize that considering sequences by using the BF method is enough to capture the user preferences, and by also using a sequential similarity we impose too many constraints on the algorithm to find proper, valid neighbors.

Moreover, regarding the other components of the BF approach, we observe that usually the best results are obtained with the Standard (Std) aggregation function in both evaluation strategies specially when using the BFsUB similarity, although other functions do not really produce very different results; however, the use of the neighbor similarity to weight her contribution is important and tends to obtain better results when it is used (T), at least in Foursquare.

Finally, in terms of freshness and novelty and diversity we can see that, in general, BFsUB reports higher results than BFUB, except for Foursquare using the system evaluation methodology. For the freshness dimension, this is an expected behavior

**Table 5.9:** Performance results on MovieTweetings and Foursquare dataset. Temporal per user split (last 2 actions for every user with at least 6 interacions to the test set). In bold, the best recommender in each metric. Best recommenders in nDCG@5 in every combination of the BF approaches.

| Recommender | | | MovieTweetings | | | | Foursquare | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Sim** | **Norm** | **Wgt** | **nDCG** | **EPC** | **MIN** | **IC** | **nDCG** | **EPC** | **MIN** | **IC** |
| BFUB | Rks | F | 0.069 | 0.921 | 0.426 | 0.061 | 0.186 | 0.927 | 0.518 | 0.170 |
| | Rks | T | 0.069 | 0.923 | 0.431 | 0.074 | 0.189 | **0.928** | 0.519 | 0.197 |
| | Std | F | 0.062 | 0.927 | 0.416 | 0.127 | 0.187 | 0.927 | 0.515 | 0.158 |
| | Std | T | 0.060 | 0.931 | 0.415 | **0.162** | **0.192** | 0.927 | 0.515 | 0.176 |
| | Def | F | 0.070 | 0.918 | 0.425 | 0.049 | 0.187 | 0.927 | 0.515 | 0.158 |
| | Def | T | 0.070 | 0.918 | 0.424 | 0.054 | 0.192 | 0.927 | 0.515 | 0.176 |
| BFsUB | Rks | F | 0.095 | 0.932 | 0.513 | 0.070 | 0.187 | 0.922 | 0.520 | 0.162 |
| | Rks | T | 0.095 | **0.933** | 0.516 | 0.079 | 0.189 | 0.924 | **0.522** | 0.186 |
| | Std | F | 0.110 | 0.930 | **0.520** | 0.091 | 0.187 | 0.922 | 0.517 | 0.140 |
| | Std | T | **0.111** | 0.928 | 0.518 | 0.086 | 0.190 | 0.925 | 0.515 | **0.259** |
| | Def | F | 0.106 | 0.927 | 0.517 | 0.056 | 0.187 | 0.922 | 0.517 | 0.140 |
| | Def | T | 0.106 | 0.927 | 0.517 | 0.058 | 0.190 | 0.925 | 0.515 | 0.259 |

since, when selecting the neighbors, we give more importance to the users that have rated the same items in the same order, hence, the items that are recommended at the end of the sequences are more likely to be retrieved. In the case of IC and EPC, we are not able to observe major differences between the BFUB and BFsUB, but we can see how, in general, for both approaches when the Std normalization is used, the item coverage increases considerably. This may be due to the fact that with Std, as the scores of the candidates are normalized, the differences between them are somewhat reduced, allowing more different items to be included in the final rankings.

## 5.4 Discussion

In this chapter, we have addressed RG3: *Develop mechanisms to add sequentiality in neighborhood-based recommender systems.* Thus, we have presented two techniques for incorporating sequentiality in neighborhood-based recommender systems. First, we have shown how to adapt the Longest Common Subsequence (LCS) algorithm to be used as a similarity metric between two users (extending the formulation presented in Section 4.4 for comparing the recommended list and the test set of the user). Secondly, we have proposed a new formulation for neighborhood-based recommendation algorithms (named backward-forward) that allows to integrate the temporal and sequential dimensions based on rank fusion techniques seamlessly and successfully, according to the reported experiments.

To evaluate the performance of our proposals, we have used two different temporal splits. The first one is a community-centered split using proportional size (temporal system); the second split performed a per user evaluation with a fixed size (temporal per user). In our experiments we have again confirmed how the evaluation methodology significantly affects the performance of the algorithms and how not always the most complex models obtain the best results. In fact, in terms of accuracy, our models have proven to be really competitive, although regarding other dimensions like novelty and diversity our approaches are (sometimes) outperformed by other approaches. Nevertheless, even though our algorithms may have a lot of potential (not only in terms of performance but also in terms of interpretability), we believe that further experiments should be performed on more datasets and other domains to generalize these results.

In this regard, we would like to emphasize again the difficulty of finding datasets with realistic temporal information. In fact, although the datasets used in these experiments have more consistent timestamps than other classical ones (like Movielens), they are still not perfect, since sometimes we have been able to observe how the same user has consumed several items in the same moment of time. We consider this aspect critical, since algorithms that build sequences by ordering the interactions in a temporal way may be creating incorrect or meaningless sequences due to “ties” between the timestamp values of the interactions. This is also a disadvantage of our proposals and therefore, we would like to test how our algorithms work in other environments or domains by performing online experiments.

# 6

# Sequences in POI recommendation

In Chapter 3 we described the Point-of-Interest recommendation problem and we showed its main differences with respect to the traditional recommendation formulation, while in Chapter 5 we investigated how to incorporate sequentiality in $k$-NN recommenders. In this chapter, we analyze the effect of sequences in Point-of-Interest recommendation and see how to apply simple techniques based on reranking to recommend not only independent POIs but complete routes to the users. To do so, first, in Section 6.1 we describe how to obtain routes from the check-ins available in LBSNs datasets. Then, in Section 6.2 we describe our reranking approaches to generate routes for the users based on sequential techniques like Markov Chains and the Longest Common Subsequence algorithm. We make use of the sequential metrics proposed in Section 4.4 and evaluate the performance of the recommenders in Section 6.3 and finally in Section 6.4 we discuss about our reranking approaches.

The work presented in this chapter has been published in the following article:

- **Pablo Sánchez** and Alejandro Bellogín. Applying reranking strategies to route recommendation using sequence-aware evaluation. *User Modeling and User-Adapted Interaction*, 30(4):659-725, 2020.

## 6.1 A general framework to build routes from check-in datasets

When analyzing touristic routes, users follow travel sequences by visiting POIs that are related with each other (e.g., it is common to visit POIs that are close to each other) (Miller, 2004) and following a specific order, where the starting and end points play an important role in the definition of such sequences. However, raw, public data in check-in datasets are not presented in the form of routes or itineraries, but as (somewhat) independent interactions between users and POIs, where a user may check-in in the same venue more than once. Hence, restoring these routes is critical if we aim to learn, recommend, and evaluate interesting and useful travel sequences – also called trips, itineraries, or trajectories in the literature – to the users.

With this goal in mind, we present here a framework that includes three steps tailored to sequence-aware venue recommendation: preprocessing the data, building the sequences, and filtering the sequences. In each step, we provide a description and solutions to the inherent challenging issues, together with a list of the main parameters that could be used when needed. For such a framework, we took inspiration from works on session identification on web search and e-commerce (Jansen et al., 2007, Spiliopoulou et al., 2003), due to the commonalities between a user session in those domains and touristic sequences or routes. It should be noted, however, that existing datasets in those domains such as YOOCHOOSE (Ben-Shimon et al., 2015) and Tmall (Liu et al., 2016b), tend to provide explicit sessions, something, at the moment, not so common in the POI recommendation domain, which emphasizes the importance of such a framework to build travel sequences. We have also integrated and formalized ad-hoc strategies developed explicitly in the venue and route recommendation domains, such as the works of Choudhury et al. (2010), Lim et al. (2015) and Lim et al. (2018), even though they were proposed to work with data from Flickr (mostly pictures taken at specific coordinates), not from standard LBSNs, without establishing a common working methodology, hence, making it difficult to perform comparisons between these different approaches.

Thus, by borrowing ideas from the aforementioned works, we now introduce a generic framework to obtain meaningful routes or travel sequences from raw check-in datasets; for a visual description of the different steps we show Figure 6.1, together with Table 6.1 where we summarize the most important parameters considered in each step.

Step 1: Preprocessing data:

- Description: remove users or items the system has very little information about (e.g., cold-start) (Gunawardana and Shani, 2015). At the same time, only a subset of the users or items might want to be considered (for instance, only cultural venues – i.e., museums – or users identified as tourists) (Liu et al., 2017). However, due to the characteristics of the domain, further analyses could be performed on the data, analogous to those taken on e-commerce or web search data, where noise in the interactions is also prevalent.
- Solution: identify and remove noisy check-ins, for example, too many check-ins in the same instant, either intentionally or because of a bug in the application that collects the data; also users could be classified as spam, e.g., due to explicit attacks (Burke et al., 2015) or because a not-human behavior – bot – is identified, i.e., when users visit a high number of POIs in a short time.
- Parameters to consider: minimum number of check-ins provided by users and items to be included in the processed data ($p_u$ and $p_i$), alternatively, a $p_c$-core subset of the data might be computed by forcing that every user has

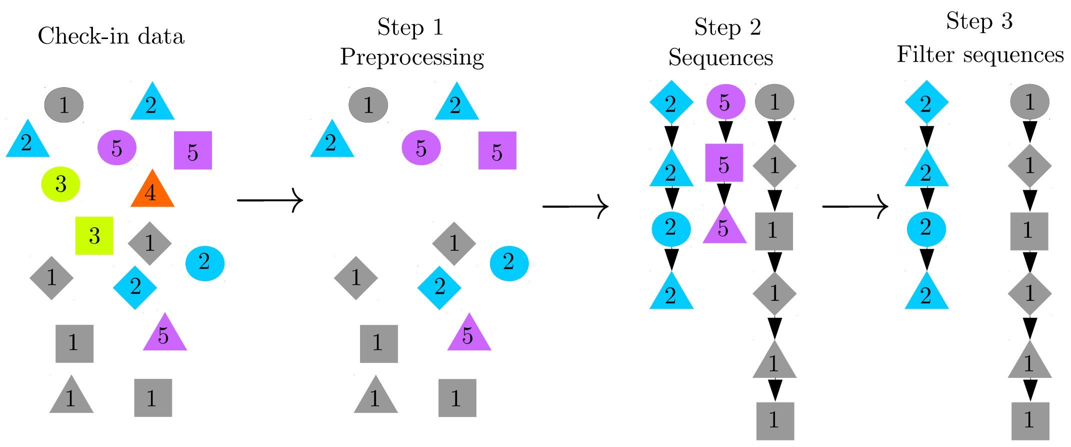


**Figure 6.1:** Visual description of the framework presented in Section 6.1 to obtain sequences from raw check-ins. Each number and color represent a different user in the system and each shape depicts the type of the POI (museums, restaurants, hotels, etc.) a user interacted with (not necessarily a different POI each time).

interacted with at least $p_c$ items and every item by at least $p_c$ users (Rendle et al., 2010), a concept borrowed from graph theory to describe that in a $k$-core (where $k = p_c$) every node has at least $k$ connections; additionally, a reported strategy to distinguish bots from valid users consists of removing those users that spent less than $p_b^t$ seconds transitioning between venues a given number of times ($p_b^i$), see (Palumbo et al., 2017) for more details.

- Challenging issues: correctly identifying noisy data without removing too much valid or useful information corresponding to users and items; besides, reproducibility of the reported results could be virtually impossible if these preprocessing steps are not properly explained (Said and Bellogín, 2015, Dacrema et al., 2019).

Step 2: Building sequences:

- Description: group those check-ins the user visited during the same route in a sequence (as if it was an e-commerce or web session). In this domain, temporal and geographical information are important signals that can be exploited to identify different groups of check-ins.
- Solution: build different sequences of POIs ordering them by timestamp for every user; for instance, if the corresponding venues are too far away from each other or if the timespan is too long, they may belong to different routes, since it is assumed that a user does not follow a given route indefinitely, as she has to rest and sleep (Choudhury et al., 2010, Lim et al., 2015). We

might also impose other constraints on the sequences such as a maximum number of visited venues, traveled length, or time spent in the route. In fact, these constraints could be defined based on dynamic thresholds according to previous check-ins. In any case, these alternatives are left for future work, due to the lack of large-scale datasets with ground truth information to test these hypotheses.

- Parameters to consider: maximum temporal difference between two consecutive check-ins ($b_t$) and maximum distance between two consecutive venues ($b_g$) to consider they belong to the same sequence.
- Challenging issues: building correct sequences, since they could be used to train and evaluate the algorithms, hence, if these sequences are not realistic (too long, too short, or the constraints imposed are too flexible or strict) the whole recommendation process will be adulterated and invalid conclusions might be reached.

Step 3: Filtering sequences:

- Description: identify and remove noisy or incomplete sequences, so that the system could train with the most informative data regarding users and items.
- Solution: remove routes with very few check-ins and/or users having very few sequences.
- Parameters to consider: minimum and maximum length (defined as the number of check-ins) of each travel sequence ($f_m$ and $f_M$), minimum number of routes linked to each user and item ($f_u$ and $f_i$).
- Challenging issues: as in the first step, discriminating which sequences are noisy is critical, but care must be taken to not define too strict constraints, otherwise, we might produce data biased to specific types of users or items (those with larger interactions, popular items, etc.) or very small datasets.

Once the raw check-in data has been processed according to this framework, we would obtain tuples $(u, i, t, s_u^n)$, where $s_u^n$ denotes the $n$-th travel sequence associated to user $u$, which would contain every item identified as belonging to the same route. Since we still have the typical user-item tuples, we could even make use of classical splitting, recommendation, and evaluation techniques, however, the unique advantage of such a processed dataset would be to exploit the sequences in the rest of the stages.

First, regarding data splitting, and assuming we want a time-aware evaluation strategy (Campos et al., 2014) where the test set should occur after the training set – at least in a user basis –, we have the following possibilities: find a global timestamp where all the sessions after that timestamp are included in the test set, or decide a number of sessions $t_n$ to be included in the test set and create the test split by taking the last $t_n$ sessions of every user, the remaining information would determine the training split (see Section 2.4.2 for more details). Additional constraints could be imposed on the length of the sessions in test ($t_l$) and the minimum number of sessions in training ($t_s$)

Table 6.1: Summary of the parameters used in the framework presented in Section 6.1.

| Step | Parameter | Description |
|---|---|---|
| Preprocessing data | $p_u, p_i$ | Minimum number of check-ins a user ($p_u$) or item ($p_i$) must have to be considered in the dataset |
| | $p_c$ | All users and items must interact with $p_c$ items and users respectively |
| | $p_b^t, p_b^i$ | Number of times ($p_b^i$) a user has consumed consecutive items spending less than a number of seconds ($p_b^t$) |
| Building sequences | $b_t$ | Maximum temporal difference between two consecutive check-ins to be considered in the same sequence |
| | $b_g$ | Maximum distance between two consecutive check-ins to be considered in the same sequence |
| Filtering sequences | $f_m, f_M$ | Minimum ($f_m$) or maximum ($f_M$) length of each travel sequence to be considered in the final dataset |
| | $f_u, f_i$ | Minimum number of sessions linked to each user ($f_u$) or item ($f_i$) to be considered in the final dataset |

for a user to be included in the test set, so as to have more control on the users being tested. In any case, it is important to consider complete (not partial) sequences of the users in this process (Quadrana et al., 2018).

Then, at the recommendation stage, we could again use standard recommendation approaches or ad-hoc techniques able to exploit the travel sequences. In both situations, a decision should be made about the repetitions in the system since, contrary to classic RS where the users consume an item only once, in this domain a user may check-in in the same venue an unlimited number of times. Hence, at least the following three possibilities open up: transform the data as in classic RS (only one interaction between users and items remains in the data), aggregate the check-ins in an item basis, so that the frequency could at least discriminate the most interesting venues for a user from the rest, as done with implicit feedback data (Hu et al., 2008), or keep the repeated interactions. It should be noted that only the last strategy allows to maintain the original temporal information – i.e., check-in timestamps or their sequential order available in the system.

Finally, the evaluation should be aware of the sequences followed by the user (in the test set), especially when the recommendations provided are assumed to be visited in the order returned by the algorithm, which is the main premise in this work. Because of this, we will make use of the sequential metrics that we introduced in Section 4.4. In the next section we present a general recommendation approach that aims to produce meaningful routes or travel sequences from sequence-agnostic algorithms.

## 6.2 A novel approach for sequential venue recommender systems based on reranking

### 6.2.1 Item reranking in retrieval and recommendation

In the fields of Recommender Systems and Information Retrieval, some models frequently tend to recover very similar items in top-n rankings. To solve this problem, researchers have proposed different techniques to improve the diversity of the results (Santos et al., 2015, Castells et al., 2015). Among them, possibly the best known and most popular approach is item reranking, a strategy whose objective is to improve the quality of a list of recommended items by reordering them according to a topic diversification model. Some articles related to this subject include the works of Ziegler et al. (2005), where the authors propose a diversification approach by minimizing the intra-list similarity between items, the methods to collect diverse results by exploiting past query reformulation of the user's query from Radlinski and Dumais (2006), and the probabilistic xQuAD framework presented in Santos et al. (2010) where the authors analyze the underlying aspects of a query $q$ in the form of sub-queries in order to obtain more diverse results.

At the same time, this issue has been adapted to and analyzed in the recommendation context; for instance, Vargas et al. (2011) introduced the notion of user intent as a translation of the query intents from retrieval, this idea was extended in Wasilewski and Hurley (2018), where the authors injected components based on user intents in item similarity measures; from a different perspective, Kaminskas and Bridge (2017) provided an experimental comparison where the same reranking framework is exploited but on different criteria: diversity, serendipity, novelty, and coverage, this allows to analyze the cross-effects and correlations between these criteria on several recommendation algorithms. More recently, these techniques have been used to reduce the popularity bias (Abdollahpouri et al., 2019b), which is equivalent to promote novelty on the results, as it was already explored in some of the previous works, although the authors here focused on maintaining acceptable levels of recommendation accuracy.

### 6.2.2 Using item reranking to generate sequences of venues

In this chapter, we aim to optimize different criteria, all of them related (based on our hypotheses) to more realistic and useful routes or venue sequences from the user perspective, such as shorter routes or popular transitions between venues (according to the collaborative knowledge or to their attributes). With this goal in mind, we propose to exploit item reranking techniques to create more meaningful sequences of items, in particular, we propose to start from non-sequential recommender systems and generate sequential recommendations, an approach, as far as we know, novel in the area of venue and route recommendation.

Hence, based on the formulation from Kaminskas and Bridge (2017), we define an objective function $f_{obj}(u, i, R_u)$ that is used in a greedy reranking process, where we select the item $i$ maximizing such function among the candidate items available at any moment – where the original set of candidate items come from a recommendation algorithm –; then, that item is removed from the candidate items and concatenated in the recommendation list $R_u$ to be returned to the user. As stated by Kaminskas and Bridge (2017), researchers typically formulate this objective function as a linear combination of the item's relevance and the complementary dimension that we aim to maximize (usually diversity in the works surveyed in the previous section); in our case, such function combines the output of a recommendation algorithm and the utility provided by the sequence-aware reranker component, which are denoted as $f_{rec}(u, i, R_u)$ (although since most recommendation algorithms typically ignore the previously ranked items, the notation could be simplified to $f_{rec}(u, i)$) and $f_{seq}(u, i, R_u)$, respectively:

$$f_{obj}(u, i, R_u) = \lambda \cdot f_{rec}(u, i) + (1 - \lambda) \cdot f_{seq}(u, i, R_u) \tag{6.1}$$

As we shall show later, since some of the proposed $f_{seq}$ functions are able to provide complete recommendation sequences – instead of a pointwise score in an item-basis –, we combine the scores provided by each function after doing a rank-based normalization (Renda and Straccia, 2003). This means that we use the scores provided by the recommender and the reranker components to sort the items, then, each item is assigned a score based on its position; the final, combined value of $f_{obj}(u, i, R)$ thus depends on this normalized score and the weight $\lambda$.

We propose 8 different formulations for the sequence-aware reranker component $f_{seq}$ (for simplicity, we shall refer to this function also as reranker since it is the main discriminating piece in the whole reranking process), classified in the following 3 families (a summary of these approaches can be found in Table 6.2):

- Independent: the score only depends on the target user-item pair. The reranking procedure has no memory, hence, it does not incorporate any sequential component and the reranked items are independent of each other:
  - **Random:** the items are reranked randomly: $f_{seq}^{rnd}(u, i, R_u) = \text{rnd} \in [0, 1]$.
  - **Recommender-based:** the items are reranked using a score $r(u, i)$ produced by a recommender for user $u$ and item $i$ (e.g., popularity, user neighborhood, etc.): $f_{seq}^{rec}(u, i, R_u) = r(u, i)$. It is important to note that if the recommender used to perform the reranking and the one to produce the candidate items are different, the resulting list (its order) could be very different; in particular, this reranker opens up the possibility of producing personalized sequences based on a non-personalized algorithm like the popularity recommender, where the output produced by such a reranking compared against the ones obtained directly using either recommenders will be potentially very different.
- Dependent on the previous item: the score for item $i$ depends only on the last item included in list $R_u$; let us denote such last item as $i_{n-1}$, i.e., $R_u = \{i_1, \cdots, i_{n-1}\}$. We define three rerankers that aim to maximize a particular dimension between

the target item $i$ and previous item $i_{n-1}$; hence, these rerankers optimize different criteria based on a 2-length sequence by exploiting the last item:

  - **Distance:** reranker that selects the closest venue to the previous suggested one: $f_{seq}^{dist}(u, i, R_u) = 1/dist(i_{n-1}, i)$.
  - **Feature-based Markov Chain:** reranker that selects those venues whose features maximize the transition probability with respect to target item $i$'s features: $f_{seq}^{feat}(u, i, R_u) = p(i^a|i_{n-1}^a)$, where $i^a$ denotes the attributes of item $i$.
  - **Item-based Markov Chain:** reranker that selects the venue that is more frequently visited after item $i_{n-1}$: $f_{seq}^{item}(u, i, R_u) = p(i|i_{n-1})$.

- Dependent on the whole sequence: the score depends on the entire sequence being generated, i.e., the current list $R_u$ and the potential following item $i$. We define three different algorithms:

  - **LCS-based:** reranker that maximizes the Longest Common Subsequence (an algorithm that we have defined in Section 4.4) between the sequence of item features in the current recommended list $R_u$ assuming item $i$ is recommended at the end $(R_u+i)$ and the item features built from the training set of the user $(u^a)$ ordered by timestamp: $f_{seq}^{lcs}(u, i, R_u) = lcs((R_u+i)^a, u^a)$.
  - **Suffix tree:** reranker that searches in linear time whether a specific substring exists or not in a given sequence, in this case, it searches whether the last $m-1$ recommended items attached to each of the candidate items (denoted as $\{(R_u+i)\}_m$, where $\{s\}_m$ stands for the last $m$ items in a sequence $s$) can be found in the suffix tree built from the item features of the user profile $u^a$: $f_{seq}^{stree}(u, i, R_u) = \delta_{ST(u^a)}(\{(R_u+i)^a\}_m)$, where $\delta_{ST(u^a)}(s)$ denotes whether the suffix tree ST contains the sequence $s$.
  - **Oracle:** reranker whose output will be the sequence of POIs returned by the recommender in the same order as they appear in the test set of the user, it is, hence, the ideal reranker in terms of accuracy metrics and is used as an upper-bound for the rest of the reranking strategies: $f_{seq}^{oracle}(u, i, R_u) = order_{test}(u, i)$. It should be noted, however, that this reranker is not realistic since it has complete access to the test set. As a consequence, it produces an optimal ranking (in terms of relevance metrics) based on the candidates returned by the recommender; hence, $f_{seq}^{oracle}$, like the other components, depends on the original set of candidate items to be reranked, it does not simply returns the test set of the user, but the best possible ranking using the candidate items.

Each reranker family is inspired by the three main problems related to data in the form of check-ins (Chen et al., 2016a): standard POI recommendation (independent rerankers), next-POI recommendation (dependent on the last item), and route recommendation (dependent on the whole sequence). Our main hypothesis is that those

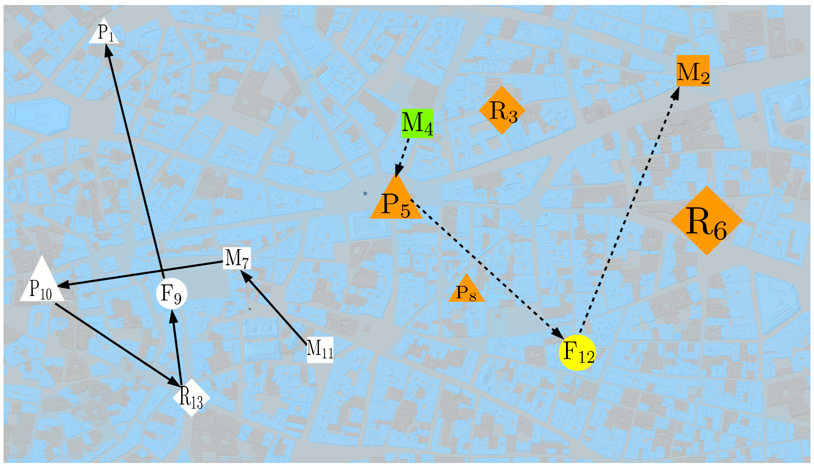
Table 6.2: Summary of the rerankers defined in Section 6.2.2.

| Family | Name | Abbr. | Description |
|---|---|---|---|
| Independent | Random | $f_{seq}^{rnd}$ | Items reranked randomly |
| | Recommender-based | $f_{seq}^{rec}$ | Items reranked according to the score given by a recommender |
| Dependent on the previous item | Distance | $f_{seq}^{dist}$ | Next selected item is the closest one to the previous item in the sequence |
| | Feature-based Markov Chain | $f_{seq}^{feat}$ | Next selected item based on the category that maximizes the transition probability with respect to the category of previous item |
| | Item-based Markov Chain | $f_{seq}^{item}$ | Next item is selected by maximizing the transition probability with respect to previous item |
| Dependent on the whole sequence | LCS-based | $f_{seq}^{lcs}$ | Items reranked by maximizing the LCS between the categories of the recommended items and the user profile |
| | Suffix tree | $f_{seq}^{stree}$ | Items reranked by searching the potential sequence as a substring in the suffix tree built based on the item categories in user profile |
| | Oracle | $f_{seq}^{oracle}$ | Reranked items follow the same order as in the test set |

rerankers that exploit more information about the sequence are the ones that should obtain a higher performance or, in general, should generate more meaningful sequences. This may translate either into better accuracy or by reducing the geographical distance of the recommended routes, since these dimensions incorporate in a natural way the user preferences and the geographical context inherent in the route recommendation problem.

### 6.2.3 Solving ties and coverage issues of reranking strategies

As it might be obvious from the definitions of the reranker components presented in the previous section, some scoring functions may return the same value for different items; a simple example is those rerankers based on features, since every candidate item with the same feature will obtain the same score. Because of that, we propose to consider other characteristics of the items that could improve the user experience when receiving a recommendation of a travel sequence: the item popularity (measured as the number of check-ins received by that POI) and the distance between consecutive venues. Hence, in the experiments, and in order to solve these ties in a deterministic



$f_{seq}^{lcs}$ $M_4 \rightarrow P_5 \rightarrow R_3 \rightarrow P_8$ $f_{seq}^{dist}$ $M_4 \rightarrow P_5 \rightarrow P_8 \rightarrow R_3 \rightarrow M_2 \rightarrow R_6$

$f_{seq}^{stree}$ $M_4 \rightarrow P_5 \rightarrow R_3$ $f_{seq}^{rec}$ $M_4 \rightarrow R_6 \rightarrow P_5 \rightarrow R_3 \rightarrow M_2 \rightarrow P_8$

$f_{seq}^{oracle}$ $M_4 \rightarrow P_5 \rightarrow M_2$

**Figure 6.2:** Comparison of 5 sequences of venues generated by different reranking strategies as presented in Section 6.2.2. Those POIs colored in white form the training set of the user; the orange POIs denote the candidate items to be reranked, that is, the items suggested by a recommender that will be reordered by the presented rerankers. We also show in yellow those POIs that appear in the test set but have not been recommended, together with the starting POI of the test sequence in green. The arrows show the order followed by the user, either based on the training set (solid line) or test set (dotted line). For each item, we show the POI categories (as M, P, F, and R, denoting museums, parks, food, and restaurants) and their ids (as subscripts of the categories), together with their popularity using the marker size (the larger the POI, the more popular it is).

way, we order the subset of candidate items that share the same maximum score by a combination of (inverse) distance and popularity and then by id, both in descending order. It should be noted that, for most of the rerankers, however, sorting by distance was counter-productive and we decided to solve ties based only on popularity; as a consequence, the rerankers based on the whole sequence are the only ones that use both criteria to solve ties.

Furthermore, another problem that may occur when instantiating these rerankers and are applied to real data is that they may have less coverage than the original recommendation list, since some candidate items may obtain no score from the reranker component. To address this issue, we propose the following strategies:

- Filling the rest of the reranked list with the items that could not be scored by the reranker but keeping the order from the original list.

- Filling the rest of the reranked list with the items that could not be scored by the reranker but ordering those items according to some criteria, for instance, by popularity.

- Not filling the list in any way, as a result, some of the candidate items only receive a score from the recommendation algorithm.

Depending on the selected strategy, the result after reranking could be very different, in particular when measuring metrics related to user or item coverage and considering the last strategy, since in that case the reranked list could be (much) shorter than the original recommendation list. Unless stated otherwise, in this chapter we use the first strategy (keeping the original order) to make a fair comparison between the baseline recommenders (without reranking) and those where a reranking strategy (probably, with some coverage problems) has been applied. Therefore, we leave for future work the analysis of the performance of the three strategies in the different rerankers.

### 6.2.4 Further details about reranker components and toy example

In this section, we present specific aspects regarding some of the presented rerankers that should be carefully considered. First, those rerankers based on probabilities ($f_{seq}^{feat}$ and $f_{seq}^{item}$) need to incorporate a smoothing component to avoid zero probabilities (due to sparsity issues of the data); we use the Jelineck-Mercer smoothing that linearly balances the prior with the conditional probability: $p(a|b) = \alpha p_{ml}(a|b) + (1-\alpha)p(a)$, where $p_{ml}$ denotes the maximum-likelihood probability.

Second, the difference between $f_{seq}^{lcs}$ and $f_{seq}^{stree}$ is subtle but important: whereas $f_{seq}^{lcs}$ aims to find those items that produce the Longest Common Subsequence when added to the recommendation list, $f_{seq}^{stree}$ checks whether a given sequence is *exactly* contained in another sequence and finds those items that allow to produce the matching sequences.

In order to help the reader to better understand the differences between the rerankers, we present in Figure 6.2 a visual example of how some of them generate sequences from a set of candidate POIs. In this figure we observe the following candidate items: $M_2$, $M_4$, $R_3$, $R_6$, $P_5$, and $P_8$. Note that $F_{12}$, even though it is in the test set, it was not recommended, so it is not a candidate item for any of the rerankers. The $f_{seq}^{stree}$ reranker obtains the sequence $M_4 \rightarrow P_5 \rightarrow R_3$ as it appears verbatim in the training set (when only the item categories are considered, not the corresponding items with those categories). This is because the suffix tree is built from the visited POIs in the training set (white markers in the figure), thus, after a museum (category M) the suffix tree continues this pattern either with another museum or with a park (category P); in this case, since $P_5$ is more popular than $M_2$, the next POI ranked by this reranker after $M_4$ is $P_5$. Once this strategy has generated the route $M_4 \rightarrow P_5$, a restaurant (category R, and hence, the POI $R_3$) is the only possible candidate based on the training data, since the user always visited a restaurant after a park. Then, this reranker cannot continue the sequence because after a park the suffix tree only accepts food venues, but there is no POI with this feature among the set of candidate

**Table 6.3:** Statistics of the three unprocessed datasets used in the experiments: users ($U$), items ($I$), and number of check-ins ($C_r$) and unique check-ins ($C_{\overline{r}}$).

| Dataset | $\|\mathbf{U}\|$ | $\|\mathbf{I}\|$ | $\|\mathbf{C_r}\|$ | $\|\mathbf{C_{\overline{r}}}\|$ |
|---|---|---|---|---|
| Global-scale check-in dataset | 266,909 | 3,680,126 | 33,263,633 | 15,074,342 |
| Semantic trails | 399,292 | 1,887,799 | 11,910,007 | 8,518,529 |
| Trip builder (photos) | 25,052 | 443,394 | 443,394 | 443,394 |
| Trip builder (clusters/POIs) | 22,611 | 1,349 | 133,089 | 133,089 |

items. In contrast, the $f^{lcs}_{seq}$ reranker is able to add another POI following the Longest Common Subsequence with $\mathrm{P}_8$, since it allows gaps when searching for the subsequence (see Section 4.4). The other sequences obtained by the reranking approaches are more straightforward: they either exploit the popularity (size) of the POIs ($f^{rec}_{seq}$ approach using the Popularity recommender as reranker) or the distance between them ($f^{dist}_{seq}$). Finally, the oracle reranker $f^{oracle}_{seq}$ returns the items in the test set in the order followed by the user, except for $\mathrm{F}_{12}$ because it does not belong to the set of candidate items.

## 6.3 Experiments

### 6.3.1 Datasets

For the experiments, we used three different datasets built from three data sources: the Global-scale check-in dataset from Yang et al. (2016) (already introduced in Section 4.5), the Semantic trails dataset by Monti et al. (2018), and Trip builder used in the work of Brilhante et al. (2013); the first two exploit the Foursquare LBSN, whereas the last one uses photos from Flickr to build the sequences (called trajectories in that work) followed by the users. Table 6.3 shows the main statistics of these datasets as provided by their authors; in the next sections we describe in more detail these datasets and how we used them in our experiments. One key process we performed in these datasets was to only select those interactions related to a particular city, instead of experimenting with all the information as a whole. The rationale for this is that, in many works on POI recommendation, authors tested their experiments on datasets built with check-ins belonging to one specific city (He et al., 2017a, Li et al., 2017b, Liu et al., 2014) because it is a realistic, natural partition of the data; besides, as our plan is to generate routes to the users, it seems reasonable to just use the information of each city independently.

#### Global-scale check-in dataset

This dataset covers more than 33M check-ins on 415 cities in 77 countries covering 18 months of user interactions with Foursquare captured through Twitter; it is available in the author's website[1]. In this dataset, most of the check-ins come from Turkey and

[1]Foursquare dataset, https://sites.google.com/site/yangdingqi/home/foursquare-dataset

**Table 6.4:** Statistics of the four cities used in the experiments. We show the number of users, number of venues, number of check-ins, number of unique check-ins (without repetitions), data density computed according to whether repetitions are considered or not, and number of routes (sessions) found by our framework. We present these values for the entire city, together with the corresponding training and test splits.

| **City** | **Split** | $\|\mathbf{U}\|$ | $\|\mathbf{V}\|$ | $\|\mathbf{C_r}\|$ | $\|\mathbf{C_{\overline{r}}}\|$ | $\frac{\|\mathbf{C_r}\|}{\|\mathbf{U}\|\cdot\|\mathbf{V}\|}\%$ | $\frac{\|\mathbf{C_{\overline{r}}}\|}{\|\mathbf{U}\|\cdot\|\mathbf{V}\|}\%$ | $\|\mathbf{S}\|$ |
|---|---|---|---|---|---|---|---|---|
| New York | Complete | 11,590 | 20,842 | 235,607 | 148,774 | 0.097 | 0.062 | 170,397 |
| | Training | 11,590 | 20,813 | 234,235 | 147,629 | 0.097 | 0.061 | 170,144 |
| | Test | 253 | 815 | 1,372 | 1,322 | 0.665 | 0.641 | 253 |
| Tokyo | Complete | 9,707 | 44,458 | 511,800 | 286,418 | 0.119 | 0.066 | 308,404 |
| | Training | 9,707 | 44,424 | 509,071 | 284,402 | 0.118 | 0.066 | 307,915 |
| | Test | 489 | 1,473 | 2,729 | 2,595 | 0.379 | 0.360 | 489 |
| Rome | Complete | 7,954 | 394 | 61,330 | 49,608 | 1.957 | 1.583 | 27,252 |
| | Training | 7,954 | 394 | 58,201 | 47,358 | 1.857 | 1.511 | 26,778 |
| | Test | 474 | 229 | 3,129 | 2,723 | 2.883 | 2.509 | 474 |
| Petaling Jaya | Complete | 14,858 | 18,385 | 149,904 | 119,760 | 0.055 | 0.044 | 79,418 |
| | Training | 14,858 | 18,302 | 148,689 | 118,831 | 0.055 | 0.044 | 79,165 |
| | Test | 253 | 772 | 1,215 | 1,186 | 0.622 | 0.607 | 253 |

Indonesia, however, to better compare against the state-of-the-art, we decided to select the cities of New York and Tokyo, as they are the most commonly used in the literature and still appeared in the top-10 most checked-in cities in this dataset.

### Semantic trails

In this dataset, the authors integrated different data sources to provide semantically annotated user trails. They did this by starting from the Global-scale check-in dataset, grouping the check-ins into sequences of activities, and then enriching it with semantic information (mainly mapping the Foursquare categories to the Schema.org terms and identifying the cities and countries to their corresponding Wikidata entities).

As described in Monti et al. (2018), the authors applied three filters to remove problematic check-ins: multiple check-ins in a row in the same POI by a user are ignored and only the last one is maintained, those check-ins performed by a user in less than one minute were discarded, and those check-ins that required a speed greater than Mach 1 to move from one venue to the next one were also removed. Then, the remaining check-ins were grouped in trails assuming that two check-ins that are not distant in time more than eight hours belong to the same trail.

It is interesting to note that the authors provide two versions of the dataset, the first one (STD 2013) is mostly based on the Global-scale check-in dataset described before, whereas the second one (STD 2018) is an expanded and updated version of the first one, which is the one we use in this chapter and can be obtained here[2]. In particular,

[2]Semantic Trails dataset, `https://figshare.com/articles/Semantic_Trails_Datasets/7429076`

among all the cities available in the STD 2018 dataset, we decided to select the second one with the highest number of interactions, as the city with more check-ins was Tokyo, already selected in the other dataset. Thus, the selected city was Petaling Jaya (with Wikidata code Q864965). As Petaling Jaya belongs to Kuala Lumpur, most of the check-ins with that identifier also contains check-ins from Kuala Lumpur. However, we decided to maintain the name of Petaling Jaya for the selected city as it is the identification associated with code Q864965. To obtain comparable information with the other cities used in our experiments, we used the Foursquare API to obtain the categories and the coordinates of its POIs, since these are not provided by the authors.

### Trip builder

This dataset, as the previous one, also combines two data sources (Flickr and Wikipedia) and is the only one that does not use Foursquare as the main source for the user interactions. The authors performed the following process for three Italian cities: the Wikipedia pages corresponding to the geographical region of each city were downloaded – it was assumed that each of these pages corresponds to a POI –, these POIs were clustered according to their geographical coordinates – since the authors assumed that a user does not have to take a photo of one POI if she takes a photo of a very close POI –, then, user trajectories were obtained by using Flickr and retrieving the metadata of photos taken in the given cities, finally a matching between the photos and the POIs was performed. The dataset is available in this repository[3].

In our experiments, we selected the largest city among the three provided: Rome. We considered the clusters as the items in the dataset, hence, mapping real sets of venues with an item from the point of view of the recommenders and the evaluation. As we can observe in Table 6.3, by doing this process, the number of items in the system is greatly reduced, which is an important aspect to consider when performing the experiments, since the characteristics of this dataset are very different to the other ones; in fact, probably these items reflect more faithfully interesting POIs from a tourist perspective than those included in the other datasets, since they are mostly limited to neighborhoods, museums, and religious buildings. Finally, a manual mapping had to be performed between the categories provided by the authors (taken from the Wikipedia page and, thus, very specific) and a subset of the first level categories from Foursquare.

### Datasets processing

The three datasets described earlier have been processed to obtain 4 cities where the recommenders will be trained and evaluated: New York (NYC) and Tokyo (TOK) from the Global-scale check-in dataset, Petaling Jaya (PJ) from Semantic trails, and Rome (ROM) from Trip builder. We selected (on purpose) different cities from each dataset so that we could show how the recommendation algorithms work on different types of cities with different inherent characteristics. Moreover, selecting the same city

[3]TripBuilder dataset, `https://github.com/igobrilhante/TripBuilder`

from all the datasets would raise additional problems, namely: not all the datasets contain the same information and, in particular, the same cities, so in order to perform such a comparison we would need to restrict ourselves to the smallest dataset and select the cities available in it, which might be under-represented in the other datasets; moreover, two of the presented data sources take the check-ins from the same LBSN (i.e., Foursquare), hence, by using the same city we would probably obtain the same or very similar results at the end; another issue that we believe is very important is that, as it is standard in the literature, researchers typically use more than one city, usually those well-known or more touristic, such as New York or Tokyo, a criteria we also follow here (as described before).

The steps developed to transform the check-ins included in these datasets into routes or venue sequences match those presented in Section 6.1, and even though we aimed to process all of them in the same way, due to their inherent characteristics some minor changes had to be done either in the value of the parameters or in the followed process. Thus, as the first step (preprocessing data), we performed a 2-core in New York and Tokyo ($p_c = 2$); we also identified and removed those users who made consecutive check-ins in 60 seconds or less more than 3 times for all the cities ($p_b^t = 60, p_b^i = 3$). Besides, for Petaling Jaya and Rome we removed all users with just one interaction. The second step (building routes) was only performed on the first dataset, since we assume the trails and trajectories included in the other datasets are valid; hence, in New York and Tokyo we build the venue sequences so that the difference between consecutive POIs is less than 8 hours ($b_t = 8, b_g = \infty$). Finally, regarding the third step (filtering sequences), we did not impose any constraint on this (hence, $f_m = 0, f_M = \infty, f_u = 0, f_i = 0$), although we used some of these constraints when creating the training-test splits, as explained in the next section.

### 6.3.2 Evaluation methodology

As discussed at the end of Section 6.1, there are different possibilities to split sequential data. We decided to build the test set with the last route identified for each user in order to have more control and experiment with a large number of users ($t_n = 1$). Additionally, we impose the following constraints to the users so the recommenders are tested on enough training and test data: only users with a minimum of 3 routes and at least 4 venues in the last route are included in the test set, all the other information is kept in the training split. The statistics of the four cities with their corresponding training and test sets once this methodology is applied can be seen in Table 6.4. Note that due to these constraints, the number of users and items in the resulting test sets is much lower than in the complete or training sets, and, hence, the density of these subsets is remarkably higher; however, these values are only included for completeness, since no recommendation algorithm is trained using the test data and, hence, the density of the test set has no effect on this aspect.

Some recommenders need item attributes such as the categories and geographical coordinates of the venues. For the cities with a Foursquare id (New York, Tokyo, Petaling Jaya) we take this information from Foursquare, using the categories of level

1 unless stated otherwise; for Rome, we use the information available in the dataset and a manual mapping for the categories to obtain a comparable number of categories across the four cities, as described in the previous section.

Finally, due to the intrinsic nature of the user-item interactions in this domain, we may have repetitions in both splits (users who have visited the same POI more than once, so that they may appear both in training and test set and also more than once in the training or test splits). As some recommenders may not deal well (or their behavior is not defined) when this happens, we create two different training sets:

- Aggregated: we aggregate all the check-ins where a user interacted with the same venue by assigning the number of times a user checked-in in the POI as the preference value and the first timestamp of the repetitions as its timestamp.
- Not aggregated: we leave the training set with the repetitions.

As we shall show in Section 6.3.4, depending on the assumptions of the recommendation algorithms, we use one or the other training set. We do not change the test set because it does not affect the recommenders but their evaluations, and the proposed sequential metric (described in Section 4.4) is able to deal with repeated items.

Unless stated otherwise, the results are shown using the TrainItems methodology (Said and Bellogín, 2014), that is, the candidate POIs are the ones appearing in the training set that the target user has not visited before. More specifically, all recommenders rank all the possible items, although they have been configured to store only the top-100 items for each user; this parameter has no effect in the reported results since we always report lower cutoffs. Furthermore, since some recommenders need an starting POI to build the sequence, in order to make a fair comparison across all the recommenders, the first POI in the test sequence of each user is also provided to every recommendation algorithm, as it is often done in the literature (Kumar et al., 2017, Monti et al., 2018, Zhao et al., 2018b).

### 6.3.3 Evaluation metrics

To analyze the performance of the recommenders, we use metrics oriented at measuring different dimensions like accuracy (relevance), novelty, and diversity; additionally, since we deal with potentially real recommendations, we want to measure the distance of the obtained route, to assess how realistic such recommendations might be[4]. For relevance, we show the results using Precision (P) and nDCG, together with their sequential counterparts: $\mathrm{P_s}$ and $\mathrm{nDCG_s}$, according to the sequential metrics defined in Section 4.4. On the other hand, for novelty and diversity, we show the results in terms of EPC and

[4]While it is true that a lower distance between recommended items is not necessarily requested by the user, since it might depend on the actual context of the user (such as the city type or the possibility of driving a car), in the POI recommendation literature it is common to exploit closeness between points as a source of information for the algorithms (Miller, 2004), even though very few works have explicitly presented experimental results based on this dimension. However, one of our main hypotheses is that the geographical dimension is critical and should be minimized when recommending realistic routes.

Gini. We also report the distance (denoted by Dist) in Km as the sum of the distance between each venue in the recommendation list and the next one. Except for the distance metric, higher values indicate better results (i.e., recommendations are more relevant/novel/diverse).

Furthermore, because of the high sparsity in this domain and the difficulty to match the exact same venue the user visited in the test set, some authors report accuracy metrics but at the category level instead of the item level, this means that a metric like precision, for example, would measure how many categories that appear in the test set are recommended by the algorithm, or other variations, such as measuring the likelihood that the recommended categories would be produced at random (perplexity), or measuring the interest of a user in a recommended tour based on the time she spends on venues that belong to those categories (He et al., 2017a, Brilhante et al., 2013, Palumbo et al., 2017, Lim et al., 2015). According to this, we define the Test Feature Precision (TFP) that takes into account the features of the POIs that we retrieved correctly but each category is only taken into account based on the number of categories of each type available in the test set, so, for instance, if the recommended list consists of 3 museums and in the test set there are only 2, only 2 of them will be used in the computation of the metric.

It is worth noting that, by capping the maximum number of times each feature can be considered in the whole list (according to the frequency such feature appears in the test set), this metric could also measure to some extent the diversity on the recommended features, since once it saturates the metric value (because the maximum number of items with a specific feature has been reached), then recommending such feature will not improve any more the value of the metric, which is similar to how some diversity metrics are defined (Castells et al., 2015), although considering a more extreme user behavior: one where the user abandons the ranking list once too many items with a specific feature have been examined.

Additionally, and based on the sequential metrics defined in Section 4.4, we include in our evaluation a category-based sequential metric that considers the correct order of the features according to the sequence followed by the user in the test set; it is computed as $\mathrm{P_s}$ but matching categories instead of POIs, we call it Sequential Feature Precision or $\mathrm{FP_s}$. We note that this is one of the few works where both types of metrics (either matching by category or by item) are shown and reported together; moreover, this is the first time that sequentiality has been incorporated into the category-based precision.

### 6.3.4 Recommenders

We experiment with 28 algorithms, covering different types and information sources, because of this, we decided to group them in the following 6 families: Basic, Classic, Temporal, Geo, Tour, and Skylines. We include standard methods in the Recommender Systems literature and others more oriented to the contexts of venue and route recommendation. However, we should mention that we were not able to apply some of the most typical solutions to the Tourist Trip Design Problem (TTDP) since they need

**Table 6.5:** Parameters of evaluated recommenders; the values that are not between the symbols {} are considered fixed and not tuned. Third column indicates the abbreviation used in tables and figures. Fourth column shows whether the recommender can use an aggregated training set (Y) or one where repetitions are allowed (N).

| **Family** | **Recommender** | **Abbr.** | **Agg?** | **Parameters** |
|---|---|---|---|---|
| Basic | Random | Rnd | Y | None |
| | Popularity | Pop | Y | None |
| | Training | Train | Y | None |
| | Reverse training | $\overline{\text{Train}}$ | Y | None |
| Classic | Content-based | CBUI | Y | Wgt = Yes |
| | Content-collaborative filtering | CBCF | Y | $k = \{40, 60, 80, 100, 120\}$, Wgt = {Yes, No} |
| | Item-based nearest neighbor | IB | Y | $k = \{40, 60, 80, 100, 120\}$, sim = {SetJ, VecC } |
| | User-based nearest neighbor | UB | Y | $k = \{40, 60, 80, 100, 120\}$, sim = {SetJ, VecC } |
| | MF using Alternate Least Squares | HKV | Y | $k = \{10, 50, 100\}$, $\alpha = \{0.1, 1\}$, $\lambda = \{0.1, 1\}$ |
| | MF with Bayesian Personalized Ranking | BPRMF | Y | $k = \{10, 50, 100\}$, $\lambda_u = \lambda_i = \{0.001, 0.0025, 0.005, 0.01, 0.1\}$, $\lambda_0 = \{0, 0.5, 1\}$, $\lambda_j = \lambda_u/10$, iter = 50 |
| Temporal | Temporal Popularity | TPop | Y | None |
| | User-based time decay | TD | Y | $k = \{40, 60, 80, 100, 120\}$, sim = {SetJ, VecC }, $\lambda = \{0.05, 0.1\}$ |
| | Backward-Forward (No LCS) | BFUB | N | $k = \{40, 60, 80, 100, 120\}$, sim = {SetJ, VecC }, Wgt = {T, F }, Nrm = {Def, Std }, $L_m^- = L_m^+ = \{5, 10\}$ |
| | Backward-Forward (LCS) | BFsUB | N | $k = \{40, 60, 80, 100, 120\}$, sim = {$\text{sim}_1$}, Wgt = {T, F }, Nrm = {Def, Std }, $L_m^- = L_m^+ = \{5, 10\}$ |
| | Markov Chain | MC | N | $k = \{2, 5, 10, 20\}$, $\lambda = \{0.1, 0.2\}$ |
| | Factorized Personalized Markov Chain | FPMC | N | $k = \{2, 5, 10, 20\}$, $\lambda = \{0.1, 0.2\}$ |
| | Factorized Sequences with Item Similarities | Fossil | N | $k = \{2, 5, 10, 20\}$, $\lambda = \{0.1, 0.2\}$, $L = \{1, 2, 3\}$ |
| | Convolutional Neural Network | Caser | N | T = {1, 2}, d = {10, 50}, nh = {4, 16}, L = 2, n_iters = 30, l_rate = 0.003, nv = 4, drops = 0.5, ac_convs = relu, ac_fcs = relu, batch_size = 512, $l_2 = 10^{-6}$ |
| Geo | Average Distance | AvgDis | Y | None |
| | Kernel Density Estimation | KDE | Y | None |
| | Pop + UB + AvgDis | PGN | Y | $k = \{40, 60, 80, 100, 120\}$, sim = {SetJ, VecC } |
| | Instance-Region Neighborhood MF | IRenMF | Y | $k = \{50, 100\}$, $\alpha = \{0.4, 0.6\}$, $\lambda_3 = \{0.1, 1\}$, Clusters = {50, 5}, $\lambda_1 = \lambda_2 = 0.015$, GeoNN = 10, Factors = 100, $\alpha = 10$ |
| | Ranking Geographical Factorization | RankGeoFM | Y | $k = \{50, 100\}$, $\alpha = \{0.1, 0.2\}$, n = {10, 50, 100, 200}, C = 1, $\epsilon = 0.3$, iter = 120 |
| Tour | Closest by distance | DistNN | Y | None |
| | Feature Markov Chain | FeatMC | N | Smooth = {None, JM (0.1), JM (0.5), JM (0.9)} |
| | Item Markov Chain | ItemMC | N | Smooth = {None, JM (0.1), JM (0.5), JM (0.9) } |
| Skylines | TestOrder | TestOrder | N | None |
| | TestOrder Reverse | $\overline{\text{TestOrder}}$ | N | None |

additional data that is not easily available in public datasets, such as the price and schedule of the venues, although we plan to collect this information and extend the comparison in the future. In any case, we aimed at providing an exhaustive comparison of techniques using different data – to avoid biases in the results – and provide a wide perspective of how each of these techniques may perform in the real world, by considering a realistic evaluation methodology, including both the metrics and how the training-test splits was performed.

Most of the recommenders that we have used in these experiments have already been explained in the experiments of our novel approaches for evaluation (Section 4.5) and our backward-forward $k$-NN reformulation (Section 5.3). For that reason, below we will only explain in detail the algorithms that have not been previously mentioned in previous experiments (i.e., the algorithms from the Geo and Tour families).

- Basic: simple baselines, useful to test biases on the recommendations:
  - Rnd: a random recommender. Already explained in Section 4.5.1.
  - Pop: a popularity recommender. Already explained in Section 4.5.1.
  - Train: recommender that suggests the items already interacted by the user in the training set. Already explained in Section 4.5.2.
  - $\overline{\text{Train}}$: same as the Train recommender but using an inverse ordering of the items; our own implementation.
- Classic: family of classic recommenders using a collaborative filtering or a content-based component but, in any case, time and geographical agnostic:
  - CBUI: a pure content-based recommender using a Vector Space Model where users are represented as an aggregation of the features corresponding to the items she visited, and every item as a combination of those users that interacted with each item (de Gemmis et al., 2015). For the users, we allow each item feature to have a weight proportional to the number of times the user visited that item (parameter Wgt); our own implementation.
  - CBCF: the Collaborative-via content hybrid recommender. Already explained in Section 4.5.1.
  - UB: a user-based nearest neighbor algorithm. Already explained in Section 2.2.2.
  - IB: an item-based nearest neighbor algorithm. Already explained in Section 2.2.2.
  - HKV: a matrix factorization algorithm. Already explained in Section 2.2.2.
  - BPRMF: Bayesian Personalized Ranking using a matrix factorization approach. Already explained in Section 2.2.2.
- Temporal: recommenders exploiting the temporal or sequential information of the items (using the item timestamps or its ordering in the user profile):
  - TPop: similar to the Pop recommender, but the item popularity computation also involves penalizing old check-ins by applying the min-max normalization in the timestamps of the training set. It is based on the freshness models presented in Section 4 (Sánchez and Bellogín, 2018b); our own implementation.
  - TD: time-weighting user-based neighbor algorithm. Already explained in Section 4.5.1.
  - BFUB: our backward-forward approach explained in Chapter 5, with no sequential similarity.
  - BFsUB: our backward-forward approach explained in Chapter 5 with a sequential (LCS) similarity.

  - MC: Factorized Markov Chains explained in Section 4.5.1. Implementation provided by He and McAuley (2016).
  - FPMC: Factorized Personalized Markov Chains explained in Section 2.3, proposed in Rendle et al. (2010). Implementation provided by He and McAuley (2016).
  - Fossil: Factorized Sequential Prediction with Item Similarity Models explained in Section 2.3 and proposed in He and McAuley (2016).
  - Caser: Convolutional Sequence Embedding Recommendation Model explained in Section 2.3 and proposed in Tang and Wang (2018).

- Geo: family of venue recommenders that exploit the geographical influence component of the POIs:
  - AvgDis: algorithm that suggests the closest POIs to the user's average location, the average is computed by calculating the midpoint of the coordinates of the visited POIs in the user profile; our own implementation.
  - KDE: the geographical influence component from (Zhang et al., 2014), it models a probability distribution over a two-dimensional space (latitude and longitude) using Kernel Density Estimation for every user; our own implementation.
  - PGN: a hybrid POI recommendation algorithm that combines the UB, Pop, and AvgDis recommenders (giving the three recommenders the same weight); our own implementation.
  - IRenMF: weighted Matrix Factorization method proposed in (Liu et al., 2014) that also exploits the geographical influence between neighbor venues; implementation provided by other authors[5].
  - RankGeoFM: Matrix Factorization method using BPR as proposed in (Li et al., 2015a), this method exploits the geographical influence between neighbor POIs, but it also maintains an additional latent matrix to model the user geographical preferences; our own implementation based on the one available in LibRec[6].

- Tour: recommenders that explicitly aim to recommend a route or a sequence of POIs optimizing different criteria:
  - DistNN: it selects the next item in the sequence by minimizing the distance between the current venue and the next one; our own implementation.
  - FeatMC: it selects the next item in the sequence by maximizing the transition probability between the features of the current and candidate POIs, if there

[5]Liu et al. (2014) provide an implementation of this technique that is used in their experimental comparison, available at `http://spatialkeyword.sce.ntu.edu.sg/eval-vldb17/`.

[6]A Leading Java Library for Recommender Systems, available at `https://www.librec.net` and described in Guo et al. (2015b).

**Table 6.6:** Optimal parameters found in each city, see Table 6.5 for the explored range of the parameters.

| **Rec** | **New York** | **Tokyo** | **Rome** | **Petaling Jaya** |
|---|---|---|---|---|
| CBCF: k, Wgt | 100, Yes | 120, No | 120, No | 120, No |
| IB: k, sim | 60, SetJ | 120, SetJ | 100, SetJ | 100, SetJ |
| UB: k, sim | 100, SetJ | 120, VecC | 120, SetJ | 120, SetJ |
| HKV: k, $\alpha$, $\lambda$ | 50, 0.1, 1 | 10, 0.1, 0.1 | 10, 0.1, 1 | 10, 1, 1 |
| BPRMF: k, $\lambda_u = \lambda_i$, $\lambda_0$ | 100, 0.005, 0 | 50, 0.005, 0 | 100, 0.1, 0 | 50, 0.1, 0 |
| TD: k, sim, $\lambda$ | 120, SetJ, 0.1 | 120, VecC, 0.1 | 120, VecC, 0.05 | 120, VecC, 0.05 |
| BFUB: k, sim, Wgt, Nrm, $L_m^- = L_m^+$ | 120, SetJ, F, Def, 10 | 100, VecC, F, Def, 5 | 120, SetJ, T, Def, 10 | 120, VecC, T, Def, 5 |
| BFsUB: k, Wgt, Nrm, $L_m^- = L_m^+$ | 100, F, Def, 10 | 120, F, Def, 10 | 100, F, Def, 10 | 80, T, Def, 10 |
| MC: k, $\lambda$ | 2, 0.1 | 10, 0.1 | 2, 0.2 | 20, 0.2 |
| FPMC: k, $\lambda$ | 2, 0.2 | 10, 0.2 | 20, 0.2 | 20, 0.1 |
| Fossil: k, $\lambda$, $L$ | 10, 0.2, 3 | 20, 0.1, 2 | 10, 0.2, 2 | 2, 0.1, 1 |
| Caser: $T$, d, nh | 1, 50, 4 | 1, 10, 4 | 2, 10, 4 | 1, 50, 16 |
| PGN: k, sim | 100, SetJ | 100, VecC | 100, VecC | 100, SetJ |
| IRenMF: k, $\alpha$, $\lambda_3$, Clusters | 50, 0.6, 1, 50 | 100, 0.6, 1, 50 | 50, 0.6, 1, 5 | 50, 0.4, 0.1, 5 |
| RankGeoFM: k, $\alpha$, n | 100, 0.1, 100 | 100, 0.1, 10 | 100, 0.1, 200 | 100, 0.1, 200 |
| FeatMC: Smooth | JM (0.9) | JM (0.1) | JM (0.1) | JM (0.5) |
| ItemMC: Smooth | JM (0.1) | JM (0.5) | JM (0.5) | JM (0.1) |

are several items with the same probability, they will be sorted by popularity; our own implementation.

- ItemMC: it selects the next item in the sequence by maximizing the transition probability between POIs by counting how often any user checked-in in the two POIs (the current one and any of the candidates) one after the other; our own implementation.

- Skylines: oracle recommenders that recommend the test set, they are useful to analyze the maximum values that a recommender can achieve, at least in terms of accuracy metrics:

  - TestOrder: method that returns the test set for every user ordered by ascending timestamp (mirroring the order followed by the user according to the test set); since the test set may contain repeated venues, this algorithm only recommends each item once (based on its first timestamp) to be more comparable to the rest of the algorithms; our own implementation. Note that additional constraints could apply to mimic the behavior of the other recommenders being compared, for instance, by limiting the candidate items to those in the training set, as described in Section 6.3.2.

  - $\overline{\text{TestOrder}}$: same as the TestOrder but the items are ordered by descending timestamp (i.e., the route followed by the user in the test set is reversed); our own implementation.

We present in Table 6.5 the range of the parameters tested in the experiments, whereas Table 6.6 shows the best parameters found for each recommender in every city. The optimal parameters were selected according to the performance obtained using the nDCG$_s$@10 evaluation metric. We used a larger cutoff because in this case we are comparing routes, not independent items. Besides, a recent article argued that, in general, metrics with larger cutoffs tend to be more stable than metrics with smaller cutoffs (Valcarce et al., 2020).

Note that we also include in the table whether the recommender uses an aggregated training or not (as explained in Section 6.3.2); we based this decision according to the nature of each algorithm and, in some cases, because better results were obtained for the reported configuration.

### 6.3.5 Performance of venue recommender systems

Table 6.7 shows the performance for each recommender in the city of New York where all metrics are computed at cutoff 10 except the distance metric at cutoff 5. We decided to present the results in this way because most sequences in test have less than 5 POIs (see ratio between $|\mathbf{C_r}|$ and $|\mathbf{S}|$ in Table 6.4, which is between 4.8 and 6.6 for all the test sets) and, thus, computing the distance of routes for the first 10 recommended venues would produce a less realistic situation and not fair when compared against the skylines. This effect will be further analyzed in Section 6.4.

The first thing we observe in this table is that the Skylines are not achieving a perfect score (i.e., 1.0) in terms of relevance metrics such as nDCG or Precision. This is because in our experimental setting we simulate a realistic situation, where all recommenders – including the Skylines – can only recommend those venues that appear in the training set, so those items that only appeared in the test set cannot be recommended; in this way, the comparison between the Skylines and the rest of algorithms is fair, since they all consider the same set of candidate items. Besides, it should also be noted that we are measuring performance at cutoff 10 but many users have less items in their test set, which impacts the value of the evaluation metrics. Moreover, since the test set may contain repeated items for some users, the length of the test routes could decrease even more, which, together with the fact that none of the evaluated recommenders are allowed to suggest the same item more than once, may amplify such effect (see Section 6.4 for more details).

#### Analysis based on evaluation dimensions: distance, sequential and non-sequential metrics, and novelty and diversity

As we observe in the table, the total distance of the recommended venues is very low for the Skylines (which actually reflects the distance traveled by users in the test set), and specifically, much lower than most of the other recommenders, whose distances range from 20 to 40 Km (demonstrating that users tend to prefer shorter routes instead of very long ones and, thus, validating our hypothesis that geographical distance is an important dimension to be minimized in route recommendation). However, as evidenced by the poor performance in terms of relevance of DistNN, AvgDis, and KDE, it is a challenge to produce short, but interesting routes. This is one of our main motivations to integrate reranking strategies into route recommendation, where not only relevance, but other alternative criteria, ought to be maximized at the same time.

**Table 6.7:** Performance for the city of New York. All metrics presented are computed at cutoff 10, except Dist at cutoff 5. They are grouped in three categories: accuracy (P, nDCG, TFP), sequential accuracy ($P_s$, $nDCG_s$, $FP_s$), and non-accuracy (Dist, Gini, EPC). In bold, we show the best recommender in each family, a † is used to emphasize the best one for that metric without considering the skylines, whereas ▲ is used when the skylines are also considered.

| | | | Accuracy | | | Seq. Accuracy | | | Non-Accuracy | |
|---|---|---|---|---|---|---|---|---|---|---|
| Family | Rec | P | nDCG | TFP | $P_s$ | $nDCG_s$ | $FP_s$ | Dist | Gini | EPC |
| Basic | Rnd | 0.100 | 0.354 | 0.324 | 0.100 | 0.345 | **0.290** | **32.1** | ▲**0.104** | **0.997** |
| | Pop | **0.144** | **0.417** | **0.326** | **0.139** | **0.402** | 0.284 | 43.9 | 0.001 | 0.904 |
| Classic | CBUI | 0.100 | 0.354 | 0.310 | 0.100 | 0.346 | 0.294 | 32.0 | **0.080** | ▲**0.998** |
| | CBCF | 0.127 | 0.395 | 0.301 | 0.126 | 0.385 | 0.284 | 38.2 | 0.012 | 0.945 |
| | IB | 0.121 | 0.383 | 0.326 | 0.119 | 0.373 | 0.298 | **24.2** | 0.047 | 0.971 |
| | UB | 0.139 | 0.409 | **0.340** | 0.136 | 0.396 | **0.300** | 40.8 | 0.008 | 0.933 |
| | HKV | 0.121 | 0.382 | 0.323 | 0.120 | 0.372 | 0.289 | 35.8 | 0.004 | 0.949 |
| | BPRMF | **0.145** | **0.418** | 0.330 | **0.141** | **0.404** | 0.285 | 45.3 | 0.001 | 0.905 |
| Temporal | TPop | **0.145** | 0.418 | 0.321 | **0.140** | **0.404** | 0.284 | 43.2 | 0.001 | 0.904 |
| | TD | 0.135 | 0.405 | 0.342 | 0.131 | 0.391 | 0.300 | 40.6 | **0.011** | **0.937** |
| | BFUB | 0.142 | 0.417 | †**0.349** | 0.138 | 0.402 | **0.307** | 41.8 | 0.004 | 0.926 |
| | BFsUB | 0.145 | **0.420** | 0.336 | **0.140** | **0.404** | 0.302 | 42.4 | 0.003 | 0.920 |
| | MC | 0.140 | 0.413 | 0.314 | 0.138 | 0.401 | 0.284 | 43.9 | 0.002 | 0.912 |
| | FPMC | 0.138 | 0.406 | 0.323 | 0.136 | 0.395 | 0.294 | **16.1** | 0.002 | 0.932 |
| | Fossil | 0.137 | 0.405 | 0.332 | 0.135 | 0.394 | 0.300 | 29.4 | 0.002 | 0.919 |
| | Caser | 0.138 | 0.408 | 0.339 | 0.136 | 0.396 | **0.307** | 35.3 | 0.005 | 0.934 |
| Geo | AvgDis | 0.100 | 0.354 | 0.305 | 0.100 | 0.346 | 0.281 | 3.3 | 0.075 | **0.997** |
| | KDE | 0.101 | 0.354 | 0.300 | 0.101 | 0.346 | 0.278 | **3.1** | **0.087** | 0.997 |
| | PGN | 0.133 | 0.405 | 0.336 | 0.132 | 0.394 | 0.300 | 46.3 | 0.017 | 0.927 |
| | IRenMF | †**0.147** | †**0.420** | **0.345** | †**0.143** | †**0.405** | 0.306 | 43.9 | 0.002 | 0.916 |
| | RankGeoFM | 0.130 | 0.394 | 0.334 | 0.127 | 0.382 | †**0.308** | 18.0 | 0.013 | 0.956 |
| Tour | DistNN | 0.109 | 0.367 | **0.311** | 0.107 | 0.356 | **0.288** | ▲**0.1** | **0.065** | **0.997** |
| | FeatMC | 0.118 | 0.382 | 0.206 | 0.117 | 0.372 | 0.206 | 18.4 | 0.002 | 0.972 |
| | ItemMC | **0.132** | **0.404** | 0.295 | **0.129** | **0.391** | 0.279 | 44.9 | 0.001 | 0.911 |
| Skylines | $\overline{\text{TestOrder}}$ | ▲**0.453** | ▲**0.928** | ▲**0.453** | 0.205 | 0.569 | 0.335 | 11.6 | **0.023** | **0.979** |
| % | TestOrder | ▲**0.453** | ▲**0.928** | ▲**0.453** | ▲**0.453** | ▲**0.909** | ▲**0.453** | **8.3** | **0.023** | **0.979** |

When comparing P and nDCG metrics against their sequential counterparts ($P_s$ and $nDCG_s$) – hence comparing, to some extent, the performance of the algorithms on the venue recommendation task against that on route recommendation –, we find that the sequential metrics always achieve a value lower or equal than the standard metric result. This makes sense since these metrics penalize those POIs that have not been returned in the exact order with respect to the test set of the user (as mentioned in Section 4.4); however, most of the obtained differences are around 0.01 below the value of the non-sequential metric. To further analyze this issue, in Figure 6.3 we show the number of users with a specific difference between the values of P and $P_s$ (left image) and TFP and $FP_s$ (right image) for the best performing approach, which corresponds to the IRenMF algorithm; since we contrast this difference against the number of relevant items/features returned, it is obvious that the sequential variations of the metrics have more margin to affect the final result when the algorithms return more than 1 or 2 relevant items/features. The number of potential relevant items/features recommended depends, on the other hand, on the test size of each user. In other set

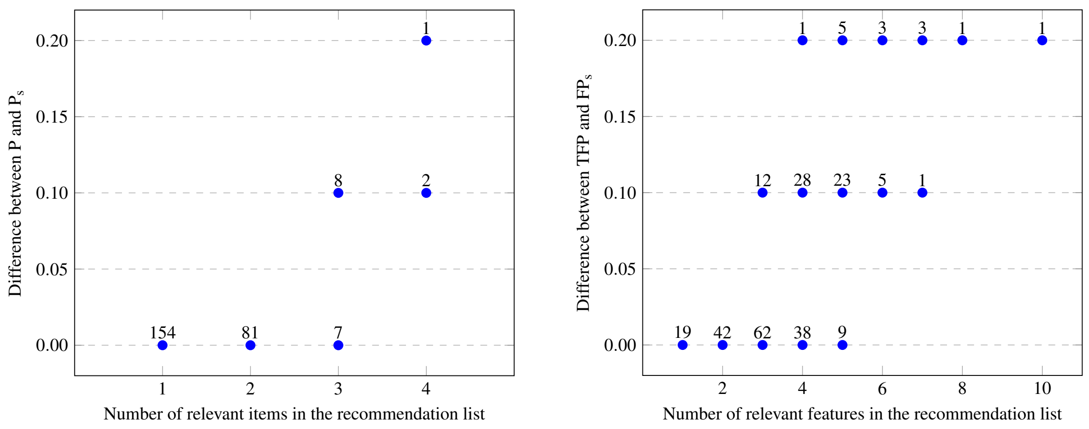


**Figure 6.3:** Difference between sequential ($P_s$ and $FP_s$) and non-sequential (P and TFP) metric values and the number of users with such difference, depending on the number of relevant items or features returned by IRenMF in New York.

of experiments (not reported for lack of space) we observe a similar situation: the larger the test size of the users, the larger the differences between sequential and non-sequential measurements. Therefore, the behavior we observe here is caused by the inherent properties of these datasets and this particular recommendation task which, as we discussed at the beginning of the paper, is very sparse; in fact, for features, where sparsity is lower, the differences tend to be higher.

Independently of the previous observation, larger differences between the sequential and non-sequential metrics are obtained for Skylines, in particular between $\overline{\text{TestOrder}}$ and TestOrder, since all the items returned by both are relevant but the former receives a strong penalty for returning the test items in the reverse order. As expected, non-sequential metrics obtain the same value for either Skyline, evidencing their lack of sensitivity to the order of the recommendation list, since they only account for the number of relevant items (which is the same in both Skylines, since every recommended item is relevant).

These results evidence that sequential metrics work as expected, but, at the same time, they also show that it is very difficult to recommend interesting venues for users that are at the same time presented in the correct order, or, in other terms, to optimize at the same time for the venue and route recommendation tasks. Because of this, when we analyze the TFP and $FP_s$ metrics we observe that, in general, it is easier to recommend fitting categories than actual POIs. Moreover, when considering the feature relevance, the sequences give us more information: for example, even though the Pop recommender has a high value in TFP, its performance in $FP_s$ decreases considerably. We want to highlight that this is the first work – to the best of our knowledge – where such comparison has been made. Additionally, it should be noted that it is straightforward to compare $FP_s$ and $P_s$ since in the ideal case (Skylines) their performance is the same, and in any other case, they are computed very similarly, although one finds matches at the category level and the other at the item level. According to this, we find some examples (such as UB and BFUB) where the recommendation algorithm is the best in terms of $FP_s$ but it is not as good in terms of $P_s$.

Now, regarding novelty and diversity (EPC and Gini), we notice a general trend where the recommenders that perform the worst in terms of relevance, they tend to be the best in terms of these dimensions (as we have already show in Section 4.5). This is related to the classical tradeoff between relevance and novelty/diversity (Vargas and Castells, 2011), but it is also related to how prone each recommendation algorithm is to recommend popular items (Jannach et al., 2015), since the novelty metric penalizes the most popular venues; diversity behaves in a similar way, since if we are recommending the same (popular) venues for all the users, the overall diversity of the system decreases.

**Behavior of evaluated recommenders**

According to the presented results in Table 6.7, we observe that the geographical and temporal recommenders are normally the ones that achieve the best results in terms of pure relevance metrics (P, $P_s$, nDCG, $nDCG_s$). Even though this might be reasonable, there are two important aspects to analyze: the first one is the strong popularity bias observed in these datasets, as this baseline is able to beat most of the recommenders (both Pop and TPop). This effect, although common in RS, is even more pronounced in the POI recommendation domain due to the high sparsity, as other baselines find it difficult to exploit the preferences of users. Also, in relation to this effect, it can be observed that algorithms using neighbors for collaborative similarity need to consider large neighborhoods (a large value for the parameter $k$, see Table 6.6 where only in two cases it is under 100), indicating that they need to exploit more information to make recommendations. In fact, as discussed by some authors recently, when nearest neighbor recommenders increase the number of neighbors, they tend to get closer to popularity, and hence, their popularity bias is stronger (Cañamares and Castells, 2017).

The second aspect to consider is related to the tour recommenders as their performance is rather low (except for ItemMC). One possible explanation for this is that taking into account the distance between items or only how frequent users go from one venue category to another is a rather incomplete heuristic (in fact, we also observe a similar effect in the AvgDis and KDE recommenders, belonging to the Geo family) and more information should be exploited in combination. More specifically, we need to consider that some of these algorithms do not work with any kind of collaborative information, just with the coordinates of the POIs the users have visited, and even though users tend to go to venues that are close to each other (as the Dist metric shows) maximizing only this component may not reflect their interests as a whole. However, when we combine the geographical component with other features like collaborative information (as in RankGeoFM, IRenMF, and PGN), the performance increases considerably.

On the other hand, it is interesting and somewhat surprising that sequential approaches like MC, FPMC, Fossil, and Caser are not able to obtain higher results than other, more simple models. The reason for this may be that these sequence-aware models are defined and formulated to consider sequences when training the preference data from users, but not to generate interesting sequences for the users, in the sense of items being consumed in sequence. That is, these models are focused on predicting the next venue, not the whole route or sequence of venues; because of this, it is not so strange that these algorithms do not perform as well as expected, or that other simple techniques (such as those based on popularity) could obtain better results. Additionally, we hypothesize this might be due to the high sparsity of our POI recommendation datasets in contrast with the original papers, where the authors filtered out users with a relatively high number of interactions – hence, increasing the data density (e.g., for Caser, the authors removed users with less than 15 and 10 interactions when working with Gowalla and Foursquare respectively).

From the Temporal family, only BFUB and BFsUB (recall that this the sequential $k$-NN presented in Chapter 5 with a classic and a sequential similarity, respectively) are able to get a performance close to the TPop recommender in terms of relevance and BFUB is the best one in the TFP metric (slightly better than the IRenMF approach), illustrating that it is possible to improve the performance of simple models like UB if we combine them with sequential components. In relation to the Classic family, we also find large differences between the two MF approaches – HKV and BPRMF– as the results for the latter are in general much better. This can be explained by the way they create the models and the assumptions they make: while HKV uses the score of the user in the minimization formula, BPRMF focuses on optimizing the ranking (why some items are consumed and why others are not), the most appropriate approach in this type of situations where no ratings or explicit scores are available.

Finally, it is also interesting to observe that the IRenMF model obtains better results than RankGeoFM. Although this contradicts the work of Liu et al. (2017), this can be due to differences in the evaluation methodology, more specifically, in that work the authors considered entire datasets, without dividing them into cities, which could confuse geographical methods like these; moreover, heavy filters were applied to those datasets, in particular, in the Foursquare dataset all users and POIs with less than 10 interactions were removed, reducing its sparsity in comparison with the ones we use here. Additionally, the implementation of these techniques could be slightly different with respect to those tested in that paper, since while for IRenMF we have taken the implementation as provided by the authors, for RankGeoFM we adapted the implementation provided by the LibRec library (see Section 6.3.4).

#### Analysis on other cities

So far, we have only explored the results for the city of New York; now in Table 6.8 we summarize the results for the best recommender in each family (according to nDCG$_s$) for New York and present the same results for Rome and Petaling Jaya, in this way showing one city from each of the three datasets used. In the Appendix A.2.1, we show and discuss the results for all the cities described in Section 6.3.1, but in the rest of this chapter we shall focus on these three cases for the sake of space.

There are some interesting similarities between the results for these cities, which are, in principle, very different (culturally but also regarding the data collected, according to Table 6.4). First, the BPRMF recommender is the best one for the Classic

**Table 6.8:** Performance for New York (NYC), Rome (ROM), and Petaling Jaya (PJ), only showing the best recommender for each family. Notation and cutoffs like in Table 6.7.

| | | | | Accuracy | | | Seq. Accuracy | | | Non Accuracy | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| City | Family | Rec | P | nDCG | TFP | $P_s$ | $nDCG_s$ | $FP_s$ | Dist | Gini | EPC |
| NYC | Basic | Pop | 0.144 | 0.417 | 0.326 | 0.139 | 0.402 | 0.284 | 43.9 | 0.001 | 0.904 |
| | Classic | BPRMF | 0.145 | 0.418 | 0.330 | 0.141 | 0.404 | 0.285 | 45.3 | 0.001 | 0.905 |
| | Temporal | BFsUB | 0.145 | †**0.420** | 0.336 | 0.140 | 0.404 | 0.302 | †**42.4** | †**0.003** | †**0.920** |
| | Geo | IRenMF | †**0.147** | †**0.420** | †**0.345** | †**0.143** | †**0.405** | †**0.306** | 43.9 | 0.002 | 0.916 |
| | Tour | ItemMC | 0.132 | 0.404 | 0.295 | 0.129 | 0.391 | 0.279 | 44.9 | 0.001 | 0.911 |
| | Skylines | TestOrder | ▴**0.453** | ▴**0.928** | ▴**0.453** | ▴**0.453** | ▴**0.909** | ▴**0.453** | ▴**8.3** | ▴**0.023** | ▴**0.979** |
| ROM | Basic | Pop | 0.227 | 0.508 | 0.517 | 0.202 | 0.447 | 0.464 | 5.0 | 0.050 | 0.848 |
| | Classic | BPRMF | 0.226 | 0.508 | 0.518 | 0.201 | 0.447 | 0.460 | 6.3 | 0.050 | 0.848 |
| | Temporal | FPMC | 0.227 | 0.508 | 0.516 | 0.201 | 0.447 | 0.469 | 4.9 | 0.052 | 0.849 |
| | Geo | RankGeoFM | 0.211 | 0.486 | 0.516 | 0.187 | 0.427 | 0.457 | 5.6 | †**0.082** | 0.865 |
| | Tour | ItemMC | †**0.231** | †**0.537** | ▴**0.519** | †**0.212** | †**0.477** | †**0.473** | ▴**2.0** | 0.076 | †**0.871** |
| | Skylines | TestOrder | ▴**0.481** | ▴**0.915** | 0.481 | ▴**0.481** | ▴**0.858** | ▴**0.481** | 2.1 | ▴**0.217** | ▴**0.915** |
| PJ | Basic | Pop | 0.128 | 0.413 | 0.249 | 0.126 | 0.404 | 0.245 | 35.0 | 0.001 | 0.915 |
| | Classic | BPRMF | 0.131 | 0.418 | 0.274 | 0.128 | 0.408 | 0.270 | 30.0 | 0.001 | 0.917 |
| | Temporal | BFsUB | †**0.135** | †**0.423** | †**0.296** | †**0.134** | †**0.416** | 0.285 | †**26.6** | 0.003 | †**0.933** |
| | Geo | PGN | 0.129 | 0.415 | †**0.296** | 0.126 | 0.406 | †**0.286** | 30.0 | †**0.008** | 0.931 |
| | Tour | ItemMC | 0.127 | 0.412 | 0.244 | 0.125 | 0.403 | 0.240 | 28.4 | 0.002 | 0.918 |
| | Skylines | TestOrder | ▴**0.369** | ▴**0.864** | ▴**0.368** | ▴**0.369** | ▴**0.853** | ▴**0.368** | ▴**7.0** | ▴**0.023** | ▴**0.980** |

family, whereas Pop performs the best for the Basic family. It should be noted that the best recommenders from other families often obtain very close or lower values in terms of relevance than Pop, evidencing a strong popularity bias. These results are only improved when including different contextual factors like temporal or geographical information. Second, the TestOrder algorithm produces recommendations with the lowest distance in every case, although in Rome, the ItemMC obtains slightly shorter routes, but the difference is negligible. An important difference, however, is that the best recommender in terms of sequential relevance belongs to the Geo family in one case (New York), to the Tour family in another (Rome), and to the Temporal family in the third one (Petaling Jaya). This confirms the importance of these recommendation families in venue and, especially, route recommendation.

We do observe, however, an interesting difference in these results regarding the larger values of the metrics in Rome. This is especially true for the $FP_s$ metric, and in particular for the ItemMC recommender which is very close to the optimal value as reported by the TestOrder skyline, although all the families obtain values much higher (and closer to the optimal ones) than in the other cases. Our assumption is that this unique behavior in Rome is related to the inherent characteristics of this dataset (see Table 6.4) in comparison with the others: the number of items is very small, which results in a more dense dataset, between 10 and 20 times less sparse than Tokyo or New York[7], together with the fact that the items in this dataset are artificially created by the authors, by clustering existing venues by distance, assuming these items may be more related from a touristic point of view. Even though these conditions may distort the obtained results, we believe it is important to include at least one dataset not based

[7]It should be noted that we tried to reproduce these statistical conditions on any of the other datasets by running simulations and discarding users, items, and check-ins, but we run out of data before we could obtain a comparable dataset.

on check-ins, despite their pervasiveness in the field, since they could provide a different perspective of user behavior.

**Short summary**
Based on the conclusions drawn so far, we can analyze how the classical collaborative filtering algorithms compare against approaches tailored to venue and route recommendation. According to our results, simple models tend to obtain better results than other complex models (mostly because these datasets suffer from very high sparsity), although adding temporal or geographical contexts tend to improve the results. Moreover, those based on geographical or sequential properties tend to be amongst the best ones, but it should be considered that a non-personalized method like a recommender based on popularity provides a strong baseline which some algorithms are not able to beat while others obtain very close, comparable performance. This conclusion, although surprising, follows from the fact that, to the best of our knowledge, this is the first work where so many families have been thoroughly compared and evaluated under a realistic evaluation. Regarding whether performance changes when user sequences in test are considered, we have not found many differences at the item level; our analysis shows that this is because of the large number of potential items in these datasets, which makes it very difficult to suggest relevant venues and in the correct order, since larger differences between sequential and non-sequential metrics were found for those users that receive more relevant recommendations. On the other hand, when the category level is considered instead, larger differences between algorithms arise, where the temporal and geographical ones tend to stand out.

In summary, we have found that many recommenders perform somewhat similarly in terms of relevance, but they differ on other dimensions more related to the route recommendation domain (geographical distance, venue categories) or to the sequentiality dimension. We aim to improve this by using reranking strategies; for instance, by taking a simple, but good enough recommender, is it possible to improve the rest of dimensions? We address this question and summarize the obtained results in the next section.

### 6.3.6 Performance of reranking strategies

In this section, we present experiments for the proposed reranking strategies. For this, we take Equation 6.1 with $\lambda = 0$, thus, only the sequence-aware reranker component is considered (later in Section 6.4 we analyze the results at different values of $\lambda$). Moreover, we rerank the first 20 items returned by each recommender for every user using the 8 components presented in Section 6.2; specifically, $f_{seq}^{rec}$ uses a user-based nearest neighbor with $k = 100$ and vector cosine as user similarity, $f_{seq}^{item}$ and $f_{seq}^{feat}$ obtain better results without the smoothing component ($\alpha = 0$) so it is not used in the reported experiments, and $f_{seq}^{stree}$ considers the last 4 items when querying the suffix tree ($m = 4$), the rest of the rerankers are used as defined previously, since they do not need additional parameters.

Table 6.9 shows the results obtained for the rerankers described in Section 6.2 in New York, Rome, and Petaling Jaya and for five out of the six families of recommenders – we do not include the Skylines because their performance is optimal since they return the test set of the user (see Section 6.3.4), so none of the reranking strategies would help in increasing their performance. In these results, we focus on 3 complementary dimensions that are very important for route recommendation according to our previous discussion: $\mathrm{nDCG_s}$ (sequence-aware item-level relevance), $\mathrm{FP_s}$ (sequence-aware category-level relevance), and Dist (distance of the recommended route). As before, complete results are presented in Appendix A.2.2 as separate tables, showing the results for all the cities presented in previous sections on every evaluation metric (see Tables A.6, A.7, A.8, A.9, and A.10).

**Performance comparison on a city basis**

We observe a similar behavior in the three cities. First, the feature-based Markov Chain reranker ($f_{seq}^{feat}$) is usually the worst (together with the random one, $f_{seq}^{rnd}$), especially in terms of $\mathrm{FP_s}$, but also for $\mathrm{nDCG_s}$, where it tends to decrease the performance with respect to the baseline (base recommender without reranking). This can be attributed to the fact that we are working with a limited number of features (note that there are only 9 categories in the level 1 of Foursquare), so venues with the same feature can be quite different (consider for instance a bar that belongs to the same category as a fast food restaurant, or a hotel whose corresponding level 1 category is the same as a bus station).

Second, it is interesting to observe that the baseline is often generating the longest routes, hence, the reranking strategies allow to create more realistic and affordable routes to the final user (except for the $f_{seq}^{rnd}$, as the reranked items are sorted randomly). However, there is a clear, non-negligible gap between the maximum values achievable with the rerankers (illustrated by the oracle-based reranker $f_{seq}^{oracle}$) and the results we obtain with the rest. Nonetheless, if we analyze this information from a different perspective, we conclude that by simply reordering the first $N$ candidates (in these results, $N = 20$), the performance of some recommenders not specifically designed for route recommendation can be improved in a varied range of percentages (depending on the dimension that we are analyzing). For example, the distance of the route obtained by the recommender from the Basic family (Pop) in PJ is 35.0 Km, while applying $f_{seq}^{dist}$ this distance is reduced to 7.2 Km – a decrease of 80% –; the value of $\mathrm{FP_s}$ in NYC for the Temporal family increases from 0.302 to 0.309 when applying the $f_{seq}^{rec}$ reranker– an improvement of 2.32% –, and the performance on $\mathrm{nDCG_s}$ for the Geo family compared against when the item-based Markov Chain reranker is applied in ROM is 0.426 and 0.467 respectively – an improvement of 9.6%. We believe these outcomes are very positive and promising, as route recommenders normally require many different information sources and long execution times in order to work well, but using this kind of techniques may help to find simple solutions for these cases by balancing a tradeoff between relevance and the rest of the dimensions.

**Performance comparison on a reranker basis**
Regarding specific rerankers, the distance-based reranker ($f_{seq}^{dist}$) is, by definition, the one that reduces the most the distance of the recommended route, but what is more important is that, in some situations, it is able to improve the performance in both $\mathrm{nDCG_s}$ and $\mathrm{FP_s}$ (see for example the results in PJ for most of the families, but especially for Tour, or the results in ROM for all the families except Temporal). The performance of the rerankers based on subsequences ($f_{seq}^{lcs}$ and $f_{seq}^{stree}$) depend heavily on the recommendation family, where in some situations they are able to decrease the distance and improve $\mathrm{FP_s}$, as in PJ, where they outperform the baseline in every case, or in NYC, where these rerankers obtain the best overall result in terms of $\mathrm{FP_s}$.

Finally, we observe that the recommender-based reranker ($f_{seq}^{rec}$) does not usually improve the recommendation performance in any of the dimensions (except in some cases in New York regarding the $\mathrm{FP_s}$); in particular, the distance tends to be very high, not surprisingly since it does not consider explicitly any geographical component. Despite these preliminary negative results, we believe this technique might evidence better results if other recommenders are used (recall that we have only tested a user-based nearest neighbor algorithm without tuning any of its parameters), especially when applied to not collaborative baselines, as in the case of the Tour family in New York where it manages to improve both $\mathrm{FP_s}$ and $\mathrm{nDCG_s}$. A special mention deserves the item-based Markov Chain reranker ($f_{seq}^{item}$), since it is able to reduce the distance of the recommended route in ROM consistently for any recommender family, and, on top of that, it produces some of the best performing results in terms of $\mathrm{nDCG_s}$ and $\mathrm{FP_s}$.

**Performance comparison on a recommender family basis**
Now, if we analyze these results from the perspective of the family of the recommenders, we observe that the behavior is pretty stable in each city, independently of the origin of the recommendations being reranked. We observe, however, that the oracle reranker obtains different values depending on the family, which makes sense since each family may produce a difference set of candidate items to be reranked (those in the top-n). These *optimal* values are lower for those in the Tour family in New York, and higher in the rest, evidencing that the candidate items have a lot of potential (except, to a lower extent, those in the Tour family) and there is room for improvement.

**Short summary**
Overall, these results confirm that it is possible to improve the performance of algorithms by reranking the recommendations at least, in terms of reducing the distance of the routes being suggested to the user, but also in terms of the category-level relevance ($\mathrm{FP_s}$), while keeping the same (or lower but very close) item-level sequential relevance ($\mathrm{nDCG_s}$). In particular, there is always a reranking strategy that allows to improve the performance of the baseline recommender in each of the three compared evaluation dimensions.

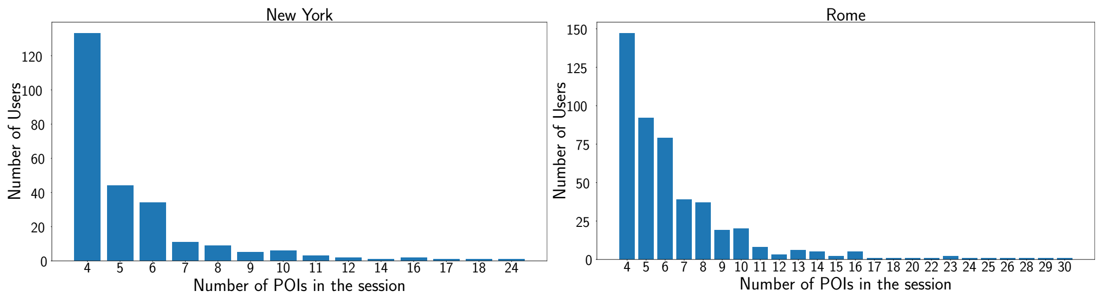


**Figure 6.4:** Frequency of the number of venues each user visited in test for New York and Rome.

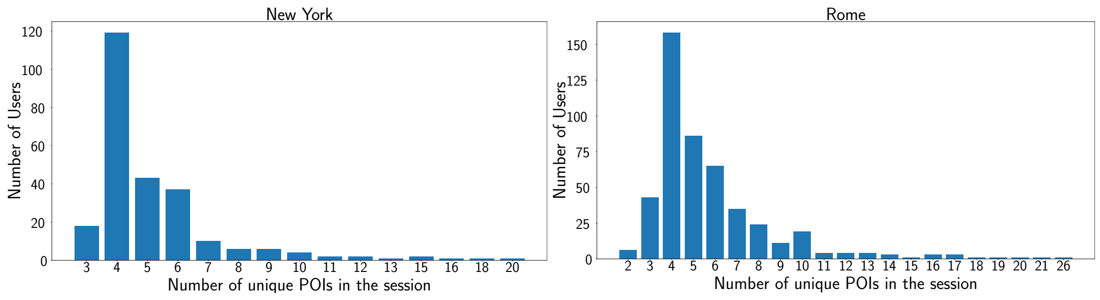


**Figure 6.5:** Frequency of the number of venues each user visited in test for New York and Rome after removing repeated items in a per-user basis.

## 6.4 Discussion

This chapter has covered RG4: *Explore the LBSNs data used in POI recommendation to explore new ways to make recommendations and enable full route recommendation.* In previous sections, we have presented and analyzed the performance of different recommendation techniques applied to the task of route recommendation using reranking tecniques by exploiting sequentiality and the POI categories. We now aim to discuss in more detail some aspects that may have a large impact on the evaluation of our approaches. We first analyze the length of the route from every user in the test set; then, we show the impact of the linear combination weight ($\lambda$ in Equation 6.1) on the different evaluation dimensions for some rerankers and recommenders. We have also analyzed the case where recommenders are allowed to return items previously interacted by the user, a situation very common in POI recommendation that produces a repetition bias since users tend to visit the same venues more than once; however, since this aspect is slightly out of the scope of this chapter, we moved those additional results to Appendix A.2.3

For the first analysis, Figure 6.4 shows a bar plot where we present the number of users with a given length in their test routes, for New York and Rome. In that figure we observe that most users have routes with very few venues (note that the minimum length, as explained in Section 6.3.2, is 4), this effect is further emphasized

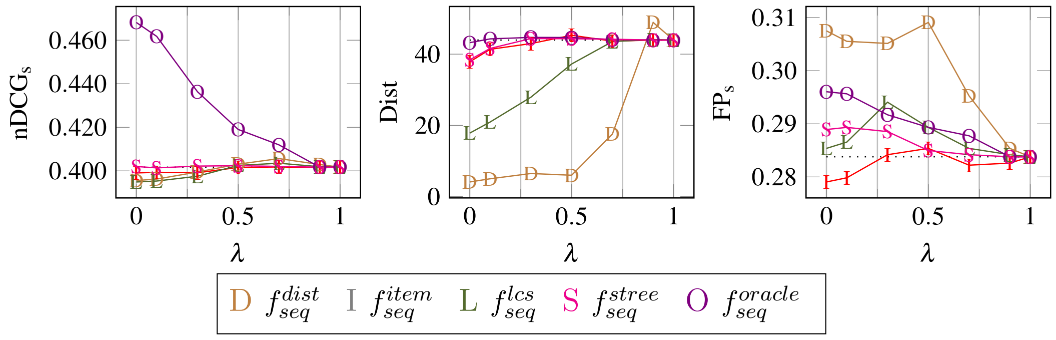


**Figure 6.6:** Sensitivity to $\lambda$ parameter on three evaluation dimensions (nDCG$_s$@10 left, Dist@5 middle, FP$_s$@10 right) for the Pop recommender using the following rerankers (with markers) on New York: $f_{seq}^{dist}$, $f_{seq}^{item}$, $f_{seq}^{lcs}$, $f_{seq}^{stree}$, and $f_{seq}^{oracle}$. The performance for the baseline (no reranker) is included as the horizontal dashed line.

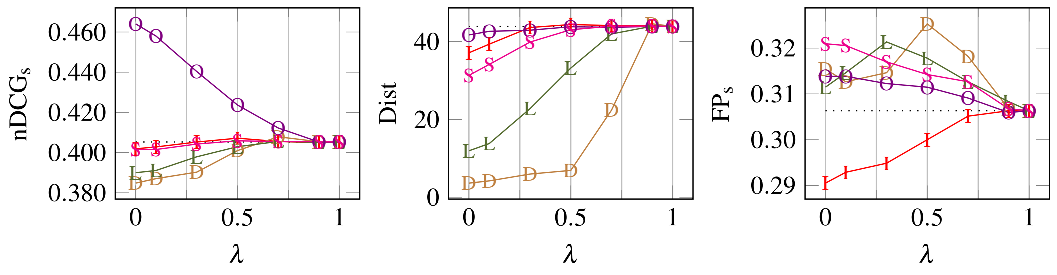


**Figure 6.7:** Same information as in Figure 6.6 but for the IRenMF recommender.

when repeated venues in the test set are removed (see Figure 6.5), where there are users with only 2 (unique) items in their test set. This situation could, in principle, affect the experiments shown in previous sections, since when calculating the metrics presented before, users with short routes contribute much more – because we compute the average of the performance obtained by each user – than those few users with longer routes. Looking at the plots, the number of users with routes of length 5 or less represents more than 50% of the test users in any city. Note that this is an issue related to the general formulation of the cold-start problem, as it reflects that user routes are usually short (regardless of whether users have more or less interactions in the training set), which makes the task of route recommendation even more difficult, in combination with the high sparsity inherent in these datasets.

To continue with our discussion, we show in Figures 6.6 and 6.7 the evolution of different $\lambda$ values to analyze the impact of the sequence-aware reranker component according to Equation 6.1 (recall that the results presented in the previous sections were for $\lambda = 0$). Figure 6.6 shows the results for the Pop recommender on New York whereas Figure 6.7 does the same for IRenMF. We select these recommenders because we want to analyze the effect of the reranking strategies on a non-personalized algorithm (Pop) and on the best-performing method (IRenMF) in this city; in both cases, we compare five rerankers: distance, item-based Markov Chain, based on LCS, based on subsequences computed using a Suffix Tree, and oracle. We observe that, for the three reported dimensions, the larger the value of $\lambda$, the smaller the gap between the rerankers and the oracle; this is a plausible result since for $\lambda = 1$ the output of the reranking strategy is equivalent to that of the baseline recommender. More importantly, the performance of most of the rerankers improve steadily until around $\lambda = 0.5$, where it tends to abruptly get closer to the final value; hence, a *safe* value for $\lambda$ to obtain a tradeoff in all the dimensions would be, as expected, $\lambda = 0.5$. This value would allow us, for instance, to improve $\mathrm{FP_s}$ and the distance of the Pop recommender by using a simple distance-based reranker, while for the IRenMF recommender, a reranker based on LCS would even slightly improve on $\mathrm{nDCG_s}$ (although the distance-based reranker would also provide a good-enough tradeoff in the other dimensions). These results evidence that reranking strategies could be exploited to create more meaningful routes based on recommendations produced by either complex models (such as IRenMF) or simple, not-personalized algorithms (such as Pop).

**Table 6.9:** Effect of the reranking strategies ($\lambda = 0$, 20 candidate items reranked) on each family of recommendation algorithms under three evaluation metrics.

| | | New York | | | Rome | | | Petaling Jaya | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Family | Reranker | nDCG$_s$ | FP$_s$ | Dist | nDCG$_s$ | FP$_s$ | Dist | nDCG$_s$ | FP$_s$ | Dist |
| | Baseline | 0.402 | 0.284 | 43.9 | 0.447 | 0.464 | 5.0 | 0.404 | 0.245 | 35.0 |
| | $f_{seq}^{rnd}$ | 0.383 | 0.297 | 28.0 | 0.402 | 0.452 | 5.9 | 0.387 | 0.274 | 29.5 |
| | $f_{seq}^{dist}$ | 0.396 | ▲**0.308** | ▲**4.1** | 0.469 | †**0.474** | ▲**1.4** | †**0.409** | ▲**0.296** | ▲**7.2** |
| | $f_{seq}^{feat}$ | 0.400 | 0.267 | 33.3 | 0.422 | 0.371 | 5.0 | 0.402 | 0.267 | 33.2 |
| Basic | $f_{seq}^{item}$ | 0.399 | 0.279 | 37.8 | †**0.473** | 0.469 | 1.8 | 0.408 | 0.262 | 19.5 |
| | $f_{seq}^{rec}$ | †**0.406** | 0.298 | 42.4 | 0.422 | 0.452 | 6.0 | 0.407 | 0.271 | 26.3 |
| | $f_{seq}^{lcs}$ | 0.395 | 0.285 | 17.8 | 0.440 | 0.446 | 2.3 | 0.403 | 0.274 | 14.8 |
| | $f_{seq}^{stree}$ | 0.402 | 0.289 | 38.4 | 0.446 | 0.466 | 3.2 | 0.403 | 0.263 | 25.9 |
| | $f_{seq}^{oracle}$ | ▲**0.468** | 0.296 | 43.2 | ▲**0.614** | ▲**0.482** | 4.2 | ▲**0.456** | 0.247 | 34.2 |
| | Baseline | 0.404 | 0.285 | 45.3 | 0.447 | 0.460 | 6.3 | 0.408 | 0.270 | 30.0 |
| | $f_{seq}^{rnd}$ | 0.382 | 0.292 | 30.3 | 0.403 | 0.450 | 5.9 | 0.394 | 0.278 | 30.7 |
| | $f_{seq}^{dist}$ | 0.395 | **0.309** | ▲**4.2** | 0.468 | †**0.475** | ▲**1.4** | †**0.410** | ▲**0.294** | ▲**7.4** |
| | $f_{seq}^{feat}$ | 0.398 | 0.267 | 33.5 | 0.424 | 0.373 | 5.0 | 0.402 | 0.269 | 34.0 |
| Classic | $f_{seq}^{item}$ | 0.400 | 0.276 | 38.0 | †**0.476** | 0.468 | 1.8 | 0.409 | 0.268 | 18.5 |
| | $f_{seq}^{rec}$ | †**0.406** | 0.300 | 42.4 | 0.422 | 0.452 | 6.0 | 0.407 | 0.273 | 26.5 |
| | $f_{seq}^{lcs}$ | 0.395 | 0.284 | 17.9 | 0.440 | 0.447 | 2.3 | 0.405 | 0.279 | 13.2 |
| | $f_{seq}^{stree}$ | 0.404 | 0.294 | 38.6 | 0.447 | 0.465 | 3.7 | 0.405 | 0.275 | 22.1 |
| | $f_{seq}^{oracle}$ | ▲**0.468** | 0.300 | 44.3 | ▲**0.612** | ▲**0.482** | 4.9 | ▲**0.455** | 0.269 | 29.0 |
| | Baseline | †**0.404** | 0.302 | 42.4 | 0.447 | †**0.469** | 4.9 | †**0.416** | 0.285 | 26.6 |
| | $f_{seq}^{rnd}$ | 0.379 | 0.317 | 25.3 | 0.409 | 0.449 | 6.0 | 0.383 | 0.308 | 28.1 |
| | $f_{seq}^{dist}$ | 0.389 | ▲**0.319** | ▲**3.5** | 0.464 | 0.468 | ▲**1.4** | 0.412 | ▲**0.326** | ▲**5.6** |
| | $f_{seq}^{feat}$ | 0.388 | 0.272 | 30.6 | 0.421 | 0.375 | 5.0 | 0.397 | 0.291 | 30.2 |
| Temporal | $f_{seq}^{item}$ | 0.400 | 0.293 | 37.2 | †**0.474** | 0.465 | 1.9 | 0.412 | 0.283 | 17.5 |
| | $f_{seq}^{rec}$ | 0.403 | 0.309 | 41.5 | 0.422 | 0.452 | 6.1 | 0.407 | 0.292 | 26.1 |
| | $f_{seq}^{lcs}$ | 0.388 | 0.314 | 10.8 | 0.441 | 0.447 | 2.3 | 0.407 | 0.311 | 10.9 |
| | $f_{seq}^{stree}$ | 0.398 | 0.311 | 29.6 | 0.445 | 0.468 | 3.1 | 0.411 | 0.301 | 17.3 |
| | $f_{seq}^{oracle}$ | ▲**0.462** | 0.308 | 39.9 | ▲**0.608** | ▲**0.482** | 4.1 | ▲**0.457** | 0.287 | 25.8 |
| | Baseline | 0.405 | 0.306 | 43.9 | 0.427 | 0.457 | 5.6 | 0.406 | 0.286 | 30.0 |
| | $f_{seq}^{rnd}$ | 0.378 | 0.307 | 22.5 | 0.397 | 0.447 | 5.9 | 0.390 | 0.307 | 25.1 |
| | $f_{seq}^{dist}$ | 0.385 | 0.315 | ▲**3.6** | 0.456 | †**0.468** | ▲**1.4** | 0.405 | ▲**0.315** | ▲**5.8** |
| | $f_{seq}^{feat}$ | 0.393 | 0.281 | 32.8 | 0.414 | 0.364 | 5.3 | 0.397 | 0.282 | 26.8 |
| Geo | $f_{seq}^{item}$ | 0.402 | 0.291 | 37.1 | †**0.467** | 0.466 | 2.1 | †**0.412** | 0.270 | 18.8 |
| | $f_{seq}^{rec}$ | †**0.405** | 0.311 | 42.0 | 0.417 | 0.453 | 6.0 | 0.407 | 0.290 | 25.8 |
| | $f_{seq}^{lcs}$ | 0.390 | 0.311 | 11.9 | 0.426 | 0.440 | 2.2 | 0.401 | 0.308 | 10.5 |
| | $f_{seq}^{stree}$ | 0.402 | **0.321** | 31.3 | 0.431 | 0.458 | 3.5 | 0.404 | 0.302 | 19.4 |
| | $f_{seq}^{oracle}$ | ▲**0.464** | 0.314 | 41.7 | ▲**0.586** | ▲**0.472** | 4.6 | ▲**0.449** | 0.287 | 28.8 |
| | Baseline | 0.391 | 0.279 | 44.9 | †**0.477** | 0.473 | 2.0 | 0.403 | 0.240 | 28.4 |
| | $f_{seq}^{rnd}$ | 0.364 | 0.305 | 23.9 | 0.400 | 0.448 | 5.7 | 0.390 | 0.291 | 30.8 |
| | $f_{seq}^{dist}$ | 0.381 | 0.311 | ▲**4.2** | 0.467 | †**0.474** | ▲**1.4** | †**0.412** | ▲**0.309** | ▲**7.1** |
| | $f_{seq}^{feat}$ | 0.374 | 0.277 | 20.9 | 0.420 | 0.359 | 5.0 | 0.401 | 0.278 | 31.6 |
| Tour | $f_{seq}^{item}$ | 0.397 | 0.283 | 38.1 | 0.477 | 0.470 | 1.8 | 0.406 | 0.271 | 16.9 |
| | $f_{seq}^{rec}$ | †**0.403** | 0.289 | 41.4 | 0.427 | 0.451 | 5.8 | 0.408 | 0.273 | 26.6 |
| | $f_{seq}^{lcs}$ | 0.382 | ▲**0.312** | 12.0 | 0.438 | 0.446 | 2.1 | 0.406 | 0.290 | 13.9 |
| | $f_{seq}^{stree}$ | 0.386 | 0.295 | 32.4 | 0.457 | 0.466 | 2.4 | 0.403 | 0.272 | 21.5 |
| | $f_{seq}^{oracle}$ | ▲**0.442** | 0.285 | 44.4 | ▲**0.600** | ▲**0.482** | 3.0 | ▲**0.455** | 0.244 | 28.0 |

# 7

# Data augmentation in POI recommendation

In Chapter 3 we defined the POI recommendation problem and performed a review of POI recommendation approaches between 2011 and 2019 showing, among other aspects, the most extended types of algorithms, evaluation methodologies, and also the challenges that still exist in the area. Moreover, in the previous chapter we explored the use of reranking techniques to generate full routes to users showing the importance of the geographical component to create these trajectories. In this chapter, we continue exploring the POI recommendation problem by performing a more in-depth analysis of the sparsity issue and the possible biases that can be found in LBSNs. For that reason, in this chapter we further explore some of these issues by presenting a complete analysis in different cities in the global-scale check-in dataset of Foursquare. Hence, we use data aggregation techniques in order to improve the performance of the recommenders in the POI recommendation domain based on cross-domain techniques. To do this, in Section 7.1 we motivate our approach and introduce the key concepts that are being addressed in this chapter. In Section 7.2 we define more in detail the cross-recommendation problem; we then introduce our aggregation strategies in Section 7.3. Then, in Section 7.4 we perform experiments in 8 independent cities from the full dataset of Foursquare, we show the evolution of the recommenders in accuracy and non-accuracy metrics using 3 different strategies: first, we will show the evolution of the recommenders in each city separately (single-city), then we will expand the data for each city with the 7 closest cities (by distance) to each of them, and finally we use the data from the eight most popular cities to produce recommendations in each target city independently. Finally, we will also show the performance of the recommendations in two different types of users, tourists and locals, and analyze the biases that each group may evidence separately. Finally, in Section 7.5 we discuss about the implications that our proposed multi-city aggregation strategies may have in the POI recommendation area.

The work presented in this chapter has been sent to publication in the Information Processing and Management journal:

- **Pablo Sánchez** and Alejandro Bellogín. On the effects of aggregation strategies for different groups of users in venue recommendation. Submitted to *Information Processing and Management*. Under Review (2nd round of review).

A preliminary version of this work was published in the following workshop:

- **Pablo Sánchez** and Alejandro Bellogín. A novel approach for venue recommendation using cross-domain techniques. In the 2nd Workshop on Intelligent Recommender Systems by Knowledge Transfer & Learning (RecSysKTL), held in conjunction with the 12th ACM Conference on Recommender Systems, Vancouver, Canada, Oct, 2018.

## 7.1 Motivation

Generally, in most areas of knowledge it is said that the accuracy of the algorithms might be improved by extending the available information with additional data. In the Recommender Systems domain this is especially important in datasets with a high level of sparsity, where it is difficult to learn patterns between users and items due to the low number of interactions between them. Transfer (or cross-domain) learning is one of these valuable techniques that allow us to use external or additional information, mainly to improve the performance in the target domain (Lu et al., 2015a, Yu et al., 2018). In the context of RS, cross-domain recommendation is a recent and active research topic, where POI recommendation has been acknowledged as a potential application area (Zheng, 2015). However, not many experimental comparisons have been performed using cross-domain or augmentation techniques in this field with a realistic evaluation.

Moreover, besides the inherent personalized results that are expected to be received by users, they are traditionally treated equally when measuring the performance of the recommenders (as explained with the user attributes in Section 4.3). However, this is slowly changing since recently the field is paying more attention to whether users with different attributes (such as age, gender, nationality, etc.) receive the same treatment, or, in other terms, if the recommender system provides *fair* recommendations (Ekstrand et al., 2018, Edizel et al., 2020). Nonetheless, these efforts are not easy to generalize to other domains, in particular because users do not share the same characteristics in different recommender systems and also because in some domains, there might be some types of users that might not be easily categorized with just with a single attribute, even though such groups exist and evidence distinct behavior. In particular, in the tourism domain (which is typically studied with data from LBSNs), check-in data has been used to characterize four types of travellers (Dietz et al., 2019): vacationers, explorers, voyagers, and globetrotters, thus going beyond the classical tourist roles that are usually considered (either leisure or business). Nonetheless, it is also acknowledged that most of these users are not actual tourists but local users, or, at least, that they tend to travel a limited distance from their home locations (an effect called *travel locality*) (Levandoski et al., 2012). In any case, and independently of these uncertainties in the types of users that can be identified in these systems, we think it is important to study if the susceptible algorithms to be used in such systems are to some extent biased towards any particular user group, or if these groups are transparent to the algorithms.

Considering these issues, in this chapter we analyze the effect of producing recommendations when using augmented information extracted from other cities, by exploiting different Multi-City Aggregation (MCA) strategies, either based on the number of interactions (popular cities) or by proximity with respect to the target city. Furthermore, we incorporate into our work an exploratory analysis with the aim to uncover biases or effects that may inadvertently be present when making venue recommendations from LBSN data, either through state-of-the-art algorithms or when using our proposed MCA strategies.

The work conducted in this chapter provides a thorough comparison of two Multi-City Aggregation strategies for venue recommendation under a realistic time-aware evaluation methodology. We report the results obtained by the MCA strategies in terms of ranking accuracy, novelty, diversity, and coverage using the global-scale check-in dataset of Foursquare. Moreover, we complement these results with an extensive analysis of the effect of these strategies (together with the base performance of state-of-the-art approaches) in two different types of users, tourists and locals.

## 7.2 Cross-domain recommendation

To properly understand the aggregation strategies proposed in this chapter, we first need to introduce the area of cross-domain recommendation, where, in fact, there exist multiple definitions of "domain" (Cantador et al., 2015): one can consider two items belong to different domains if they have different values for a specific attribute but others may argue different domains implies two separate systems or two types of items. In any case, the basic idea behind this type of recommendation is that, to improve recommendations over a target domain $\mathcal{D}_T$, some kind of knowledge from a source domain $\mathcal{D}_S$ needs to be exploited. In order to illustrate these concepts, consider for example a source domain of books and a target domain of movies. We can make recommendations of users who do not have a mature rating history in the target domain by using some information stored in the source domain, i.e., if the target user liked drama and horror books, she would probably also like movies of the same style even if the items are from different domains.

Depending on the information analyzed and the destination users to make recommendations, Cantador et al. (2015) describe the following three main recommendation goals of cross-domain recommendation:

- Linked-domain recommendations: recommend items in the target domain by analyzing both the target and source domains.
- Cross-domain recommendations: recommend items in the target domain to users in the source domain by using only the information of the source domain.
- Multi-domain recommendations: recommend items belonging to the target or the source domain to all kind of users.

Furthermore, we must consider that $\mathcal{D}_S$ and $\mathcal{D}_T$ may share some information about the users, the items, or both, which allows us to categorize the different scenarios according to the data overlap as: no overlap, item overlap, user overlap, and full overlap (both user and item overlap). Additionally, we can also classify the cross-domain techniques according to how they exploit the knowledge. Based on the work of Cantador et al. (2015), there are two different categories: “aggregating knowledge”, when the knowledge of the domains (user preferences, similarities, or single-domain recommendations) is aggregated, and “linking and transferring knowledge”, where there is a knowledge transfer between the different domains in order to produce recommendations, for instance by using common knowledge (semantic networks, items attributes, etc.), sharing latent features, or transferring the rating patterns.

Nonetheless, we want to emphasize we have not found many examples of cross-domain experiments combining more than two or three domains – usually, movies, music, and books –, except for the works presented in Rafailidis and Crestani (2017) and Sahebi and Brusilovsky (2015). In former, the authors exploited the information of 10 domains (categories of different products from Epinions, hence, not related with tourism) in order to analyze the performance of the recommendations using ranking metrics (nDCG and R), however, these datasets are much smaller in terms of ratings than the ones we use in this chapter (the largest one contains around 200K ratings) and they are all more dense; while in the second one, the authors used 21 domains (defined as the categories from different Yelp businesses) and measured the effects of cross-domain recommendation in terms of RMSE by comparing the performance on several domain pairs.

## 7.3 Aggregation techniques in POI recommendation

To avoid the inherent problems and limitations prevalent in POI recommendation, we introduce now the concept of Multi-City Aggregation (MCA) strategies, that allow us to augment the available data used by the recommendation algorithms to suggest interesting POIs to users. The basic idea behind our proposed strategies is that, to improve recommendations over a target city, the data from multiple cities will be aggregated and exploited when training the recommendation algorithms. Since an infinite number of potential strategies may exist, we shall focus on those that maximize the overlap information between users and items.

More specifically, the main goal we aim to achieve, hence, if we focus on CF algorithms, would be to find highly active users in the aggregated cities so they could alleviate the inherent sparsity problem. Therefore, we consider the following possibilities:

- Geographical nearest MCA (N): we use the $n$ closest cities to the target city as the aggregated information. We aim to capture cultural patterns (Yang et al., 2016), while, at the same time, we keep control of the number of cities we consider. Additionally, as a special case of Nearest MCA, we consider a country-based MCA (C), where the aggregated information is built by all the cities of the same

**Table 7.1:** Jaccard coefficient between users in the training splits of each city and the corresponding MCA strategy.

| Multi-City Aggregation strategy (MCA) | Cities | | | | | | | | Average |
|---|---|---|---|---|---|---|---|---|---|
| | Istanbul | Jakarta | Kuala Lumpur | Mexico City | Moscow | Santiago | São Paulo | Tokyo | |
| N-MCA | 89.54% | 67.14% | 83.44% | 89.36% | 94.48% | 85.28% | 64.87% | 79.13% | 81.66% |
| C-MCA | 67.64% | 57.54% | 66.14% | 67.80% | 61.35% | 83.68% | 34.43% | 74.52% | 64.14% |
| P-MCA | 29.09% | 13.63% | 13.53% | 9.30% | 8.71% | 7.73% | 8.99% | 10.79% | 12.72% |

country as the target city. In this strategy, we assume that users tend to visit more often those cities of the same country, or, at least, that there are some kind of patterns shared among them, mostly due to a similar culture and language (Yang et al., 2016), even though some situations (large countries) may add too much noise into the model. The idea for this strategy comes from previous works in POI recommendation that use datasets with interactions belonging to a specific country or a state/province of a country, as in Li et al. (2015a) and Xie et al. (2016a). Note that for the N-MCA there could be cities from a different country with respect to the target city while in C-MCA all selected cities will belong to the same country, even though some of them may not be so close to the target city.

- Most-popular MCA (P): we use the $n$ cities with more check-ins as the aggregated data. This strategy allows us to test whether considering those cities that the system has more information about can be useful for the model. We hypothesize that having more information should be helpful for the recommendation algorithms, however, this strategy might also be more sensitive to noise and may not improve the user overlap (even though a high user overlap may not be sufficient to obtain a performance improvement). This strategy is also influenced by other works in POI recommendation that use datasets with check-ins from all over the world, including Gao et al. (2013), Zhang and Chow (2015c) and Zhang and Chow (2015b).

As a first validation that these strategies might be actually helpful for POI recommendation, we show in Table 7.1 the percentage of common users (measured using the Jaccard coefficient) between the training set of each target city and the corresponding training set of the multiple cities being aggregated according to the different strategies described before. Using the same dataset that will be used and explained later in the experiments, we compute these percentages using the Jaccard coefficient as follows (as in Yang et al. (2016)):

$$\text{Common Users}(C_1, C_2) = \frac{|\mathcal{U}(C_1) \cap \mathcal{U}(C_2)|}{|\mathcal{U}(C_1) \cup \mathcal{U}(C_2)|} \tag{7.1}$$

where $\mathcal{U}(C)$ denotes the set of users in city $C$.

We observe that the N-MCA and C-MCA strategies are really useful to find more users in common, however, it is not clear the actual effect this result may have on the performance of venue recommendation algorithms, especially on those not based

**Table 7.2:** Description of the temporal partition evaluated (the complete dataset, the training set, Tr, and the test set, Te), where $U$, $I$, and $C$ denote the number of users, items, and check-ins (either with repetitions, $C_R$, or without, $C_r$). The column $\delta(C)$ denotes the density of dataset as computed by $C/(U \cdot I)$.

| check-in period | U | I | $\mathbf{C_R}$ | $\mathbf{C_r}$ | $\delta(\mathbf{C_R})$ | $\delta(\mathbf{C_r})$ | $\mathbf{C_r/U}$ | $\mathbf{C_r/I}$ |
|---|---|---|---|---|---|---|---|---|
| Apr'12-Sep'13 | 267k | 3.7M | 33.3M | 15.1M | 0.0034% | 0.0015% | 123.60 | 9.16 |
| Tr: May-Oct '12 | 202k | 1.1M | 9.9M | 4.8M | 0.0044% | 0.0021% | 23.67 | 4.31 |
| Te: Nov '12 | 150k | 352k | 831k | 831k | 0.0017% | 0.0017% | 5.52 | 2.36 |

on nearest neighbors. At the same time, even though the P-MCA strategy does not discover many users in common, since it includes much more data to train the recommenders, it may be more beneficial to some recommendation approaches. Hence, in the experiments we shall test which approach is actually more helpful to obtain better recommendations.

In summary, and drawing related concepts available in the literature, our proposal would be similar to some techniques from cross-domain (or transfer learning) recommendation, if we consider each city as a different domain. In that way, when we augment the available information from the different cities (according to the proposed MCA strategies), we would combine data from different domains, hence performing a *cross-domain recommendation*. More specifically, according to the taxonomy presented in Cantador et al. (2015) and presented briefly in the previous section, it would fit in the category of merging user preferences by aggregating knowledge, since we combine multiple sources of personal preferences (basically, the check-ins from various cities and the target city). Additionally, our scenario is special and more difficult according to the literature since no item overlap exists between the domains, this is because an item (venue) will never appear in a different domain since they are unique (even two places of the same food or clothing chain will be considered different).

The main advantage of applying some kind of cross-domain in venue recommendation is that we can expand the knowledge of the recommenders with a larger number of users and items, in order to establish more relationships between them and find better patterns. However, it is not obvious how we should select such knowledge: on the one hand, noise might be added to the model, on the other hand, such information may not to be useful at all. This is exactly what we propose to study and analyze in this chapter: MCA strategies that select data according to different hypotheses and criteria.

## 7.4 Experiments

### 7.4.1 Dataset

The experiments have been performed using the global-scale check-in dataset of Foursquare[1] made public by Yang et al. (2016), by capturing those check-ins posted by Twitter users (the same dataset used in Sections 4.5 and 6.3.1). Considering the worldwide nature of this dataset, it seems impractical for us to select a training/test partition that would remain comparable for any city in the world. Therefore, we aimed to maximize the amount of data included for training the models and to test their performance, while, at the same time, the temporal patterns in each split should not belong to seasons that are too different to each other. Based on this, and starting from the original 33M check-ins, we created a temporal split containing 6 months of data in its training step (from May to Oct '12) and one month for test (Nov '12); this period of the year was selected to capture seasonal trends related to a particular season of the year (summer) while including enough data to obtain significant results. In this regard, it is important to mention that it is common to find POI recommendation proposals that perform other type of splits for evaluation, such as random partitions, cross-validation, or temporal per user splits (Liu et al., 2014, Ye et al., 2011, Yuan et al., 2016). However, we argue that these types of splits are less realistic, since in those partitions we may be predicting user interactions that occurred in the past mixed with other events that occur in the future. In addition, user tastes change over time and also there may be global trends that we would not be taking into account if a temporal partition is not used.

Table 7.2 shows more statistics of both the original dataset and the training/test split used in this chapter. Additionally, as a pre-processing step, we performed a 2-core before splitting the data into training and test, which means that every user and POI has at least two interactions.

### 7.4.2 Compared baselines

We report results obtained by the following state-of-the-art recommenders grouped in different families according to the common mechanisms used to make the recommendations.

- Classic no personalized (NP). Traditional recommendation algorithms that do not learn a profile for each user:
  - Pop: a popularity recommender. Already explained in Section 4.5.1.
  - Rnd: a random recommender. Already explained in Section 4.5.1
- Only geographical (Geo). Basic algorithms used to model only the geographical component:

[1]Global-scale check-in dataset, `https://sites.google.com/site/yangdingqi/home/foursquare-dataset`

**Table 7.3:** Parameters used with the evaluated recommenders. SetC and SetJ stand for SetCosine and SetJaccard. Parameters optimized using P@5.

| Recommender | Parameters |
|---|---|
| Rnd | None |
| Pop | None |
| AvgDis | None |
| KDE | None |
| IB | Sim={SetC, SetJ }, $k$={5, 10, $\cdots$ , 100} |
| UB | Sim={SetC, SetJ }, $k$={5, 10, $\cdots$ , 100} |
| HKV | Factors={10, 50, 100}, $\alpha$={0.1, 1, 10}, $\lambda$={0.1, 1, 10} |
| BPRMF | Factors={10, 50, 100}, Iter=50, LearnRate=0.5, RegJ=RegU/10, BiasReg={0, 0.5, 1}, RegU=RegI={0.0025, 0.001, 0.005, 0.01, 0.1} |
| GeoBPR | MaxDist={1, 4}, Factors={10, 50, 100}, BiasReg={0, 0.5, 1}, Iter=50, RegU=RegI={0.0025, 0.001, 0.005, 0.01, 0.1}, LearnRate=0.05 |
| IRenMF | $k$=10, Clusters=50, $\lambda_1$=$\lambda_2$=0.015, Factors={50, 100}, $\alpha$={0.4, 0.6}, $\lambda_3$={0.1, 1} |
| RankGeoFM | Factors={10, 50, 100}, $k$={10, 50, 100, 200}, $\alpha$={0.1, 0.2}, c=1, dec=1, $\varepsilon$=0.3, boldDriv=true, Iter=120, LearnRate=0.001, mRate=0.001 |
| FMFMGM | MGM: $\alpha$ = {0.2, 0.4}, $\theta$ = {0.02, 0.1}, dmax=15. PMF: Iter=30, Factors={50, 100}, $\alpha_2$={20, 40}, $\beta$=0.2, LearnRate=0.0001, sigmoid=false |
| GeoSoCa | Sim=SetJ, $k$=100 |
| PGN | $k$=100, Similarity=SetJ |

– AvgDis: algorithm that suggests the closest POIs to the user's average location. Explained in Section 6.3.4.

– KDE: Kernel Density Estimation recommender already explained in Section 6.3.4.

- Classic collaborative filtering (CF-NN). Traditional recommendation algorithms based on nearest neighbors:

  – IB: an item-based nearest neighbor algorithm. Already explained in Section 2.2.2.

**Table 7.4:** Optimal parameters of models for each city. The order of the presented parameters is: for UB and IB, similarity and neighborhood size; for BPRMF, factors, RegU, BiasReg; for HKV, factors, $\alpha$, $\lambda$; for IRenMF, factors, $\alpha$, $\lambda_3$; for RankGeoFM, factors, neighborhood size, $\alpha$; for GeoBPR, maxDist, factors, RegU, BiasReg; for FMFMGM, $\alpha$, $\theta$, Factors, $\alpha_2$.

| City | UB | IB | BPRMF | HKV | IRenMF | RankGeoFM | GeoBPR | FMFMGM |
|---|---|---|---|---|---|---|---|---|
| IST | SetJ, 90 | SetC, 100 | 50, 0.001, 0 | 10, 10, 10 | 100, 0.4, 1 | 100, 50, 0.1 | 4, 100, 0.001, 0 | 0.4, 0.02, 100, 20 |
| JAK | SetJ, 100 | SetC, 80 | 100, 0.0025, 0 | 10, 10, 10 | 100, 0.4, 1 | 100, 200, 0.2 | 1, 100, 0.001, 0 | 0.2, 0.02, 100, 20 |
| KUA | SetJ, 100 | SetJ, 100 | 50, 0.01, 0 | 10, 10, 10 | 100, 0.4, 1 | 100, 50, 0.2 | 1, 50, 0.001, 0 | 0.4, 0.02, 100, 20 |
| MEX | SetJ, 100 | SetJ, 100 | 100, 0.01, 0 | 10, 10, 10 | 100, 0.4, 1 | 100, 100, 0.2 | 1, 100, 0.001, 0 | 0.4, 0.1, 100, 20 |
| MOS | SetC, 100 | SetJ, 100 | 100, 0.01, 0 | 50, 10, 1 | 100, 0.4, 1 | 100, 200, 0.1 | 1, 50, 0.0025, 1 | 0.4, 0.1, 100, 20 |
| SAN | SetJ, 90 | SetJ, 80 | 50, 0.005, 0 | 10, 10, 10 | 100, 0.4, 1 | 100, 100, 0.1 | 1, 50, 0.001, 0 | 0.4, 0.02, 100, 20 |
| SAO | SetJ, 100 | SetJ, 100 | 100, 0.1, 0 | 50, 10, 0.1 | 100, 0.6, 0.1 | 100, 50, 0.2 | 1, 100, 0.001, 1 | 0.4, 0.02, 100, 20 |
| TOK | SetJ, 80 | SetC, 80 | 50, 0.1, 0 | 10, 10, 10 | 100, 0.4, 1 | 100, 10, 0.1 | 4, 10, 0.001, 0 | 0.4, 0.02, 100, 20 |

  - UB: a user-based nearest neighbor algorithm. Already explained in Section 2.2.2.

- Classic matrix factorization (CF-MF). Traditional recommendation algorithms based on matrix factorization approaches:

  - HKV: a matrix factorization algorithm. Already explained in Section 2.2.2.
  - BPRMF: Bayesian Personalized Ranking using a matrix factorization approach. Already explained in Section 2.2.2.

- POI models (POI). State-of-the-art POI recommendation algorithms:

  - GeoBPR: a POI recommendation approach that uses BPR to optimize the model (Yuan et al., 2016). It incorporates the geographical influence by assuming that users prefer to visit close rather than remote POIs with respect to the ones that the user has already visited.
  - IRenMF: weighted POI Matrix Factorization method explained in Section 6.3.4.
  - RankGeoFM: ranking-based matrix factorization approach explained in Section 6.3.4.

- Hybrid POI recommendation models (H-POI). POI recommendation algorithms whose final score is the combination of two or more independent algorithms:

  - FMFMGM: is a fusion model proposed in Cheng et al. (2012) that combines the Multi-center Gaussian Model technique (MGM) with Probabilistic Matrix Factorization (PMF).
  - GeoCFCa: a hybrid POI recommendation model based on the technique proposed in Zhang and Chow (2015b) that combines the geographical influence using a two-dimensional KDE and the social and categorical influences modeled by two different power-law distributions. As in the Foursquare dataset the social information is not available for all the users, we decided to use a $k$-NN algorithm as a substitute for the social component; because of this, we changed its name from the original GeoSoCa to GeoCFCa.
  - PGN: hybrid POI recommendation algorithm that combines the UB, Pop, and AvgDis recommenders. Explained in Section 6.3.4.

In order to make a fair comparison among all the evaluated baselines, we removed repetitions in a user basis for some classic algorithms. When repetitions are allowed, we either aggregated them as frequencies or keep all the repeated interactions. This means that we kept three versions of the training set: with and without check-in frequencies but where each user-item pair only appeared once, and another scenario where the user-item pairs might be repeated. More specifically, the following POI recommendation algorithms could exploit the frequency of users when visiting a specific venue: AvgDis, IRenMF, FMFMGM, RankGeoFM and GeoCFCa; whereas KDE is the only recommender that uses the training set with repetitions.

For every recommender except the IRenMF algorithm, we used the RankSys library (Castells et al., 2015); for IRenMF we used the implementation provided by Liu et al. (2014), available here[2]; for RankGeoFM and GeoBPR, we implemented our own versions on top of RankSys, based on the implementation provided by LibRec[3] for the former, and on the code provided by Han and Yamana (2020) for the latter. Finally, we also implemented our own versions of GeoCFCa and FMFMGM based on the code provided by Liu et al. (2017) for both of them and Han and Yamana (2020) for GeoCFCa.

The similarities used in the $k$-NN recommenders are based on set operations as the data does not include ratings: SetJ is the well-known Jaccard Index and SetC is based on the similarity proposed in Aiolli (2013), always considering that the similarities between users/items are symmetrical.

We want to note that the algorithms based on matrix factorization, since they start from a random initialization and the number of iterations is also a parameter, they may obtain different results each time they are executed. This would affect the following algorithms: FMFMGM, IRenMF, RankGeoFM, HKV, BPRMF and GeoBPR.

### 7.4.3 Experimental setup

Based on the temporal split presented in Table 7.2, we decided to focus on the eight largest cities in terms of number of check-ins: Istanbul (IST), Jakarta (JAK), Kuala Lumpur (KUA), Mexico City (MEX), Moscow (MOS), Santiago (SAN), São Paulo (SAO) and Tokyo (TOK). Hence, we will compare the recommendations produced using the information of the training set of each of these eight cities independently (Single City) with the recommendations obtained by augmenting the data from any other cities in the dataset according to the proposed strategies.

To evaluate the recommenders, we applied the already explained methodology in previous chapters called TrainItems (Said and Bellogín, 2014), where only the venues that appear in the training set of each target city are considered as candidates, except the ones already rated by the user. Besides, we filter out in the test set all interactions that appear in the training set, so the test set is only composed by new preferences. Moreover, since the performance metrics we use are not well-defined when repetitions exist in the data, we eliminate all repeated interactions from the test set, simulating that each user will visit each POI only once. Then, we compare the recommendations generated by the different algorithms when using different training information according to the presented MCA strategies: multiple cities selected based on the nearest cities (N-MCA and C-MCA) and based on most popular cities (P-MCA). We want to emphasize that regardless of the data used to augment and train the algorithms, the candidate POIs will always belong to the target city.

[2]IRenMF implementation, http://spatialkeyword.sce.ntu.edu.sg/eval-vldb17/
[3]LibRec, https://www.librec.net/

**Table 7.5:** Performance results (nDCG@5) when no MCA strategy is used. In bold, we show the highest value for each city in each family and we show with a dagger the highest value in each city.

| Family | Rec | IST | JAK | KUA | MEX | MOS | SAN | SAO | TOK |
|---|---|---|---|---|---|---|---|---|---|
| NP | Pop | **0.054** | **0.066** | **0.066** | **0.041** | **0.027** | **0.051** | **0.053** | **0.069** |
|  | Rnd | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| Geo | AvgDis | 0.001 | 0.001 | 0.001 | 0.001 | 0.002 | 0.001 | 0.002 | 0.001 |
|  | KDE | **0.003** | **0.004** | **0.004** | **0.006** | **0.006** | **0.005** | **0.008** | **0.005** |
| CF-NN | IB | 0.059 | 0.035 | 0.042 | 0.013 | 0.017 | 0.026 | 0.015 | 0.048 |
|  | UB | **0.073** | **0.070** | **0.073** | **0.044** | **0.037** | **0.053** | **0.049** | **0.069** |
| CF-MF | HKV | **0.070** | **0.066** | **0.066** | **†0.047** | **0.039** | **0.050** | **0.048** | 0.059 |
|  | BPRMF | 0.053 | 0.057 | 0.064 | 0.037 | 0.029 | 0.047 | 0.035 | **0.066** |
| POI | GeoBPR | 0.064 | 0.067 | **†0.073** | **0.046** | **†0.044** | **0.052** | **0.052** | **0.068** |
|  | IRenMF | **†0.074** | **†0.071** | 0.072 | 0.042 | 0.035 | 0.049 | 0.044 | 0.065 |
|  | RankGeoFM | 0.059 | 0.044 | 0.051 | 0.027 | 0.019 | 0.031 | 0.025 | 0.036 |
| H-POI | FMFMGM | 0.060 | 0.050 | 0.060 | 0.029 | 0.028 | 0.033 | 0.034 | 0.061 |
|  | GeoSoCa | 0.033 | 0.026 | 0.029 | 0.017 | 0.016 | 0.013 | 0.017 | 0.026 |
|  | PGN | **0.067** | **0.067** | **0.070** | **0.043** | **0.032** | **†0.054** | **†0.057** | **†0.070** |

As in the previous chapters, we use an array of metrics to test the performance of the recommenders in terms of ranking accuracy, novelty, diversity, and coverage. For accuracy, we will make special emphasis on normalized Discounted Cumulative Gain (nDCG), although results in terms of Precision (P) and Recall (R) will also be shown (Gunawardana and Shani, 2015). The optimal parameters (shown in Table 7.4) were selected according to P@5 in the scenario when no MCA strategy (SC, from Single City) is applied. The values obtained for the tested parameters are presented in Table 7.3. This means that we use the optimal parameters found in that case and apply the same values for the scenarios when either MCA strategy is used. We have decided to use P instead of nDCG to select the best recommenders because it is more common to use this metric in the area (as shown in Table 3.7). In addition, it should be noted that the relevance in this type of dataset is binary. For novelty, diversity, and coverage we present results for EPC, Gini, and IC. Additionally, unless stated otherwise, all metrics are reported at a cutoff of 5.

### 7.4.4 Performance of the recommenders by using the augmented information through MCA strategies

In this section, we aim to answer the following question: *Are state-of-the-art recommendation algorithms able to exploit augmented information through MCA strategies for venue recommendation?* With this goal in mind, we analyze which POI recommendation algorithms tend to improve or deteriorate their performance under the different MCA strategies proposed.

First, we present in Table 7.5 the results of each recommender (for all families) in each of the eight cities. In this case, no MCA strategies are applied, so both the training and test sets correspond to the target city. The most noticeable result that we observe in this table is the low values obtained by all the recommenders. These low values are mostly due to the high sparsity of the data (see Table 7.2 and Table 7.7 for additional statistics about the cities and aggregation strategies used), together with the fact that we are using a temporal split, which makes the recommendation task even more difficult, since, for instance, a small subset of the few items a user may have in her test set may not appear in the training set at all. Moreover, another aspect that makes it less likely to achieve high accuracy values is that, because of the temporal split, there could be some new users that did not appear in the training set, so in those cases it would be impossible to produce recommendations using any personalized model.

Regarding the recommenders, the AvgDis recommender is the second worst algorithm (after Rnd) and followed by KDE, which evidences that simply modeling the user by the geographical coordinates of her (most frequent) visited venues is not enough to predict her future interactions. Additionally, we observe that the IRenMF approach, even though it remains very competitive, it is not always the optimal recommender across all the cities, somewhat in contradiction with reported experiments in previous works (Liu et al., 2014, 2017). We attribute this behavior to the following reasons: first, its claimed superior performance was only tested using a random split in Liu et al. (2014) instead of a temporal evaluation as we do here; and second, classical recommendation algorithms such as Pop or standard CF approaches were neglected in Liu et al. (2017) which, together with our previous discussion, definitely disturbs such comparisons.

Similarly, RankGeoFM performs poorly, only slightly superior to IB. This could again be explained by the different conditions of the experiments in the original paper (Li et al., 2015a) with respect to ours. For example, in the original paper the authors tested their algorithm only in one city on Foursquare (Singapore) and in two states in Gowalla (Nevada and California), while we have selected 8 different cities on Foursquare. On the other hand, GeoBPR shows very good results for some cities (such as KUA and MOS), although not as good in others (see IST). In this context, we must take into account that for all the recommenders that use matrix factorization techniques, there is a large number of configurable and tunable parameters, making it very difficult (and costly) to find the best configuration in all the situations.

For the rest of the recommenders, we observe that UB is one of the best approaches for most of the cities, usually very close to the optimal one. However, it should be taken into account that all personalized recommenders (except PGN) have less user coverage than the NP family and we may find users in the test set that have not rated any item in the training set. Later, in Section 7.4.6 we shall discuss this aspect again. Additionally, it is interesting to note the relatively high performance of PGN, since it is able to beat the rest of the baselines in many cities, despite its simplicity and the fact that we did not perform any parameter tuning. A possible explanation for this effect is the popularity bias, which is an important component of the PGN algorithm. As we observe in these results, the pure popularity recommender (Pop) obtains a very good performance, being able to surpass other more complex algorithms such as IB or

BPRMF.

We now present in Table 7.6 the results for all cities and the best recommender (according to nDCG@5) of each family when the two proposed MCA strategies are used: most-popular MCA (P), geographical nearest MCA (N), and a MCA (C) based on the cities of the same country. In the first case, for every city, the training set is built by aggregating the training data from the eight most-popular cities; in the second one, the training set is built by taking the nearest 7 cities with respect to the target city, so the number of cities under consideration is comparable to that of P-MCA, and finally for the country-based strategy, the training set is built using the check-ins of all the cities belonging to the same country of the target city. In Appendix A.3 we show the selected cities according to the N-MCA and C-MCA strategies for each target city. In all scenarios the test set corresponds to the one of the target city. It should be noted that we do not show results for the non-personalized family as the MCA strategies do not affect this type of recommenders. The reason for this is that, even though some models may be trained with data from different cities, we only allow to recommend items from the target city; hence, the most popular items of a given city will remain the same regardless of the set of cities used as source domain. Similarly, the random recommender will always give a random score for each item of the target city notwithstanding the aggregation strategy used.

Thus, Table 7.6 shows the relative improvement with respect to the base performance of each algorithm when no MCA strategy is used to augment the training information – that is, the performance of that algorithm when only information from the target city is used, which corresponds to the results shown in Table 7.5. We observe that classical CF algorithms (CF-NN and CF-MF) are able to exploit quite successfully the augmented information using the P-MCA strategy, although the CF-MF often has a lower performance with respect to the base scenario (column SC, from Single City). Moreover, CF-NN always evidence a positive improvement under the N-MCA and C-MCA strategies. We argue this trend for the CF techniques is related to whether the new users from the aggregated cities – it should be noted that the new items found in the augmented training set will never have overlap with the target items, since when using information from other cities the POIs will always be different – have some level of interaction with the target city. In this sense, when combining information from nearby cities it is more likely to find similar users with useful suggestions or learning relevant latent representations more related to the target items. Additionally, having more data available so that the sparsity is reduced does not guarantee better recommendations, since the MF approaches from CF-MF and POI families tend to deteriorate their performance under the P-MCA strategy.

On the other hand, the performance of the Geo family is always worse when using any of the MCA strategies. The reason for this might be quite obvious, since considering other cities to compute a new user's centroid will certainly move such centroid far away from the target city, which is not useful when we are only interested in recommending venues inside of that specific city. Nevertheless, it is interesting to observe that the POI family, which includes a geographical component, also benefits from the MCA strategies

**Table 7.6:** Performance in terms of nDCG@5 when augmented information is used for training. The improvement in performance with respect to Single City (SC) is represented as Δ(%).

| | | | MCA | | | | Δ MCA (%) | |
|---|---|---|---|---|---|---|---|---|
| **City** | **Family** | **SC** | **N** | **C** | **P** | **Δ N (%)** | **Δ C (%)** | **Δ P (%)** |
| IST | Geo | 0.003 | 0.003 | 0.002 | 0.003 | −14.16 | −27.72 | −0.90 |
| | CF-NN | 0.073 | 0.073 | 0.075 | **0.073** | 0.32 | 3.83 | 0.35 |
| | CF-MF | 0.070 | 0.071 | 0.073 | 0.068 | 1.98 | 4.79 | −3.41 |
| | POI | **0.074** | **0.076** | **0.077** | 0.072 | **2.88** | 3.46 | −3.25 |
| | H-POI | 0.067 | 0.068 | 0.071 | 0.068 | 1.61 | **6.17** | **0.91** |
| JAK | Geo | 0.004 | 0.003 | 0.003 | 0.004 | −19.73 | −24.52 | **1.83** |
| | CF-NN | 0.070 | 0.075 | **0.078** | **0.071** | 6.68 | **10.46** | 0.37 |
| | CF-MF | 0.066 | 0.070 | 0.072 | 0.060 | 6.31 | 9.34 | −8.50 |
| | POI | **0.071** | **0.076** | 0.077 | 0.064 | **8.16** | 9.39 | −8.63 |
| | H-POI | 0.067 | 0.070 | 0.072 | 0.068 | 4.75 | 6.86 | 0.63 |
| KUA | Geo | 0.004 | 0.003 | 0.003 | 0.004 | −37.28 | −41.48 | −0.73 |
| | CF-NN | 0.073 | **0.076** | **0.078** | **0.073** | 4.13 | 6.20 | **0.29** |
| | CF-MF | 0.066 | 0.075 | 0.077 | 0.065 | **13.79** | **17.14** | −1.55 |
| | POI | **0.073** | 0.075 | 0.075 | 0.062 | 2.20 | 2.58 | −15.48 |
| | H-POI | 0.070 | 0.072 | 0.073 | 0.070 | 2.12 | 3.65 | 0.02 |
| MEX | Geo | 0.006 | 0.005 | 0.004 | 0.005 | −13.79 | −26.19 | −1.37 |
| | CF-NN | 0.044 | 0.045 | **0.047** | **0.045** | 1.62 | **7.17** | 1.24 |
| | CF-MF | **0.047** | **0.045** | 0.045 | 0.037 | −5.03 | −3.95 | −22.07 |
| | POI | 0.046 | 0.044 | 0.043 | 0.031 | −3.14 | −4.64 | −31.44 |
| | H-POI | 0.043 | 0.044 | 0.046 | 0.044 | **2.21** | 7.13 | **1.33** |
| MOS | Geo | 0.006 | 0.005 | 0.005 | 0.006 | −13.91 | −22.09 | −0.56 |
| | CF-NN | 0.037 | 0.038 | **0.041** | **0.037** | **2.53** | 10.83 | **0.33** |
| | CF-MF | 0.039 | 0.039 | 0.041 | 0.036 | 1.84 | 6.11 | −7.70 |
| | POI | **0.044** | **0.041** | 0.038 | 0.022 | −7.60 | −13.54 | −49.03 |
| | H-POI | 0.032 | 0.033 | 0.036 | 0.032 | 0.78 | **10.87** | 0.06 |
| SAN | Geo | 0.005 | 0.003 | 0.004 | 0.005 | −36.08 | −34.94 | −1.40 |
| | CF-NN | 0.053 | **0.060** | **0.064** | 0.054 | 12.98 | 19.34 | 0.86 |
| | CF-MF | 0.050 | 0.060 | 0.062 | 0.046 | **20.21** | **24.22** | −7.87 |
| | POI | 0.052 | 0.057 | 0.055 | 0.038 | 9.46 | 6.74 | −27.19 |
| | H-POI | **0.054** | 0.059 | 0.060 | **0.055** | 8.68 | 10.16 | **1.14** |
| SAO | Geo | 0.008 | 0.007 | 0.006 | 0.008 | −12.00 | −19.03 | −0.49 |
| | CF-NN | 0.049 | 0.056 | **0.060** | 0.049 | **15.42** | **23.91** | −0.22 |
| | CF-MF | 0.048 | 0.056 | 0.058 | 0.047 | 15.23 | 19.78 | −2.09 |
| | POI | 0.052 | 0.047 | 0.040 | 0.031 | −10.26 | −22.55 | −40.43 |
| | H-POI | **0.057** | **0.057** | 0.058 | **0.057** | 0.41 | 2.06 | **0.53** |
| TOK | Geo | 0.005 | 0.003 | 0.003 | 0.004 | −42.38 | −45.74 | −3.43 |
| | CF-NN | 0.069 | 0.073 | 0.074 | 0.069 | **5.38** | **7.41** | −0.20 |
| | CF-MF | 0.066 | 0.066 | 0.065 | 0.062 | 0.93 | −0.63 | −6.13 |
| | POI | 0.068 | 0.071 | 0.070 | 0.064 | 3.90 | 2.89 | −6.30 |
| | H-POI | **0.070** | **0.073** | **0.074** | **0.070** | 4.89 | 6.15 | **−0.16** |

**Table 7.7:** Statistics for training splits of the reported cities and MCA strategies used. Notation as in Table 7.2. The last two columns show the amount of information in the MCA strategies that was already included in the target city $C$.

| **City** | | **U** | **I** | $\mathbf{C_R}$ | $\mathbf{C_r}$ | $\frac{\mathbf{C_R}}{\mathbf{\lvert U \rvert \lvert I \rvert}}$ | $\frac{\mathbf{C_r}}{\mathbf{\lvert U \rvert \lvert I \rvert}}$ | $\frac{\mathbf{C_R(C)}}{\mathbf{C_R}}$ | $\frac{\mathbf{C_r(C)}}{\mathbf{C_r}}$ |
|---|---|---|---|---|---|---|---|---|---|
| IST | SC | 23k | 40k | 668k | 392k | 0.072% | 0.042% | 100.0% | 100.0% |
| | N-MCA | 26k | 51k | 784k | 458k | 0.060% | 0.035% | 85.21% | 85.70% |
| | C-MCA | 34k | 88k | 1.2M | 691k | 0.039% | 0.023% | 56.62% | 56.79% |
| JAK | SC | 11k | 39k | 347k | 182k | 0.082% | 0.043% | 100.0% | 100.0% |
| | N-MCA | 16k | 80k | 678k | 354k | 0.052% | 0.027% | 51.13% | 51.52% |
| | C-MCA | 19k | 104k | 861k | 441k | 0.044% | 0.023% | 40.31% | 41.40% |
| KUA | SC | 11k | 28k | 312k | 170k | 0.102% | 0.056% | 100.0% | 100.0% |
| | N-MCA | 13k | 63k | 642k | 341k | 0.078% | 0.042% | 48.62% | 49.93% |
| | C-MCA | 16k | 87k | 856k | 438k | 0.061% | 0.031% | 36.43% | 38.79% |
| MEX | SC | 7k | 27k | 285k | 143k | 0.144% | 0.073% | 100.0% | 100.0% |
| | N-MCA | 8k | 35k | 344k | 172k | 0.119% | 0.059% | 82.80% | 83.43% |
| | C-MCA | 11k | 55k | 506k | 248k | 0.084% | 0.041% | 56.34% | 57.91% |
| MOS | SC | 7k | 29k | 304k | 153k | 0.150% | 0.075% | 100.0% | 100.0% |
| | N-MCA | 7k | 33k | 328k | 164k | 0.137% | 0.068% | 92.58% | 93.38% |
| | C-MCA | 11k | 64k | 584k | 279k | 0.081% | 0.039% | 52.08% | 54.64% |
| SAN | SC | 6k | 25k | 324k | 130k | 0.211% | 0.085% | 100.0% | 100.0% |
| | N-MCA | 7k | 36k | 433k | 173k | 0.168% | 0.067% | 74.92% | 75.39% |
| | C-MCA | 7k | 38k | 455k | 182k | 0.162% | 0.065% | 71.23% | 71.72% |
| SAO | SC | 7k | 28k | 294k | 120k | 0.145% | 0.059% | 100.0% | 100.0% |
| | N-MCA | 11k | 50k | 491k | 195k | 0.089% | 0.035% | 59.76% | 61.66% |
| | C-MCA | 21k | 117k | 1.1M | 446k | 0.047% | 0.018% | 25.65% | 26.90% |
| TOK | SC | 9k | 29k | 328k | 164k | 0.133% | 0.067% | 100.0% | 100.0% |
| | N-MCA | 11k | 60k | 631k | 301k | 0.097% | 0.046% | 51.97% | 54.40% |
| | C-MCA | 11k | 70k | 705k | 337k | 0.088% | 0.042% | 46.48% | 48.47% |
| All | P-MCA | 80k | 245k | 2.9M | 1.5M | 0.015% | 0.007% | - | - |

in some scenarios. For example, in IST, JAK, KUA, SAN, and TOK using both the N-MCA and C-MCA strategies, these algorithms obtain a better performance than in the Single City scenario. However, when using the P-MCA strategy, the performance of this family is always worse. This result is particularly interesting since in some works researchers perform experiments using datasets with information from several cities grouped together (Zhang and Chow, 2013, Gao et al., 2015b, Manotumruksa et al., 2019a), and as we can observe, this can affect negatively the performance of some models.

From the perspective of the MCA strategies, we observe that the performance improvements obtained when using the P-MCA strategy is usually negligible. In general, most of the improvements when using this strategy are very close to zero and, for many of the city-recommender family combinations, extremely negative. At the same time, N-MCA and C-MCA usually produce larger improvements with less training data involved since those cities that belong to the same country or are geographically nearest to the target city always include less check-ins than the originally selected cities, which were the most popular ones in our dataset; see Table 7.7 for more details). Even if these results seem to confirm that better data is more useful than more data, we now analyze these effects in more detail.

To properly understand which of the MCA strategies are more suitable to augment the data available for recommendation, we include in Table 7.7 the sparsity of each resulting augmented training set, together with the amount of information already included in the target city with respect to each aggregation (last two columns). Based on these statistics, we infer that the user overlap of each MCA strategy (reported in Table 7.1) is not the only factor to consider in the success of the proposed data augmentation approaches. For instance, the three cities with more user overlap (IST, MEX, and MOS, with more than 82%) also show high ratios of $\mathbf{C_r}(\mathbf{C})/\mathbf{C_r}$ when N-MCA is used, which means that such aggregation strategy incorporates very little additional information with respect to the original training data, resulting in a more sparse dataset (to be expected from any MCA strategy, due to the large amount of new items and users added) but where most of the interactions come from the same city. Going back to the results in Table 7.6, we observe that the N-MCA strategy is less useful (for the CF-MF approaches in particular, but also for the classical CF-NN methods) essentially when such ratio is too large, since this means that the original and augmented training splits are very similar and, hence, the performance improvement would be minimal.

Thus, we are able to answer our first research question: we have seen that some classic recommenders (i.e., nearest neighbors and matrix factorization) are able to benefit from the augmented information if the cities used to define the Multi-City Aggregation strategy are selected properly. In particular, we conclude that selecting cities by proximity (and, as a special case, by country) has a greater benefit than selecting them by the amount of information they contain (popularity). However, other approaches more tailored for venue recommendation may decrease their performance when exploiting knowledge from other domains. This is especially noticeable in strategies that give great importance to geographical influence whenever the number of common users is negligible, as in the P-MCA proposed method. This opens up the possibility of using alternative cross-domain techniques that may benefit other algorithms that are not so dependent on user overlap, such as item similarity models like SLIM, FISM, or those based on embeddings Ning and Karypis (2011), Kabbur et al. (2013). However, we leave as future work the analysis of these similarity models for POI recommendation together with alternative data augmentation techniques.

### 7.4.5 What is the impact of venue recommenders on different groups of users?

As we have already mentioned previously, in the tourism domain it is possible to characterize different types of users. In particular, we define two groups following the work of Choudhury et al. (2010): tourists and locals. More specifically, we have established that those users whose check-ins exist in the same city for more than 21 days are considered locals, and the rest are considered tourists. However, to avoid noisy or non-human behavior, we filtered out in the test set for both groups those users who have performed three or more consecutive check-ins with a temporal difference smaller than 60 seconds,

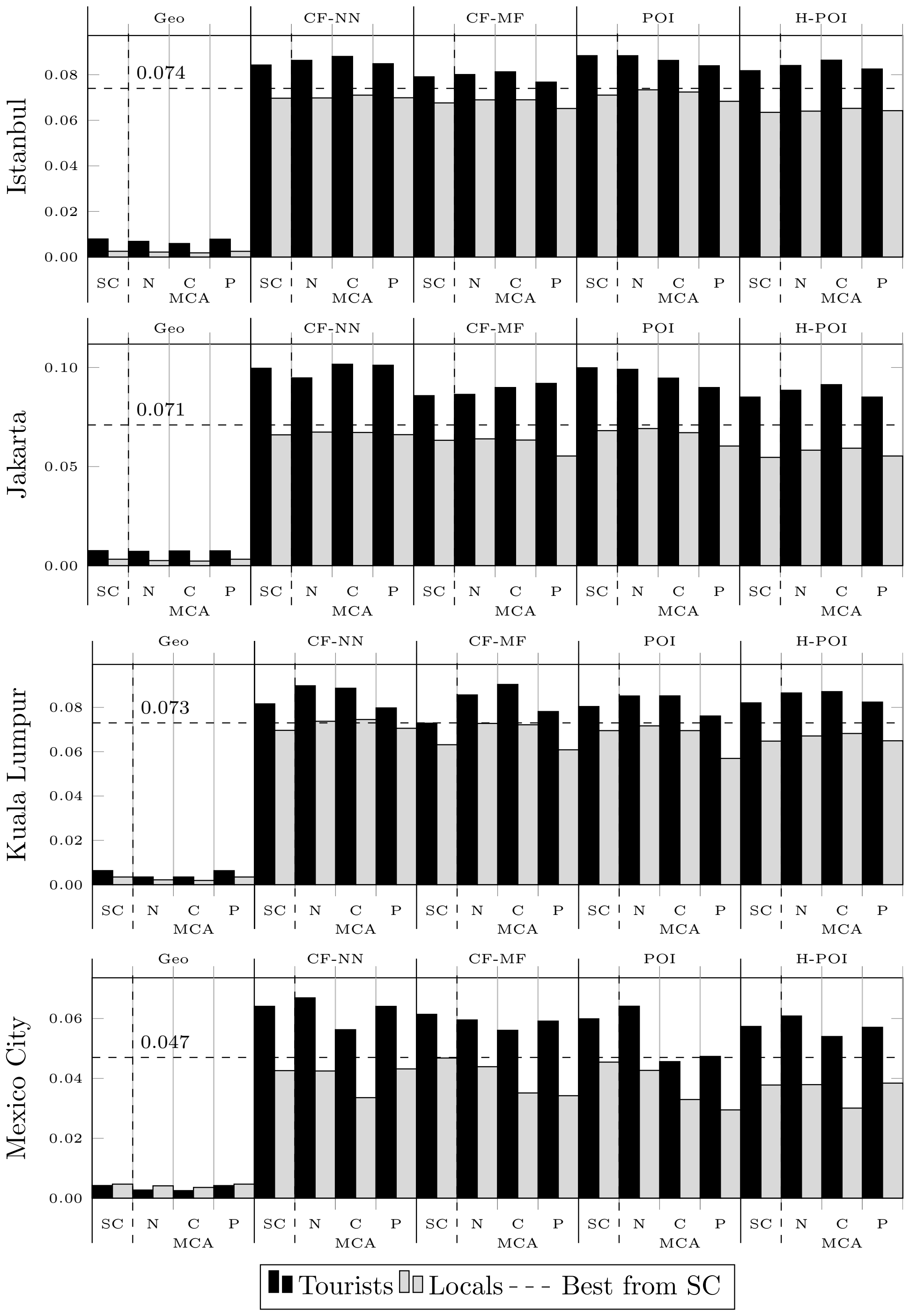


**Figure 7.1:** Results of tourists and local users in Istanbul, Jakarta, Kuala Lumpur and Mexico City in terms of nDCG@5. Labels SC, N, and P in the x-axis represent the single-city (baseline) configuration, and N-MCA and P-MCA strategies respectively. Dashed line indicates the performance of the best recommender when all users are considered under the SC configuration, as shown in Table 7.5.

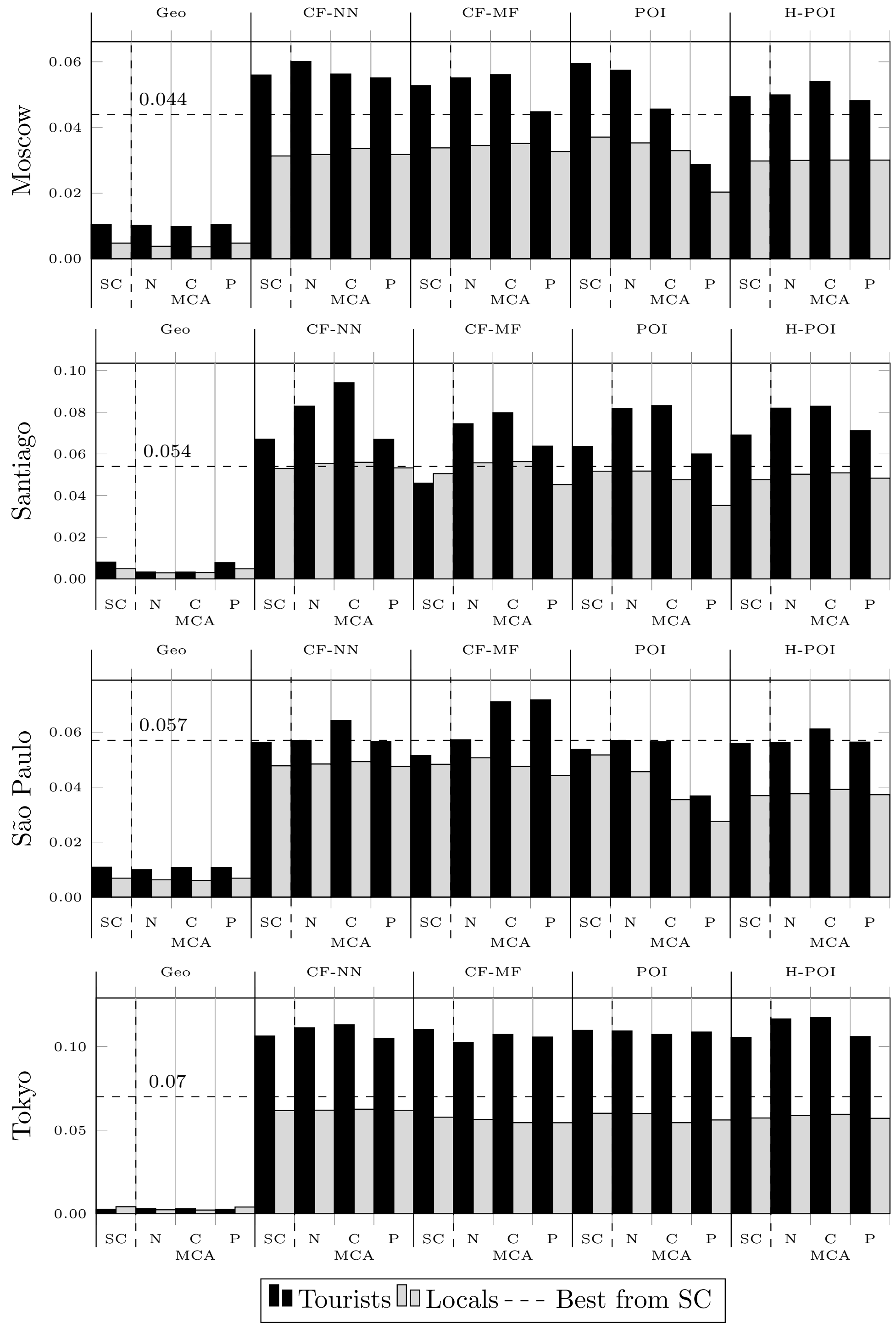


**Figure 7.2:** Results of tourists and local users in Moscow, Santiago, São Paulo and Tokyo in terms of nDCG@5. Rest of notation as in Figure 7.1.

since they can be considered bots as in previous works (Palumbo et al., 2017). These so-called bots do not count either as tourists or locals (hence, in this section they are completely ignored), even though their performance is considered whenever the global performance is measured (that is, in those tables or figures that appear in the rest of the chapter). However, these unusual consecutive check-ins are not always caused by bots. It may also be due to bugs in the application when recording interactions. Therefore, we will classify these users as outliers. Besides, as the data of these users may be useful for the rest of the recommenders, we keep the interactions of these users in the training set.

Based on this, in Figures 7.1 and 7.2 we contrast the results of each type of users in terms of nDCG@5 for all cities when no MCA strategy is used (SC) and when the MCA strategies presented in the previous experiment are used (N, for N-MCA, C, for C-MCA, and P, for P-MCA). We observe that for all cases (except in the Geo family in Mexico City and Tokyo), tourists obtain significantly better results than locals. We hypothesize this may be attributed to tourists having a more similar behavior in common among the users in the same group: for example, when someone visits Paris, regardless of where they come from, they are more likely to visit touristic venues such as the Eiffel Tower or the Louvre museum rather than some suburban neighborhoods in the city. On the other hand, locals are probably more heterogeneous, and hence, more different behaviours are aggregated in the same group, making it much more difficult to the recommendation algorithms to guess their preferences correctly. Besides, the number of tourist users, because of its definition, tends to be smaller than local users, which helps to obtain more coherent user groups.

Consistent with the results discussed in the previous section, the MCA strategy by proximity obtains better performance and, in general, improves the base results more than the strategy based on popularity for both types of users in most cities. There are some exceptions, as in the case of São Paulo for tourists, where the performance improvement of the P-MCA strategy is striking. However, the general trend is that this strategy is outperformed by both the N-MCA and C-MCA; we note even some cases where they produce worse performance than the base scenario, for example in Moscow for most recommendation families or in Kuala Lumpur for the CF-NN and POI families.

Since we observe no different behavior with respect to the user groups between using MCA strategies or not, we come to the conclusion that venue recommenders evidence a strong bias towards tourist users, in particular, this group of users seem to be much easier to recommend. Hence, we summarize that every recommendation family except the basic geographical algorithms improve their results when analyzing the subset of tourists in isolation. In agreement with the previous research question, the N-MCA and C-MCA strategies are also beneficial in this case, obtaining much better results, in general, than P-MCA.

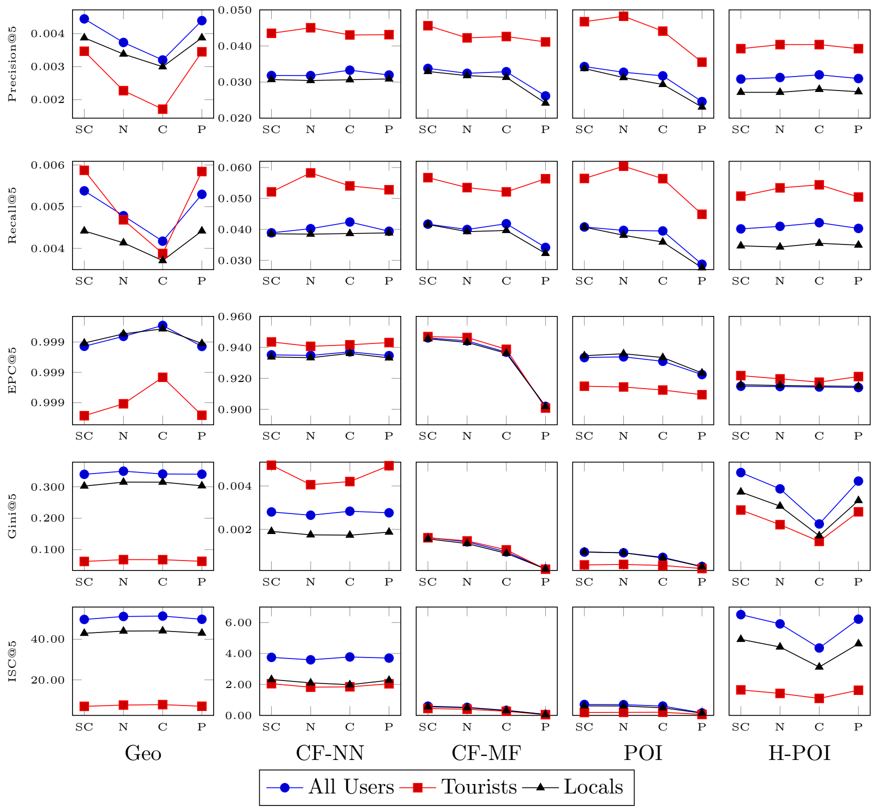


**Figure 7.3:** Results of tourists (10.42% of the users) and local (71.60% of the users) users in Mexico City in terms of accuracy metrics (Precision, Recall), novelty (EPC) and diversity (Gini, ISC). Labels in the x-axis as in Figure 7.1, together with C representing the C-MCA strategy.

### 7.4.6 Performance analysis on beyond-accuracy evaluation dimensions

An important aspect that is sometimes ignored when evaluating recommendation algorithms is finding a good balance between novelty, diversity, and accuracy (Gunawardana and Shani, 2015); this is what we analyze in this section. For this, we present in Figures 7.3, 7.4 and 7.5 the results for the recommendation families used before using all the metrics presented in Section 7.4.3 for the cities of Mexico City, Santiago and Tokyo, that is, Precision, Recall, EPC for novelty, and Gini and IC for diversity. We complement these results with user coverage of all cities in Table 7.8.

Our decision to select these three cities was because they evidence a different behavior with respect to the ratio $\mathbf{C_r(C)/C_r}$ when comparing the N-MCA and C-MCA strategies: whereas TOK and SAN obtains a very similar ratio for both strategies with a percentage around 50% and 73% respectively, MEX presents substantially different values (see Table 7.7). Moreover, inspired by the very positive results found so far for

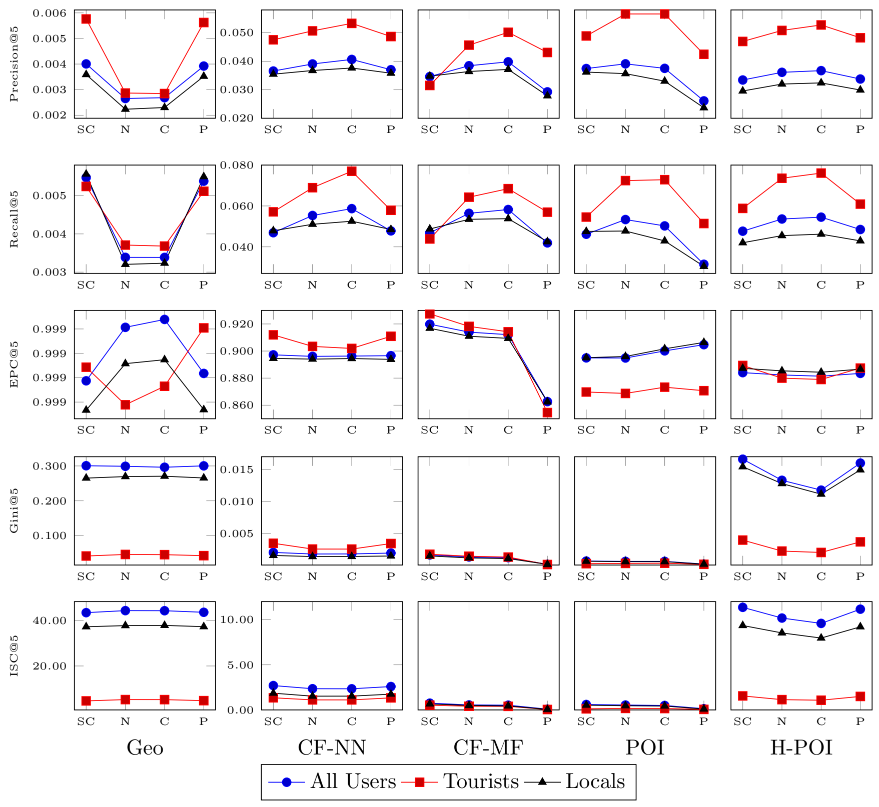


**Figure 7.4:** Results of tourists (7.62% of the users) and local (72.43% of the users) users in Santiago; notation as in Figure 7.3.

N-MCA, we will also consider all the cities that belong to the same country as the target city, and name it C-MCA (C), from country-based N-MCA.

Based on the results reported in Figures 7.3, 7.4 and 7.5, and considering all users, one observation that may draw our attention is that the algorithms of the Geo family have higher novelty and diversity than the other families. This is because these recommenders are based solely on recommending POIs that are close to the target user, ignoring other factors like the popularity of the POIs (something that, after observing the results obtained by the rest of the recommenders, seems to confirm the popularity bias of the data). Besides, this type of recommender is the worst in terms of accuracy, as shown by the nDCG metric in the previous figures and with Precision and Recall in the ones shown here, and it is well-known that there is typically a tradeoff between accuracy and novelty/diversity.

Regarding the other recommenders, we observe that the H-POI family tends to obtain better diversity results than the other families, although the novelty of its recommendations is usually lower. This can perhaps be explained by the fact that the best algorithm of this family is always the PGN recommender, which combines popularity and collaborative filtering with the distance between the POIs. The first two

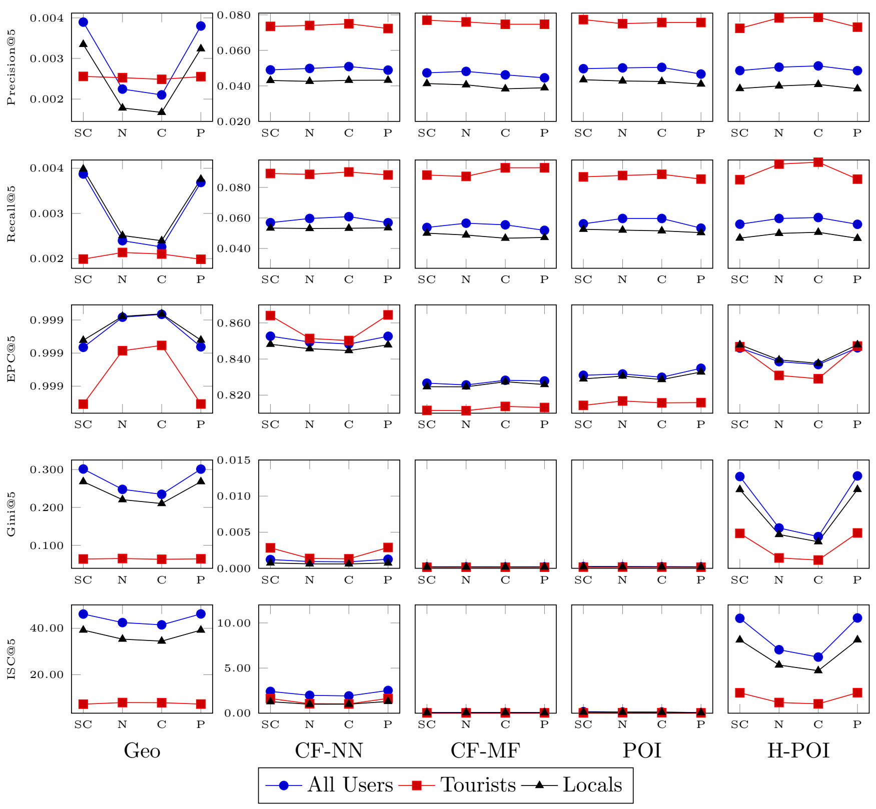


**Figure 7.5:** Results of tourists (9.32% of the users) and local (62.44% of the users) users in Tokyo; notation as in Figure 7.3.

contributions reduce the novelty and diversity, but the latter, as we have seen in the Geo family, increases both dimensions so it makes sense that this approach may improve to some extent either dimensions. It is also worth considering that, in general, POI and CF-MF families achieve lower levels of novelty and diversity than the rest of the algorithms. It must be taken into account that both families use some kind of matrix factorization techniques, as the GeoBPR and IRenMF. Low diversity values are indicative that very few different items are actually recommended, whereas low novelty values suggest in this case that most of the recommended POIs are those that have been visited by more users in the training set (popular items). This means that there is a significant popularity bias in the recommendations provided by these families, which is actually corroborated because their performance in terms of relevance is also high (Jannach et al., 2015). On the other hand, the behavior of CF-MF in terms of EPC (novelty) might be reinforced by another aspect. In Mexico City and Santiago we observe that the novelty for the P-MCA strategy decreases steadily. We believe this might be caused because in this type of strategy, we are increasing considerably the number of items and users in the system, but at the same time we only recommend POIs from the destination city; hence, the latent factors of those POIs that are more

**Table 7.8:** User coverage obtained by the recommenders when augmented information is used for training. The improvement in user coverage with respect to SC is represented as Δ(%).

| City | Family | SC | N | C | P | Δ N (%) | Δ C (%) | Δ P (%) |
|---|---|---|---|---|---|---|---|---|
| IST | Geo | 81.65 | 83.82 | 87.65 | 81.74 | **2.66** | **7.36** | **0.12** |
| | CF-NN | 84.58 | 86.38 | 89.92 | 84.61 | 2.12 | 6.32 | 0.03 |
| | CF-MF | 85.10 | 86.93 | 90.25 | 85.19 | 2.14 | 6.05 | 0.10 |
| | POI | 85.10 | 86.93 | 90.25 | 85.19 | 2.14 | 6.05 | 0.10 |
| | H-POI | **100.00** | **100.00** | **100.00** | **100.00** | 0.00 | 0.00 | 0.00 |
| JAK | Geo | 86.81 | 91.63 | 92.81 | 87.73 | 5.56 | **6.92** | **1.06** |
| | CF-NN | 88.05 | 92.97 | 93.95 | 88.54 | **5.59** | 6.69 | 0.55 |
| | CF-MF | 89.51 | 93.49 | 94.51 | 90.43 | 4.45 | 5.59 | 1.03 |
| | POI | 89.51 | 93.49 | 94.51 | 90.43 | 4.45 | 5.59 | 1.03 |
| | H-POI | **100.00** | **100.00** | **100.00** | **100.00** | 0.00 | 0.00 | 0.00 |
| KUA | Geo | 80.77 | 88.91 | 91.03 | 81.36 | **10.08** | **12.71** | 0.74 |
| | CF-NN | 85.01 | 90.79 | 92.31 | 85.37 | 6.80 | 8.58 | 0.42 |
| | CF-MF | 85.30 | 91.06 | 92.56 | 85.95 | 6.76 | 8.51 | **0.77** |
| | POI | 85.30 | 91.06 | 92.56 | 85.95 | 6.76 | 8.51 | 0.77 |
| | H-POI | **100.00** | **100.00** | **100.00** | **100.00** | 0.00 | 0.00 | 0.00 |
| MEX | Geo | 88.58 | 90.90 | 93.76 | 88.75 | **2.62** | **5.85** | 0.20 |
| | CF-NN | 91.05 | 92.81 | 95.07 | 91.11 | 1.93 | 4.41 | 0.06 |
| | CF-MF | 91.27 | 93.04 | 95.34 | 91.46 | 1.94 | 4.46 | **0.21** |
| | POI | 91.27 | 93.04 | 95.34 | 91.46 | 1.94 | 4.46 | 0.21 |
| | H-POI | **100.00** | **100.00** | **100.00** | **100.00** | 0.00 | 0.00 | 0.00 |
| MOS | Geo | 85.27 | 85.83 | 89.55 | 85.74 | **0.67** | **5.02** | 0.56 |
| | CF-NN | 87.57 | 88.00 | 91.55 | 87.41 | 0.49 | 4.55 | −0.18 |
| | CF-MF | 88.07 | 88.50 | 92.03 | 88.59 | 0.49 | 4.50 | **0.59** |
| | POI | 88.07 | 88.50 | 92.03 | 88.59 | 0.49 | 4.50 | 0.59 |
| | H-POI | **100.00** | **100.00** | **100.00** | **100.00** | 0.00 | 0.00 | 0.00 |
| SAN | Geo | 90.54 | 94.19 | 94.88 | 91.23 | **4.03** | **4.79** | **0.76** |
| | CF-NN | 92.06 | 95.00 | 95.64 | 92.26 | 3.19 | 3.89 | 0.21 |
| | CF-MF | 92.28 | 95.27 | 95.91 | 92.94 | 3.24 | 3.93 | 0.72 |
| | POI | 92.28 | 95.27 | 95.91 | 92.94 | 3.24 | 3.93 | 0.72 |
| | H-POI | **100.00** | **100.00** | **100.00** | **100.00** | 0.00 | 0.00 | 0.00 |
| SAO | Geo | 82.21 | 85.81 | 90.44 | 82.62 | 4.37 | 10.01 | **0.50** |
| | CF-NN | 83.33 | 87.84 | 92.02 | 83.46 | **5.41** | **10.43** | 0.15 |
| | CF-MF | 85.04 | 88.51 | 92.69 | 85.37 | 4.08 | 8.99 | 0.39 |
| | POI | 85.04 | 88.51 | 92.69 | 85.37 | 4.08 | 8.99 | 0.39 |
| | H-POI | **100.00** | **100.00** | **100.00** | **100.00** | 0.00 | 0.00 | 0.00 |
| TOK | Geo | 87.75 | 92.68 | 93.64 | 88.01 | **5.61** | **6.71** | **0.30** |
| | CF-NN | 90.32 | 94.15 | 94.78 | 90.32 | 4.24 | 4.94 | 0.00 |
| | CF-MF | 90.52 | 94.29 | 94.96 | 90.78 | 4.17 | 4.91 | 0.29 |
| | POI | 90.52 | 94.29 | 94.96 | 90.78 | 4.17 | 4.91 | 0.29 |
| | H-POI | **100.00** | **100.00** | **100.00** | **100.00** | 0.00 | 0.00 | 0.00 |

popular are updated more frequently. If we look at the Tokyo results, the novelty of this strategy does not decrease so much because it is already very low for all strategies. This can be attributed to the fact that in Tokyo the optimal CF-MF algorithm is the BPRMF, which is not so sensitive to the previous behavior, while in the other two cities, is the HKV (see Table 7.5).

When we analyze the results by types of users (tourists or locals), it is important to consider the third type of users that is being ignored in the figure, the bots, together with the new users (in the test set) that do not appear in the training set and, thus, do not fit in any category. Because of this, we include in the caption of each figure the percentage of tourists and locals in the test set of the corresponding city, so the rest of the users would be labeled as bots or no belonging to any group. This piece of information is important in this case because we show together information from all users and separated by user group. Based on these results, we corroborate that in terms of accuracy metrics (Precision and Recall) tourists achieve better performance than locals, except in the Geo family.

To better understand the rationale behind these results, we have analyzed in more detail the biases in the data and in the recommendations produced by the algorithms, and discovered that tourists are more likely to visit popular POIs (according to the training set) than locals, as evidenced by their check-ins in the test set. This is also observed in the figures, due to the higher novelty and diversity values achieved by most recommendation families for the local users (except for the CF-NN). As discussed before, this could be attributed to several reasons. Firstly, there are more local users than tourists, so it is more likely that there are more different recommended items for this type of user. Secondly, it is more likely that tourists tend to visit the most touristic venues in a city, but this should be considered in combination with the fact that locals are probably visiting a larger variety of POIs, since they spread more evenly across the city and throughout longer periods of time. This could be an explanation as to why the novelty in locals tend to be higher than in tourists.

One dimension that deserves further attention is user coverage. This measurement, as reported in Table 7.8, accounts for the number of users that have received at least one recommendation. In these results, the first observation we make is that the H-POI is the only recommender family with full user coverage, something that does not change when using any aggregation strategy; the reason for this is that the best recommender in this family is the PGN and one of the algorithms exploited by this hybrid recommender is based on popularity, which is a non-personalized algorithm and, hence, also has full user coverage by design. A more interesting result that emerges from this analysis is that the C-MCA strategy always improves the coverage of the recommenders, in fact, according to the column depicting the relative improvements with respect to results from the single city, this strategy produces the largest improvements for every family in all the cities. This is a very important outcome, since together with the results shown in Figures 7.3, 7.4 and 7.5, where this strategy obtains better or equal accuracy results than N-MCA for most of the families in every city, it means that it is able to improve the results for more users in the system, simply by integrating carefully selected additional information. This process, in any case, would be achieved at a lower cost than the P-MCA strategy, hence, allowing more efficient computations. In particular, this allows us to create a single training set containing the check-ins of the cities we need and make recommendations from this training, instead of making an independent set by each city and hence allowing more efficient computations. Nevertheless, we can observe an interesting result in the city of Moscow. If we analyze the user coverage of this city

**Table 7.9:** Performance in terms of nDCG@5 of the Popularity recommender in all cities in both Tourists and Locals.

| City | All Users | Tourists | Locals | Δ Tourists (%) | Δ Locals (%) |
|---|---|---|---|---|---|
| Istanbul | 0.054 | 0.064 | 0.048 | 19.04 | −9.77 |
| Jakarta | 0.066 | 0.091 | 0.053 | 38.33 | −19.92 |
| Kuala Lumpur | 0.066 | 0.077 | 0.060 | 17.34 | −8.46 |
| Mexico City | 0.041 | 0.059 | 0.034 | 45.69 | −15.70 |
| Moscow | 0.027 | 0.037 | 0.026 | 34.02 | −4.48 |
| Santiago | 0.051 | 0.067 | 0.044 | 30.47 | −13.21 |
| São Paulo | 0.053 | 0.061 | 0.031 | 14.85 | −40.33 |
| Tokyo | 0.069 | 0.106 | 0.056 | 53.48 | −18.73 |

more in detail, we can see how the user coverage decreases in the P-MCA strategy for CF-NN recommenders. This behavior, although it may seem counter intuitive, is due to the fact that those algorithms that work with similarities between users or items (users, in this case) might obtain a large number of neighbors with the same degree of similarity, some of them coming from the aggregated cities and with potentially no check-ins in the target city. In this sense, and according to these results, by using the P-MCA strategy it is more likely that we may find new neighbors with near-zero overlap with items in the target city that are, hence, not able to recommend any POI there, thus, reducing the coverage of such algorithms

As a summary, we have observed that the N-MCA strategy is the safest one both in terms of accuracy and beyond-accuracy metrics, although C-MCA obtains very similar results while improving the user coverage, hence, impacting positively to more users. These strategies also show good results for tourists, although all users get some kind of improvement with these approaches. Regarding the effect in the groups of users, tourists are positively affected in terms of accuracy but negatively for other dimensions such as novelty and diversity.

## 7.5 Discussion

In this chapter we have covered RG5: *Improve the performance of the algorithms in the POI recommendation domain.* According to the presented results, applying Multi-City Aggregation strategies to augment the data available in venue recommendation can improve the results obtained in some situations, although their effect is not as great as one might expect (mainly due to the temporal split we used and the dataset being too sparse). Nevertheless, since we explored some basic recommendation techniques across a wide range of algorithmic families, these results are promising and may open the door to debate about the importance of the geographical distribution of the check-ins in evaluation. First of all, because we have seen that some algorithms are able to make better recommendations (in some cases, up to a 20% improvement), and in some situations – mostly under the C-MCA strategy – the user coverage is enhanced; however, further analysis should be done to properly understand the impact of such improvements in other evaluation dimensions, such as novelty or diversity, and how it

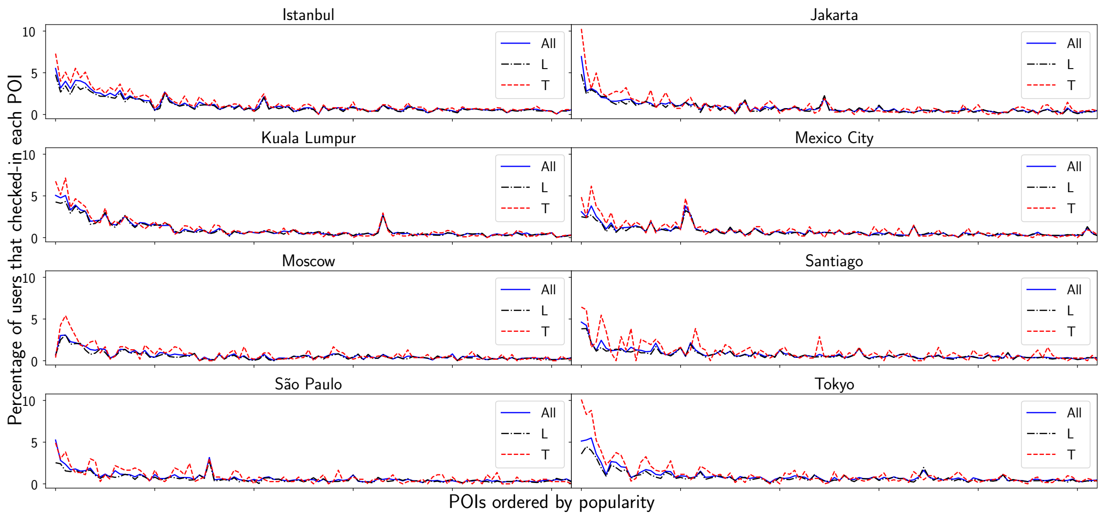


**Figure 7.6:** Popularity bias in the eight selected cities. Each plot shows the percentage of users belonging to each group who have a check-in in the test set for each corresponding item. Items are sorted according to their popularity in the training set.

generalizes to different cities. Similarly, the N-MCA strategy has generally returned cities belonging to the same country with respect to the target city (except in the case of Santiago, where one of its 7 closest cities was Cordoba, a city of Argentina). This, in particular, should be further analyzed in the future, since at the moment the reasons behind the difference between C-MCA and N-MCA is not clear; hence, more experiments should be conducted where cities with nearby cities from other countries are included, to properly assess the performance difference between these strategies

Secondly, due to the well-known popularity bias (Jannach et al., 2015), such a simple technique could outperform other methods like IB, KDE, or AvgDis (see Table 7.5), and it has resulted in a very positive component to be integrated in a hybrid algorithm (i.e., PGN), even though this type of baseline is usually ignored in POI recommendation literature. Thirdly, we have observed that it is not the same to train the recommenders with interactions of a certain city as training them with the check-ins of a whole country. Hence, POI recommendation proposals that are trained using information of specific regions may not be comparable to others trained with data from around the world. In fact, we already explored this issue in Chapter 3, as in Table 3.5 we showed there were researchers that did not perform any kind of region split.

At the same time, we have observed that if we distinguish between two types of users (tourists and locals), almost every recommendation algorithm produces very different results to each user group. To further understand this, we now analyze in more detail the effect of popularity bias in our experimental settings. First, in Table 7.9 we show the results of the Popularity recommender in the Single City (SC) configuration. In that table we show the results obtained taking into account all users in the test set as well as the results for the tourists and locals. The last two columns represent the change in performance (as a percentage, negative or positive depending on whether the performance improved or decreased) obtained by each group of users with respect to the value obtained by considering that all users belong to the same group. As we can observe, tourists tend to obtain a result between a 14% and a 50% higher than the rest of the users. This seems to confirm that tourist users tend to visit more popular POIs than locals. To further view this effect in detail, in Figure 7.6 we show the top 1% of the most popular POIs of every city that appear in the test set with the percentage of users (of each group) that checked-in in that POI. In this figure we can see how the most popular POIs in general receive more visits from tourist users (relatively) than from local users. This makes sense since, as we have indicated before, for a user to be considered a tourist she has to perform check-ins in the city for at most 21 days, so it is more likely that many of those tourists did not have enough time to visit the most popular POIs in the training set and hence they visit them in the test set.

This evidences a general trend or systematic bias, since it is much easier to make relevant recommendations to tourists due to the type of POIs they usually visit. In particular, this result – which is, to the best of our knowledge, novel in the area – would open up several possibilities in terms of deciding how many resources should be devoted to each user group.

Additionally, a negative result we observed is that when the distance between the venues is considered in the recommendation algorithm, augmenting the available information through MCA strategies can be counterproductive in terms of accuracy, although other dimensions might be benefited from such augmentation. This effect is not conclusive for the two types of users considered, although we have observed a negative trend for tourists, whose diversity and novelty values tend to be much lower than those for local users, in particular when MCA strategies are exploited.

It should be noted that some of our results are consistent with those discussed in Sahebi and Brusilovsky (2015). The authors found that there are specific experiments where cross-domain recommendation works worse than classic recommendation, even though in general it behaves better or as good as strategies where cross-domain is not used (single-domain). However, only comparisons between single- and cross-domain approaches on three different algorithms and without considering any temporal split were presented in that study; hence, our work helps on generalizing the conclusions obtained in such paper.

We want to emphasize that considering information from different cities (understood as different domains), despite being computationally more expensive, has a clear advantage: such system would only need to train once whenever recommendations are required for any of the cities included in the MCA strategy, whereas considering each city as an isolated training domain (when no MCA strategy is used) only allows to generate recommendations for a single city; hence, the recommendation model built in such a way can be re-used more often in the former case, at the expense of being more expensive (although this would depend on the actual strategy considered) in terms of memory and time consumption. This conclusion may help other researchers in the area in order to apply a more favorable data preprocessing for the POI recommendation models that they are developing.

Nonetheless, as we have shown here, if the MCA strategy is generated based on the *right* cities, significant performance improvements can be achieved, not always by selecting the cities with more information but those that are closer and more likely to have overlap between their users, probably because they are culturally related and share similar mobility patterns (Yang et al., 2016).

# Part III

# Conclusions

# 8

# Conclusions and Future Work

Recommender Systems have been progressively integrated into an increasing number of applications, to the extent that they are now indispensable for handling the vast amount of data available in a large number of technological companies. As a result, research into these systems is crucial in order to improve the user experience by providing them with better recommendations, adapting them to the current context of the users.

This thesis has focused on two main topics: the integration of temporal and sequential information in Recommender Systems (both in the recommendation and evaluation steps) and the detailed study of the problem of Point-of-Interest recommendation by analyzing the main issues and challenges, while also proposing solutions to palliate them. Regarding the classical recommendation scenario, we have presented the current state-of-the-art of the Recommender Systems area, showing both classical algorithms and the most common metrics that are still used today, analyzing some of the challenges of these classical approaches in order to define new metrics and algorithms using contextual information, specially temporal and sequential information. These contextual recommenders are currently the algorithms that can offer a better experience to users, as they can adapt more easily to their tastes and needs. For this, we have defined first new metrics incorporating these contexts to analyze the recommendations produced by the algorithms in the following aspects: freshness, anti-relevance, sequentiality, and biases depending on user and item attributes, showing that all this information can also be used to evaluate the recommenders. Second, we have proposed a new similarity between users based on the Longest Common Subsequence (LCS) algorithm to be integrated into neighbor-based Recommender Systems that takes into account both temporal and sequential information. Besides, we have proposed a redefinition of the neighbor-based algorithms to make recommendations exploiting the last common interaction between the target user and the rest of her neighbors, producing better and more temporal novel recommendations than other state-of-the-art algorithms, showing that simple models remain applicable in the area.

Additionally, we have analyzed in detail the Point-of-Interest (POI) recommendation problem by classifying a large number of recent proposals according to the type of information, algorithms, and evaluation methodology used. With this review we have been able to confirm that the POI recommendation problem is still relevant today as there are an increasing number of new works that are proposed every year. However, we have also detected a problem regarding the reproducibility of the results obtained by the models, since most of the algorithms are not comparable with each other because they have major discrepancies in the way recommendations and their evaluations are made. Subsequently, we have also explored the recommendation of routes from independent POIs, using categorical and sequential information to generate sequences from independent recommended POIs by exploiting reranking techniques. Although our reranking approaches have not shown remarkable superiority with respect to other algorithms in terms of ranking accuracy, they have shown promising results in other dimensions, like improving the category accuracy while reducing the distance to be followed by the users based on the recommendations received. Finally, as in Point-of-Interest recommendation the data sparsity is a severe challenge (more than in classical recommendation), we have applied data aggregation strategies based on cross-domain techniques to improve the performance of POI recommenders in different regions. Using these simple strategies we have been able to improve the performance of some recommenders in several dimensions such as accuracy and user coverage.

In this chapter we present the main conclusions obtained in this thesis. In Section 8.1 we provide more details about the contributions of this research and in Section 8.2 we present some research directions that could be addressed in future work.

## 8.1 Summary of contributions

In the following subsections we discuss and summarize the main contributions of this thesis by addressing the research goals stated in Chapter 1. First, for RG1, we reviewed the state-of-the-art of POI recommendation approaches and characterize them in terms of the information, type of algorithms, and evaluation methodologies used. With respect to RG2, we developed a set of metrics incorporating additional information like time, sequences, and user and item attributes where we tested them in two well-known datasets. For RG3, we defined a similarity metric between users by exploiting the Longest Common Subsequence between their interactions; we also redefined the formulation of $k$-NN Recommender Systems by generating a ranking for the candidate user by exploiting her last interactions with other users in the system. For RG4, we showed that we can generate full trajectories from Location-Based Social Networks data by using reranking techniques. Finally, for RG5 we studied how aggregation techniques derived from the cross-domain area con help us improving the performance and the user coverage of Point-of-Interest recommenders.

### 8.1.1 Alternative metrics for Recommender Systems

In Chapter 4 we presented a set of new metrics to be used in Recommender Systems that exploit temporal, sequential, and categorical information. We also propose variations of metrics that account for the cases when items with low ratings are recommended.

We showed the importance of using this kind of information at the evaluation step in order to further analyze the results of the recommenders.

With our time-aware novelty metrics, we have shown that there is a clear relationship between the relevant items and their temporal novelty. This is in fact really useful as these metrics allow us to detect possible biases and temporal burst of interactions within the system. Besides, our time-aware novelty metrics allow us to build item profiles with repetitions, making them suitable to other recommendation domains such as music.

By using our anti-relevance models we have determined that sometimes even if the recommenders suggest relevant items to users, they also return items that users have rated negatively. Because of this, we believe that when this information is available (i.e., when using datasets with explicit ratings), those items recommended with a very low rating should be further penalized when evaluating the recommenders. Besides, we consider this an important aspect to be taken into account, since sometimes a really bad recommendation might cause a great distrust in the users of the system. In fact, even though we think that this aspect should be analyzed more in-depth, we have not found many studies analyzing the "bad" recommendations produced by the algorithms; hence, we consider it a novel dimension that this work has contributed to the field of RS evaluation.

Regarding the user and item attributes in recommendations, we have observed that users are typically classified into groups, which in turn may obtain very different results depending on the groups they belong to. This observation connects with the analysis of fairness in recommendation (since in those cases, some users will obtain better or worse recommendations only because of some inherent characteristics), which implies that some models may not be returning fair recommendations. With the item attributes we can better distinguish the results returned by the algorithms and increase the performance obtained by the algorithms on very sparse datasets. However, we believe that allowing to evaluate with attributes should be done carefully as we may end up unrealistically increasing the performance of recommenders if we do not apply appropriate penalizations.

The metrics developed in Chapter 4 have therefore allowed us to obtain a more complete view of the results obtained by the recommenders. We have tested the usefulness of our metrics on two well-known datasets: Movielens1M and Foursquare, in order to analyze differences and similarities in both domains using two different evaluation methodologies (random and time-aware splits). Finally, although this is a result already known by the community, we have, in agreement with other researchers in the field, found differences in the results obtained by the algorithms depending on the recommendation scenario or evaluation methodology used (type of split, parameter tuning, optimizing the algorithms, etc.), see (Beel et al., 2016, Dacrema et al., 2019). For this reason, we again stress the importance of being as transparent as possible when evaluating recommendation – or, in fact, any Machine Learning – algorithm.

### 8.1.2 Sequential information in $k$-NN Recommender Systems

In Chapter 5 we presented two complementary proposals for neighbor-based Recommender Systems. First, we proposed to adapt the use of the Longest Common Subsquence algorithm to be considered as a classical similarity metric such as cosine similarity or Pearson correlation. This similarity, even though it is computationally more expensive than the aforementioned ones, it has two major advantages. On the one hand, it allows us to take into account sequential components, on the other hand, it is flexible enough to operate with additional information of the items, like their attributes, in order to create a hybrid algorithm (with the additional advantage that it is easy to explain and implement).

Second, we redefined the nearest neighbor algorithms to generate a ranking for the target user based on the last interactions they have in common with their corresponding neighbors. Our main proposal here was to use ranking fusion techniques, which allowed us to generate recommendations that integrate sequential information. Besides, this new formulation is capable to operate with any similarity metric, either classical ones like Pearson or cosine, or the sequential similarity metric proposed earlier in our work. These approaches have been evaluated against other state-of-the-art algorithms showing that in some circumstances our proposal is more competitive than the rest of the algorithms, whereas in other situations they are still effective. With this new formulation of nearest neighbor algorithms, we demonstrate that these types of proposals are still applicable and adaptable in the field because they are more interpretable, efficient (since they can be easily parallelized), and simpler than other algorithms such as neural networks. We have tested our recommenders in two different datasets with realistic timestamps: a MovieTweetings subset (from the movies domain) and a subset of the Foursquare dataset (from the Point-of-Interest domain) under two time-aware evaluation methodologies, a more realistic one considering a temporal split at the system level and another temporal split per user to demonstrate that our proposal can obtain competitive results in both methodologies.

### 8.1.3 New perspectives on Point-of-Interest Recommender Systems

In Chapter 3 we conducted a survey on the Point-of-Interest problem, showing that this area is still relevant for the researchers. Although we have been able to identify that algorithms that exploit geographical and temporal information are widely used, we have also detected that there is not a common evaluation protocol for analyzing the performance of the recommenders. In this regard, we have observed that most models work with very different datasets and they use diverse evaluation methodologies (different types of splits, differences in data filtering, etc.), and hence, making the comparison between them, sometimes unfeasible.

In Chapter 6 we have explored the problem of route or trajectory recommendation on the basis of the recommendation of independent POI. First, we define a framework to generate routes from LBSNs data and then we propose the use of reranking techniques to generate routes from the recommendations produced by classical recommendation algorithms. Specifically, we used three different reranking techniques: independent, where the score of the reranked items only depends on the user-item pair, dependent on the last item, where the score of each reranked item depends on the previous item, and finally the rerankers that depend on the whole sequence, where the items are reranked to optimize the full recommended route according to different conditions (e.g., distance, item probability, or category probability). Although our reranking techniques do not obtain substantial improvements in terms of relevance, they do allow improvements in terms of POI category hits while achieving similar levels of accuracy starting from very simple methods. This in particular shows that sometimes routes can be generated in a simple way by taking advantage of recommendations previously produced by other algorithms. To evaluate our proposals, we have used four different real world datasets. Our methods worked well in all cases, specially in the one that contained more touristic information.

Finally, in Chapter 7 we developed a set of multi-city aggregation techniques in order to improve the performance of both classical and POI recommendation algorithms. Specifically, we have made recommendations to a subset of independent cities with more check-ins from the Foursquare dataset using three different strategies: selecting the geographical closest cities to each of them, selecting the rest of cities in each country for each target city, and using the information of the most popular cities to perform recommendations in each city independently. Through our experiments, we have been able to prove that the accuracy and user-level coverage of most of the models can be improved more by using the strategies based on proximity or by country than by using the cities with the most check-ins, demonstrating that the algorithms depend more on the quality than on the quantity of the data. Besides, we have been able to verify that while the geographical component is useful if we perform recommendations for each city independently, when aggregation strategies are used, the performance is not always improved.

## 8.2 Future work

Throughout this thesis we have shown only a small fraction of the Recommender Systems area. In fact, although we have proposed solutions and advanced in the state-of-the-art in both classical and Point-of-Interest recommendation, we believe that some of these contributions can be further extended in the future. For this reason, in this section we summarize the main lines of research that can be further developed.

### 8.2.1 On Recommender Systems evaluation

First, we believe that the developed metrics shown in Chapter 4 have enough potential to be applied in other areas of recommendation. For example, we believe that the time-aware novelty metrics have a special interest on streaming RS, where the time-aware model of the recommender might not be the same as the one for the metric: e.g., while the metric could be recomputed every day as the temporal novelty is crucial in this domain, the recommender could be trained once a week. In this way, a more fine-grained analysis may be derived so, for instance, the optimal period to train the recommender can be computed, or a day-by-day sensitivity could be explored for different recommenders. It is worth mentioning that, although this is also possible to achieve with offline data, the conclusions will not be as significant because most of the datasets are very sparse, hence, it is necessary to use real, online data.

In the case of the anti-relevance framework, we aim to extend the analysis of those metrics to more families of algorithms and also in specially difficult recommendation tasks such as cold-start or cross-domain, to understand the behavior of the recommendation techniques in those scenarios. More importantly, we would like to analyze how to extend our framework to situations where no explicit ratings are available, but other form of anti-relevance can be inferred, either directly or through the user interaction with the system.

Regarding the features of users and items, by exploiting user attributes we can extend the analysis performed in this thesis to other experiments, such as discriminating between active or influential users, or between bots or any other type of attacker or different groups of users beyond tourists and locals in the tourism domain. On the other hand, the analysis of item attributes can be interesting in order to detect biases in the recommendations (e.g., if the recommendations are biased towards specific categories of the items or if users belonging to a particular group tend to consume items with a distinct category) as well as to apply relevance metrics to match the categories of recommended items with test items, in domains where there are a large number of items, as in POI or music recommendation.

### 8.2.2 On sequential-based $k$-NN Recommender Systems

The results obtained by our novel similarity metric based on the Longest Common Subsequence and the results of our reformulation of neighborhood based algorithms show that these simple models are still competitive and can be adapted to include temporal and/or sequential information. As future work, we plan to explore the use of alternative aggregation functions – such as those based on the score distribution (Manmatha et al., 2001) – when integrated in our proposal. Furthermore, an exhaustive analysis – with more datasets, baselines such as SVD++ with temporal information (Koren and Bell, 2015) or other Neural Networks techniques (Hidasi et al., 2016, Donkers et al., 2017, He et al., 2017b), and other evaluation methodologies – should be made to better understand each component of the proposed models. For instance, the number of items allowed to be selected before and after the last common interaction, together with alternative definitions for sequence-aware similarity metrics. As an example, we aim to extend the proposed similarity based on LCS by exploiting other dimensions to build the sequence upon (such as item features, ratings, or combinations thereof) or even by applying filters to select those items that have been rated with a higher value than a specific threshold to create the user sequences, as we recently analyzed in Sánchez and Bellogín (2019). However, for POI this might be counterproductive, as we may end up adding more sparsity by filtering too much data.

Furthermore, it would be interesting to observe the impact on users' online behavior once they receive the recommendations, as it was recently analyzed for social networks in Falavarjani et al. (2019). Besides, it would also be interesting to perform a complete analysis on the impact of optimizing the parameters using (or not) a validation subset. In fact, we have observed how difficult it was to find consistent results (from the validation subset to the full dataset) in Chapter 4. Because of this, we aim to analyze these issues in more detail in the future to obtain guidelines or theoretical guarantees that the parameters learned using a temporal validation split would fit well using the complete data, since we believe this is the most realistic way to tune the parameters, as if it was done in a real-world system.

### 8.2.3 On the POI recommendation problem

Regarding the POI recommendation problem, thanks to the survey performed in Chapter 3, we have been able to identify a large number of approaches that are not comparable among them, mostly because the evaluation methodology conducted in most of those works varies substantially (e.g., different types of splits, datasets, parameter tuning, etc.). For this reason, as a future work, we believe that it is necessary to conduct more experimental surveys (as in Liu et al. (2017)) comparing the most recent and most cited approaches of POI recommendation while performing different types of splits in at least the following conditions: recommendation in the same city, recommendation in different cities around the world, and recommendation in specific regions. This will help us to identify if the algorithms can exploit geographical influence in each of these situations correctly. In addition, such an analysis can be used to understand in more detail if there are more algorithms that can benefit from the cross-domain strategies mentioned in Chapter 7.

With respect to these strategies, we consider that it is important to use other aggregation strategies that, instead of maximizing the number of matching users, might be based on establishing similarities between items. In this aspect, we consider that it could be interesting to see how algorithms such as SLIM, FISM, or item embeddings behave under theses circumstances.

Finally, we would like to emphasize once again that any conducted research work should also publish the framework used to perform the experiments, since it is generally acknowledged that the same model in different frameworks might produce completely different results (Said and Bellogín, 2014). In the same way, all the baselines used should be tuned properly in order to find a competitive performance against the proposed algorithms.

# Part IV

# Appendices

# Appendix A

# Additional results on POI recommendation

## A.1 Complete Tables for the POI recommendation survey

In this section we extend the systematic review of the state-of-the-art algorithms of POI reported in Section 3.6. Hence, in Tables A.1, A.2, A.3 and A.4 we show all the algorithms (not only the most representative ones, as in Chapter 3) analyzed between the years 2011 and 2019 following the same configuration as in Table 3.2.

## A.2 Sequences in POI recommendation

### A.2.1 Complete performance results of venue recommender systems

Table A.5 shows the results in the four cities described in Section 6.3.1 for the best recommender for each family. We observe that the distance obtained in the TestOrder recommender is the lowest in every city except Rome. Moreover, the Pop recommender is the best one for the Basic family in all cities; in fact, it is a very hard baseline to beat since the best recommenders from other families often obtain very close or lower values in terms of relevance. Finally, we also observe that the best recommenders in terms of relevance are among the Temporal, Geo, and Tour families. These results, hence, confirm that there is a strong popularity bias in all cities and that including different contextual factors like temporal or geographical information is critical to improve the effectiveness of the models.

Regarding specific recommenders, the behavior of the ItemMC and the BPRMF recommenders is interesting, as they are the best in their families, obtaining positive results in all the metrics, but especially in the sequential ones. The reason for the good performance of ItemMC might be attributed to the fact that it exploits collaborative – but sequential – information, which suffers from a popularity bias, which, as analyzed before, tends to produce good results just for statistical reasons. The BPRMF approach, on the other hand, is tailored to optimize the ranking by modeling the implicit

**Table A.1:** Summary of analyzed POI recommendation approaches sorted by publication year.

| Details | | | Information used | | | | | Model | | | | | | | Split type | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Year | Reference | Acronym | Geographical | Social | Content | Sequential | Temporal | CF sims | Factorization | Probabilistic | DNN | Graph/Link | Hybrid | Other | Random | Temporal | Other |
| 2011 | Symeonidis et al. (2011) | (N.A.) | | | | | | | ✓ | | | | | | ✓ | | |
| 2011 | Ye et al. (2011) | USG | ✓ | ✓ | | | | ✓ | | ✓ | | | ✓ | | ✓ | | |
| 2012 | Levandoski et al. (2012) | LARS | ✓ | | | | | ✓ | | | | | | | ✓ | | |
| 2012 | Bao et al. (2012) | (N.A.) | ✓ | | ✓ | | | ✓ | | | | ✓ | | | | | ✓ |
| 2012 | Ying et al. (2012) | UPOI-Mine | ✓ | ✓ | ✓ | | | ✓ | | | | | ✓ | | | | |
| 2012 | Noulas et al. (2012) | RW, Weighted-RW | | ✓ | | | | | | ✓ | | ✓ | | | | ✓ | |
| 2013 | Zheng et al. (2013) | CRTCF | | ✓ | ✓ | | | ✓ | ✓ | ✓ | | | | | | | ✓ |
| 2013 | Zhao et al. (2013) | GMM (1), GA-GMM (2) | ✓ | | | | | | | ✓ | | | | **2** | | ✓ | |
| 2013 | Rahimi and Wang (2013) | PCLR | ✓ | | ✓ | | ✓ | | | ✓ | | | ✓ | | ✓ | | |
| 2013 | Yang et al. (2013) | LBSMF | | ✓ | ✓ | | | ✓ | ✓ | ✓ | | | | | ✓ | | |
| 2013 | Long and Joshi (2013) | (N.A.) | | ✓ | | | | | | | | ✓ | | | ✓ | | |
| 2013 | Liu and Xiong (2013) | TL-PMF | | | ✓ | | | | ✓ | ✓ | | | | | ✓ | | |
| 2013 | Cheng and Chang (2013) | CLW | | ✓ | | | ✓ | ✓ | | | | | ✓ | | ✓ | | |
| 2013 | Liu et al. (2013b) | (N.A.) | ✓ | | ✓ | ✓ | ✓ | | ✓ | ✓ | | | ✓ | | | ✓ | |
| 2013 | Ference et al. (2013) | UPS-CF | | ✓ | | | | ✓ | | | | | | | ✓ | | |
| 2013 | Liu et al. (2013a) | GT-BNMF | ✓ | | ✓ | | | | ✓ | ✓ | | | | | ✓ | | |
| 2013 | Cheng et al. (2013) | FPMC-LR | ✓ | | | ✓ | | | ✓ | ✓ | | | | | | ✓ | |
| 2013 | Gao et al. (2013) | LRT | | | | | ✓ | | ✓ | | | | | | ✓ | | |
| 2013 | Wang et al. (2013) | LFBCA | ✓ | ✓ | | | | | | | | ✓ | | | | ✓ | |
| 2013 | Zhang and Chow (2013) | iGSLR | ✓ | ✓ | | | | ✓ | | ✓ | | | ✓ | | | ✓ | |
| 2013 | Yuan et al. (2013) | UTE+SE | ✓ | | | | ✓ | ✓ | | ✓ | | | ✓ | | ✓ | | |
| 2014 | Lu et al. (2014) | Ricochet | | | ✓ | | ✓ | | | | | | ✓ | | | | ✓ |
| 2014 | Ozsoy et al. (2014) | MO | ✓ | ✓ | | | | ✓ | | | | | | | | ✓ | |
| 2014 | Kosmides et al. (2014) | PNN | ✓ | ✓ | | | ✓ | | | ✓ | ✓ | | | | ✓ | | |
| 2014 | Zhao et al. (2014) | BPTFSLR | | ✓ | | | ✓ | | ✓ | ✓ | | | | | | ✓ | |
| 2014 | Hu and Ester (2014) | ST | | ✓ | ✓ | | | | ✓ | ✓ | | | | | ✓ | | |
| 2014 | Nunes and Marinho (2014) | DGM | ✓ | | | | | ✓ | | | | ✓ | | | ✓ | | |
| 2014 | Wang et al. (2014) | GPUR, GPLR | | ✓ | ✓ | | | ✓ | | | | | | | ✓ | | |
| 2014 | Zou et al. (2014) | ITF | ✓ | | | | | | ✓ | | | | | | ✓ | | |
| 2014 | Ying et al. (2014) | UPOI-Walk | ✓ | ✓ | ✓ | | | | | | | ✓ | | | | | |
| 2014 | Yuan et al. (2014) | GTAG | ✓ | | | | ✓ | | | | | ✓ | | | ✓ | | |
| 2014 | Lian et al. (2014) | GeoMF | ✓ | | | | | | ✓ | | | | | | ✓ | | |
| 2014 | Liu et al. (2014) | IRenMF | ✓ | | | | | | ✓ | | | | | | ✓ | | |
| 2014 | Zhang et al. (2014) | LORE | ✓ | ✓ | | ✓ | | ✓ | | ✓ | | | ✓ | | | ✓ | |
| 2014 | Zhou and Wang (2014) | sPCLR | ✓ | | ✓ | | ✓ | ✓ | | | | | ✓ | | ✓ | | |
| 2015 | Fukuda and Aritsugi (2015) | (N.A.) | | | | | ✓ | | | | | | | ✓ | | ✓ | |
| 2015 | Sattari et al. (2015) | EFC | | | | | | ✓ | ✓ | | | | ✓ | | ✓ | | |
| 2015 | Gao et al. (2015b) | gSCorr | ✓ | ✓ | | | ✓ | ✓ | ✓ | ✓ | | | ✓ | | | ✓ | |
| 2015 | Kojima and Takagi (2015) | URG+SM | ✓ | | | | ✓ | ✓ | | ✓ | | | ✓ | | | | |
| 2015 | Guo et al. (2015a) | TSLR (1), TSLRS (2) | **2** | | ✓ | | | | ✓ | ✓ | | ✓ | | | ✓ | | |
| 2015 | Bagci and Karagoz (2015) | RWCAR | ✓ | ✓ | ✓ | | | | | | | ✓ | | | | ✓ | |
| 2015 | Huang et al. (2015) | FGLR | ✓ | ✓ | | | | | ✓ | ✓ | | | | | | ✓ | |
| 2015 | Zahálka et al. (2015) | City Melange | | | ✓ | | | | ✓ | ✓ | ✓ | | ✓ | ✓ | | | ✓ |
| 2015 | Mudda and Giordano (2015) | REGULA | ✓ | ✓ | | | ✓ | | | | | | ✓ | | | ✓ | |
| 2015 | Zhang and Wang (2015) | LTSCR | ✓ | ✓ | | | ✓ | | ✓ | | | | | | ✓ | | |
| 2015 | Griesner et al. (2015) | GeoMF-TD | ✓ | | | | ✓ | | ✓ | | | | | | ✓ | | |
| 2015 | Lin et al. (2015) | USPB | ✓ | ✓ | | | | ✓ | | ✓ | | | ✓ | | ✓ | | |
| 2015 | Zhang et al. (2015a) | iGeoRec | ✓ | ✓ | | | | | | ✓ | | | | | | ✓ | |
| 2015 | Yao et al. (2015) | TenInt | | ✓ | | | ✓ | | ✓ | | | | | | ✓ | | |
| 2015 | Li et al. (2015b) | TCL-K | | | | | ✓ | | ✓ | ✓ | | | | | | ✓ | |
| 2015 | Yin et al. (2015) | LA-LDA | ✓ | | ✓ | | | | ✓ | ✓ | | | | | ✓ | | |
| 2015 | Lu et al. (2015b) | LURA | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | | | ✓ | | | ✓ | |
| 2015 | Wang et al. (2015) | UGC | | | ✓ | | | ✓ | ✓ | | | | | | ✓ | | |
| 2015 | Qi et al. (2015) | HRWR | | ✓ | ✓ | | | | | | | ✓ | | | | | |
| 2015 | Gupta et al. (2015) | (N.A.) | ✓ | ✓ | ✓ | | | | | | | ✓ | | | ✓ | | |
| 2015 | Li et al. (2015c) | MARS | | | ✓ | | | | ✓ | | | | | | | | ✓ |
| 2015 | Zhao et al. (2015) | TA-FPMC | ✓ | | ✓ | ✓ | ✓ | | ✓ | ✓ | | | | | ✓ | | |
| 2015 | Zhang and Chow (2015a) | CoRe | ✓ | ✓ | | | | ✓ | | ✓ | | | ✓ | | | ✓ | |
| 2015 | Abdel-Fatao et al. (2015) | STS | ✓ | | ✓ | | ✓ | ✓ | | | | | ✓ | | ✓ | | |
| 2015 | Zhang et al. (2015b) | ORec | ✓ | ✓ | ✓ | | | ✓ | | ✓ | | | ✓ | | ✓ | | |
| 2015 | Liu et al. (2015) | Poisson Geo-PFM | ✓ | | | | | | ✓ | ✓ | | | | | ✓ | | |
| 2015 | Li et al. (2015a) | RankGeoFM | ✓ | | | | ✓ | | ✓ | | | | | | | ✓ | |
| 2015 | Zhang and Chow (2015b) | GeoSoCa | ✓ | ✓ | ✓ | | | | | ✓ | | | ✓ | | | ✓ | |
| 2015 | Feng et al. (2015) | PRME-G | ✓ | | | ✓ | ✓ | | ✓ | ✓ | | | | | | ✓ | |
| 2015 | Gao et al. (2015a) | CAPRF | | | ✓ | | | | ✓ | | | | | | ✓ | | |

**Table A.2:** Summary of analyzed POI recommendation approaches (same configuration as Table A.1).

| Details | | | Information used | | | | | Model | | | | | | | Split type | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Year | Reference | Acronym | Geographical | Social | Content | Sequential | Temporal | CF sims | Factorization | Probabilistic | DNN | Graph/Link | Hybrid | Other | Random | Temporal | Other |
| 2015 | Li et al. (2015d) | SFPMF,UIPMF | ✓ | ✓ | | | | | ✓ | ✓ | | | ✓ | | | ✓ | |
| 2015 | Lian et al. (2015) | ICCF | | | ✓ | | | | ✓ | | | | | | ✓ | | |
| 2016 | Li et al. (2016b) | (N.A.) | ✓ | | ✓ | ✓ | ✓ | | | | | | | ✓ | | ✓ | |
| 2016 | Vakeel and Ray (2016) | (N.A.) | | | ✓ | | | | | | | | | ✓ | | | ✓ |
| 2016 | Nie et al. (2016) | (N.A.) | | | ✓ | | | | | | | ✓ | | | | | ✓ |
| 2016 | Trattner et al. (2016) | (N.A.) | ✓ | | ✓ | | ✓ | | ✓ | | | | | | | ✓ | |
| 2016 | Capdevila et al. (2016) | GeoSRS | | | ✓ | | | ✓ | ✓ | ✓ | | | ✓ | | | ✓ | |
| 2016 | Li et al. (2016c) | CPMFPPC | | | | | ✓ | ✓ | ✓ | ✓ | | | ✓ | | ✓ | | |
| 2016 | Habib et al. (2016) | (N.A.) | ✓ | | ✓ | | ✓ | | | | | | ✓ | | | | |
| 2016 | Yao et al. (2016) | TM-PFM | | | ✓ | | ✓ | | ✓ | ✓ | | | | | ✓ | | |
| 2016 | Lian et al. (2016) | RCTF | ✓ | | | | ✓ | | ✓ | ✓ | | | | | | ✓ | |
| 2016 | Albanna et al. (2016) | IALBR | ✓ | | ✓ | | | | | | | ✓ | ✓ | | | | ✓ |
| 2016 | Stepan et al. (2016) | (N.A.) | ✓ | ✓ | | | ✓ | ✓ | | | | | ✓ | | | ✓ | |
| 2016 | Ozsoy et al. (2016) | (N.A.) | | ✓ | | | ✓ | ✓ | | | | | | | | ✓ | |
| 2016 | Xie et al. (2016a) | GE | ✓ | | ✓ | ✓ | ✓ | | ✓ | ✓ | | | ✓ | | | ✓ | |
| 2016 | Manotumruksa et al. (2016) | DeepReg | | ✓ | ✓ | | | | ✓ | | | | | | ✓ | | |
| 2016 | Zhu and Hao (2016) | (N.A.) | | | ✓ | | ✓ | | ✓ | | | | | | | ✓ | |
| 2016 | Chen et al. (2016c) | (N.A.) | ✓ | | ✓ | ✓ | | | ✓ | ✓ | | ✓ | | | | ✓ | |
| 2016 | Yuan and Li (2016) | STS_Grid, STS_DBSCAN | ✓ | | | | ✓ | ✓ | | | | | | | | ✓ | |
| 2016 | Pipanmaekaporn and Kamonsantiroj (2016) | (N.A.) | ✓ | | ✓ | | | ✓ | | | | | | | | | ✓ |
| 2016 | Baral and Li (2016) | MAPS | ✓ | ✓ | ✓ | | ✓ | | | | | ✓ | | | ✓ | | |
| 2016 | Cheng et al. (2016) | BPRLR1, BPRLR2 | ✓ | | | | | | ✓ | ✓ | | | ✓ | | ✓ | | |
| 2016 | Jueajan et al. (2016) | (N.A.) | | | ✓ | | | ✓ | | | | | | | | | ✓ |
| 2016 | Eravci et al. (2016) | PNS (1), CNF (2) | ✓ | | ✓ | | | **2** | | **1** | | | | | | | ✓ |
| 2016 | Zhang and Chow (2016) | TICRec | ✓ | ✓ | | | ✓ | ✓ | | ✓ | | | | | | ✓ | |
| 2016 | Hosseini and Li (2016) | USGT | ✓ | ✓ | | | ✓ | ✓ | | ✓ | | | ✓ | | ✓ | | |
| 2016 | Zhang et al. (2016a) | CTS | ✓ | | | | ✓ | ✓ | | | | | ✓ | | | ✓ | |
| 2016 | Guo et al. (2016) | CoSoLoRec | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ | | | ✓ | | | ✓ | |
| 2016 | Debnath et al. (2016) | PLTSRS | | | ✓ | ✓ | ✓ | ✓ | | ✓ | | | ✓ | | | ✓ | |
| 2016 | Li et al. (2016d) | STPMF | ✓ | | ✓ | | ✓ | | ✓ | ✓ | | | | | | ✓ | |
| 2016 | Xu et al. (2016a) | Topical-GeoMF | ✓ | | ✓ | | | | ✓ | | | | | | ✓ | | |
| 2016 | Maroulis et al. (2016) | CoTF | | | ✓ | ✓ | | | ✓ | | | | | | ✓ | | |
| 2016 | Chen et al. (2016b) | MUG | ✓ | | | | | | ✓ | ✓ | | | ✓ | | ✓ | | |
| 2016 | Zhao et al. (2016b) | ATTF | | | | | ✓ | | ✓ | | ✓ | | | | ✓ | | |
| 2016 | Zheng et al. (2016) | TGTM-1, TGTM-2 | ✓ | | ✓ | | ✓ | | ✓ | ✓ | | | | | ✓ | | |
| 2016 | Xu et al. (2016b) | SSR | ✓ | ✓ | ✓ | | | | | | | ✓ | | | ✓ | | |
| 2016 | Rojas et al. (2016) | MultiGran | | | | | ✓ | ✓ | | | | | | | ✓ | ✓ | |
| 2016 | Fang and Dai (2016) | SSR | | ✓ | | | | | | | | | ✓ | | | | |
| 2016 | Zhang et al. (2016b) | SGMF | ✓ | ✓ | | | | ✓ | | | | | ✓ | | ✓ | | |
| 2016 | Baral et al. (2016) | GeoTeCS | ✓ | ✓ | ✓ | | ✓ | | ✓ | ✓ | | | | | ✓ | | |
| 2016 | Wang et al. (2016) | RC, DCC | ✓ | | ✓ | | | ✓ | | | | | | ✓ | ✓ | | |
| 2016 | Xie et al. (2016b) | GME (1), GME-S (2) | | | | **2** | ✓ | | ✓ | ✓ | | | | | | ✓ | |
| 2016 | Katarya et al. (2016) | (N.A.) | | | | | | | | | | ✓ | | | ✓ | | |
| 2016 | Guan et al. (2016) | (N.A.) | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ | | | ✓ | | | | |
| 2016 | Zhu et al. (2016) | (N.A.) | | | ✓ | | | | ✓ | ✓ | | | ✓ | | | | ✓ |
| 2016 | Li et al. (2016a) | ASMF | ✓ | ✓ | ✓ | | | | ✓ | | | ✓ | | | | ✓ | |
| 2016 | Zhao et al. (2016a) | STELLAR | | | | | ✓ | | ✓ | | | | | | | ✓ | |
| 2016 | He et al. (2016) | (N.A.) | ✓ | | | ✓ | | | ✓ | ✓ | | | | | | | |
| 2016 | Liu et al. (2016a) | WWO | | | | ✓ | ✓ | | | ✓ | | | | | | ✓ | |
| 2016 | Lv et al. (2016) | ELR-DC | | ✓ | | | | ✓ | | | | | | | ✓ | | |
| 2016 | Yuan et al. (2016) | GeoBPR | ✓ | | | | | | ✓ | ✓ | | | | | ✓ | | |
| 2017 | Rios et al. (2017) | (N.A.) | ✓ | ✓ | ✓ | | | ✓ | | | | | | | ✓ | | |
| 2017 | Kefalas and Manolopoulos (2017) | USTT$_c$ | ✓ | | ✓ | | | ✓ | | | | | ✓ | | ✓ | ✓ | |
| 2017 | Zhao et al. (2017a) | Geo-Teaser | ✓ | | | ✓ | ✓ | | ✓ | ✓ | ✓ | | | | | ✓ | |
| 2017 | Zhao et al. (2017b) | HGMF | ✓ | | | | | | ✓ | | | | | | ✓ | | |
| 2017 | Manotumruksa et al. (2017b) | PRFMC | ✓ | ✓ | | | | | ✓ | ✓ | | | ✓ | | ✓ | | |
| 2017 | Manotumruksa et al. (2017a) | GRMF, MLRP | ✓ | | | ✓ | | | ✓ | | ✓ | | ✓ | | | ✓ | |
| 2017 | Ravi and Subramaniyaswamy (2017a) | SPTW | | | ✓ | | | | ✓ | | | ✓ | | | | | |
| 2017 | tao Zheng et al. (2017) | TPR-UM | ✓ | | | | ✓ | | ✓ | ✓ | | | | | ✓ | | |
| 2017 | Gau et al. (2017) | UGSE-LR | ✓ | | | ✓ | | ✓ | | | | ✓ | ✓ | | ✓ | | |
| 2017 | Xia et al. (2017b) | VRer | | | ✓ | | ✓ | | | | | | | ✓ | | | |
| 2017 | Yang et al. (2017b) | TDLDA, TATD | | | ✓ | | ✓ | | ✓ | ✓ | | | | | ✓ | | |
| 2017 | Oppokhonov et al. (2017) | CLB | | | | ✓ | ✓ | ✓ | | | | | | | ✓ | | |
| 2017 | Chen et al. (2017) | (N.A.) | | | ✓ | | ✓ | ✓ | | | | | ✓ | ✓ | ✓ | | |
| 2017 | Yang et al. (2017a) | PACE | ✓ | ✓ | | | | | | | ✓ | | | | | ✓ | |

**Table A.3:** Summary of analyzed POI recommendation approaches (same configuration as Table A.1).

| Details | | | Information used | | | | | Model | | | | | | | Split type | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Year | Reference | Acronym | Geographical | Social | Content | Sequential | Temporal | CF sims | Factorization | Probabilistic | DNN | Graph/Link | Hybrid | Other | Random | Temporal | Other |
| 2017 | Wang et al. (2017b) | LSARS | ✓ | | ✓ | | | | ✓ | ✓ | | | | | ✓ | | |
| 2017 | Si et al. (2017) | CTF-ARA | | | | ✓ | ✓ | ✓ | | | | | | | ✓ | | |
| 2017 | Guo et al. (2017) | AGSG | ✓ | ✓ | ✓ | | | | | | | ✓ | | | ✓ | | |
| 2017 | Wagih et al. (2017) | (N.A.) | | ✓ | | | | | | | | ✓ | | | | | |
| 2017 | Luan et al. (2017) | PCTF | | | ✓ | | ✓ | | ✓ | | | | | | ✓ | | |
| 2017 | Ying et al. (2017) | TAP | | ✓ | ✓ | | ✓ | | ✓ | | | ✓ | | | ✓ | | |
| 2017 | Ren et al. (2017) | TGSC-PMF | ✓ | ✓ | ✓ | | | | ✓ | ✓ | | | | | ✓ | | |
| 2017 | Zhu et al. (2017) | SEM-PPA | ✓ | ✓ | ✓ | ✓ | | | | | | ✓ | | | | ✓ | |
| 2017 | Yu et al. (2017) | (N.A.) | | | | | | | | ✓ | | ✓ | | | ✓ | | |
| 2017 | Ravi and Subramaniyaswamy (2017b) | EIUCF, EIICF | | | ✓ | | | ✓ | | | | | ✓ | | ✓ | | |
| 2017 | Han and Yamana (2017) | (N.A.) | ✓ | | ✓ | | | | ✓ | ✓ | | | | | ✓ | | |
| 2017 | He et al. (2017a) | LBPR | ✓ | | ✓ | ✓ | | | ✓ | ✓ | | | ✓ | | | ✓ | |
| 2017 | Li et al. (2017a) | IEMF | ✓ | | | | | | ✓ | | | | | | | ✓ | |
| 2017 | Zhao et al. (2017c) | Geo-PRMF | ✓ | | | | | | ✓ | | | | | | ✓ | | |
| 2017 | Wang et al. (2017a) | VPOI | | | ✓ | | | | ✓ | ✓ | ✓ | | | | ✓ | | |
| 2017 | Rahimi et al. (2017) | LBA, BF | ✓ | | ✓ | | ✓ | | ✓ | ✓ | | | **2** | | ✓ | | |
| 2017 | Li et al. (2017c) | CBGeoMFC | ✓ | | ✓ | | | ✓ | ✓ | ✓ | | | | | ✓ | | |
| 2017 | Hosseini et al. (2017) | NH-JTI | | | | | ✓ | ✓ | | ✓ | | | | | ✓ | | |
| 2017 | Xu et al. (2017a) | SCCF | | ✓ | ✓ | | | | ✓ | | | | ✓ | | | | |
| 2017 | Xia et al. (2017a) | ARNN | ✓ | | ✓ | ✓ | | | | | ✓ | | | | | ✓ | |
| 2017 | Zeng et al. (2017) | TSG | ✓ | ✓ | | | | ✓ | | | | ✓ | | | ✓ | | |
| 2017 | Xu et al. (2017b) | SSLR | ✓ | ✓ | | ✓ | | | | ✓ | | ✓ | ✓ | | | ✓ | |
| 2017 | Shi and Jiang (2017) | LST | ✓ | | | ✓ | | ✓ | | | | | ✓ | | ✓ | | |
| 2017 | Xing et al. (2017) | (N.A.) | ✓ | | | ✓ | ✓ | | | | | ✓ | | | ✓ | | |
| 2017 | Erande and Chaugule (2017) | (N.A.) | | ✓ | ✓ | | | ✓ | | | | | ✓ | | | | |
| 2018 | Chen et al. (2018b) | (N.A.) | | | ✓ | | | ✓ | | | | | ✓ | | ✓ | | |
| 2018 | Liu and Wang (2018) | (N.A.) | ✓ | | | ✓ | ✓ | | | ✓ | | | ✓ | | ✓ | | |
| 2018 | Liu et al. (2018) | PCRM | | | ✓ | | | | ✓ | ✓ | | | | | ✓ | | |
| 2018 | Gao et al. (2018b) | STSCR | | ✓ | | ✓ | ✓ | | ✓ | ✓ | | | | | ✓ | | |
| 2018 | Qian et al. (2018) | TransTL | ✓ | | | | ✓ | | ✓ | | | | | | | ✓ | |
| 2018 | Griesner et al. (2018) | ALGeoSPF | ✓ | | | ✓ | | | ✓ | ✓ | | | | | | | |
| 2018 | Ma et al. (2018) | SAE-NAD | ✓ | | | | | | | | ✓ | | | | ✓ | | |
| 2018 | Zhu et al. (2018b) | LTSR | ✓ | ✓ | | | ✓ | ✓ | ✓ | ✓ | | | ✓ | | | ✓ | |
| 2018 | Zhao et al. (2018c) | GR-DELM | ✓ | ✓ | | | | ✓ | ✓ | | | ✓ | | | ✓ | | |
| 2018 | Xing et al. (2018) | ReGS | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ | ✓ | | | | | ✓ | |
| 2018 | Amirat et al. (2018) | LocRec | | ✓ | | ✓ | ✓ | | | | | | | ✓ | ✓ | | |
| 2018 | Wang et al. (2018c) | ULE | ✓ | ✓ | | | | ✓ | ✓ | ✓ | | | ✓ | | ✓ | | |
| 2018 | Liao et al. (2018) | (N.A.) | | | ✓ | | ✓ | | ✓ | ✓ | | | | | | | |
| 2018 | Baral et al. (2018) | ReEl | ✓ | | ✓ | | | | ✓ | | ✓ | | | | ✓ | | |
| 2018 | Gao et al. (2018) | GSBPR | ✓ | ✓ | | | | | ✓ | ✓ | | | | | ✓ | | |
| 2018 | Xu et al. (2018b) | BTC | | | ✓ | | ✓ | | | | | | | ✓ | | | ✓ |
| 2018 | Manotumruksa et al. (2018) | CARA | ✓ | | | ✓ | ✓ | | | | ✓ | | | | | ✓ | |
| 2018 | Xu et al. (2018a) | GeoUMF | ✓ | | | | | | ✓ | | | | | | ✓ | | |
| 2018 | Liu et al. (2018) | (N.A.) | | | | | ✓ | | ✓ | | | | | | | ✓ | |
| 2018 | Wang et al. (2018) | (N.A.) | ✓ | | ✓ | | ✓ | | ✓ | | | | | ✓ | | | ✓ |
| 2018 | Guo et al. (2018) | NBPR | ✓ | | | | | ✓ | | ✓ | | | | | ✓ | | |
| 2018 | Su et al. (2018) | CUSPG | ✓ | ✓ | | | | ✓ | | ✓ | | | ✓ | | | ✓ | |
| 2018 | Baral and Li (2018) | FCDST (1), MF(2) | ✓ | ✓ | ✓ | | ✓ | | **2** | | | | **1** | | ✓ | | |
| 2018 | Yao et al. (2018) | TenMF | ✓ | ✓ | | | ✓ | | ✓ | | | | | | ✓ | | |
| 2018 | Gao et al. (2018a) | GeoEISo | ✓ | ✓ | | | | | ✓ | ✓ | | | | | ✓ | | |
| 2018 | Zhu et al. (2018a) | TSG | ✓ | ✓ | | | | ✓ | | ✓ | | ✓ | ✓ | | ✓ | | |
| 2018 | Gao and Yang (2018) | ABPR | ✓ | | ✓ | | | | | ✓ | | | | | | | |
| 2018 | Naserianhanzaei et al. (2018) | APPR | ✓ | | ✓ | ✓ | ✓ | | | ✓ | | | | | | ✓ | |
| 2018 | Wang et al. (2018a) | DeepRec | ✓ | ✓ | | | ✓ | | | | ✓ | | | | ✓ | | |
| 2018 | Wang et al. (2018b) | GeoIE | ✓ | | | | | | ✓ | ✓ | | | | | | ✓ | |
| 2018 | Zhu et al. (2018c) | ImSoRec | ✓ | ✓ | | | | | ✓ | ✓ | | | ✓ | | ✓ | | |
| 2018 | Zhang and Cheng (2018) | CGA | ✓ | ✓ | | | | | | | ✓ | ✓ | | | | ✓ | |
| 2018 | Zhong and Ma (2018) | (N.A.) | ✓ | | | | ✓ | | ✓ | ✓ | | | | | ✓ | | |
| 2018 | Pourali et al. (2018) | (N.A.) | ✓ | ✓ | ✓ | | | | | | | ✓ | | | ✓ | | |
| 2018 | Chen et al. (2018a) | (N.A.) | ✓ | | ✓ | | ✓ | ✓ | | ✓ | | | ✓ | | ✓ | | |
| 2018 | Ding et al. (2018b) | ST-DME | ✓ | | | ✓ | ✓ | | | | ✓ | | ✓ | | | ✓ | |
| 2018 | Zhang and Liu (2018) | TSG-list MF | ✓ | ✓ | ✓ | | | | ✓ | | | | ✓ | | ✓ | | |

**Table A.4:** Summary of analyzed POI recommendation approaches (same configuration as Table A.1).

| Details | | | Information used | | | | | Model | | | | | | | Split type | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Year | Reference | Acronym | Geographical | Social | Content | Sequential | Temporal | CF sims | Factorization | Probabilistic | DNN | Graph/Link | Hybrid | Other | Random | Temporal | Other |
| 2018 | Xia et al. (2018) | LBIMC | | | | | | | ✓ | | | | | | | ✓ | |
| 2018 | Yang et al. (2018) | MFRA | | ✓ | ✓ | | | ✓ | | | | | ✓ | | ✓ | | |
| 2018 | Li et al. (2018b) | TMCA | ✓ | | ✓ | ✓ | ✓ | | | | ✓ | | | | | ✓ | |
| 2018 | Christoforidis et al. (2018) | JLGE | ✓ | ✓ | | | ✓ | | ✓ | | | | | | ✓ | | |
| 2018 | Tal and Liu (2018) | TCENR | | | ✓ | | | | | | ✓ | | | | ✓ | | |
| 2019 | Manotumruksa et al. (2019b) | DRTL | | | | ✓ | | | ✓ | | ✓ | | | | | ✓ | |
| 2019 | Gupta et al. (2019) | (N.A.) | | ✓ | ✓ | | ✓ | ✓ | | | | | ✓ | | | | |
| 2019 | Wang et al. (2019) | (N.A.) | ✓ | ✓ | ✓ | | | | ✓ | | ✓ | | ✓ | | ✓ | | |
| 2019 | Manotumruksa et al. (2019a) | CRCF | ✓ | | | ✓ | ✓ | | | | ✓ | | | | | ✓ | |
| 2019 | Kala and Nandhini (2019) | CCS-POI-RS | ✓ | | | ✓ | ✓ | | | | ✓ | | | | | ✓ | |
| 2019 | Zhang et al. (2019b) | HWREC | ✓ | | | ✓ | ✓ | ✓ | | | | | | ✓ | | ✓ | |
| 2019 | Guo et al. (2019a) | L-WMF | ✓ | | | | | ✓ | ✓ | ✓ | | | | | | ✓ | |
| 2019 | Jiao et al. (2019a) | (N.A.) | ✓ | | | ✓ | ✓ | | ✓ | | | | ✓ | | | ✓ | |
| 2019 | Zhou et al. (2019a) | UFC | ✓ | ✓ | | | | ✓ | | ✓ | | ✓ | ✓ | | | | |
| 2019 | Zhou et al. (2019b) | AKAWO | | | ✓ | | ✓ | | | | | | ✓ | ✓ | ✓ | | |
| 2019 | Baral et al. (2019) | HiRecS | ✓ | ✓ | ✓ | | ✓ | | | ✓ | | | ✓ | ✓ | ✓ | | |
| 2019 | Li et al. (2019a) | GPDM, PPDM | ✓ | | ✓ | ✓ | | | ✓ | ✓ | | | | | | ✓ | |
| 2019 | Rahmani et al. (2019) | CATAPE | | | ✓ | ✓ | | | | | ✓ | | | | | ✓ | |
| 2019 | Ying et al. (2019) | MEAP-T | | | | ✓ | ✓ | | ✓ | ✓ | | | | | | ✓ | |
| 2019 | Zhang et al. (2019c) | VCG | ✓ | ✓ | ✓ | | | | ✓ | ✓ | | | ✓ | | ✓ | | |
| 2019 | Hao et al. (2019) | RealTime-MF | | | ✓ | | ✓ | | ✓ | | ✓ | | | | ✓ | | |
| 2019 | Gao et al. (2019) | ST-RNet | ✓ | | | | ✓ | | | | ✓ | | | | ✓ | | |
| 2019 | Cai et al. (2019) | LC-G-P | ✓ | ✓ | | | | ✓ | | | | ✓ | ✓ | | | | |
| 2019 | Yin et al. (2019) | ADPR | ✓ | | ✓ | | | | | ✓ | ✓ | | | | | ✓ | |
| 2019 | Su et al. (2019a) | PRFPF | ✓ | ✓ | | | | | ✓ | ✓ | | | | | | ✓ | |
| 2019 | Jang et al. (2019) | (N.A.) | ✓ | | | ✓ | | | | | | ✓ | | | ✓ | | |
| 2019 | Liu et al. (2019a) | GT-HAN | ✓ | | | ✓ | ✓ | | | | ✓ | | | | | ✓ | |
| 2019 | Li et al. (2019b) | LORI | ✓ | | | | ✓ | ✓ | | ✓ | | | ✓ | | | ✓ | |
| 2019 | Zhan et al. (2019) | (N.A.) | | | ✓ | ✓ | | | | | ✓ | | | | | ✓ | |
| 2019 | Lu et al. (2019) | PEU-RNN | ✓ | | | ✓ | | | | | ✓ | | | | ✓ | | |
| 2019 | Hosseini et al. (2019) | MATI | | | | | ✓ | | ✓ | ✓ | | | | | ✓ | | |
| 2019 | Jiao et al. (2019b) | R2SIGTP | ✓ | | | | ✓ | | ✓ | | | | ✓ | | | | ✓ |
| 2019 | Yu et al. (2019) | DMGMT | ✓ | | ✓ | | ✓ | | ✓ | ✓ | | | ✓ | | | | |
| 2019 | Li et al. (2019c) | PA-Seq2Seq | ✓ | | | ✓ | ✓ | | | | ✓ | | | | | ✓ | |
| 2019 | Liu et al. (2019b) | HMM | | | ✓ | ✓ | | | | ✓ | | | | | | | |
| 2019 | Xing et al. (2019) | CPC | ✓ | | ✓ | | | | ✓ | ✓ | ✓ | | | | ✓ | | |
| 2019 | Yang et al. (2019) | NRLRS | ✓ | ✓ | ✓ | | | | ✓ | ✓ | | ✓ | | | ✓ | | |
| 2019 | Geng et al. (2019) | MLR | ✓ | ✓ | | | | ✓ | | ✓ | | | | ✓ | ✓ | | |
| 2019 | Chuang et al. (2019) | UGR | ✓ | | | | | ✓ | | | | | ✓ | | ✓ | | |
| 2019 | Gan and Gao (2019) | U-CF-Memory | | | | | ✓ | ✓ | | | | | | | | ✓ | |
| 2019 | Guo et al. (2019b) | AGS-MF | ✓ | ✓ | ✓ | | | | ✓ | | | ✓ | | | | ✓ | |
| 2019 | Tang et al. (2019) | (N.A.) | | ✓ | ✓ | ✓ | | ✓ | | ✓ | | | ✓ | | | | |
| 2019 | Si et al. (2019) | APRA-SA | ✓ | | | | ✓ | | | ✓ | | | ✓ | | ✓ | | |
| 2019 | Guo et al. (2019c) | Geo-SRank | ✓ | ✓ | | | | | ✓ | ✓ | ✓ | | | | ✓ | | |
| 2019 | Zhu and Guo (2019) | SG-NeuRec | ✓ | ✓ | | | ✓ | | | | ✓ | | | | | ✓ | |
| 2019 | Xu et al. (2019) | SSSER | ✓ | ✓ | | ✓ | | | | ✓ | ✓ | | | | ✓ | | |
| 2019 | Rahimi et al. (2020) | BLR | ✓ | | ✓ | ✓ | ✓ | | | ✓ | | | ✓ | | ✓ | | |
| 2019 | Doan et al. (2019) | ASTEN | ✓ | | | ✓ | ✓ | | | | ✓ | | | | ✓ | | |
| 2019 | Yu et al. (2019) | HeteGeoRanRec | ✓ | ✓ | ✓ | | | | ✓ | ✓ | | | | | | ✓ | |
| 2019 | Qian et al. (2019) | STA | ✓ | | | | ✓ | ✓ | ✓ | | | | | | | ✓ | |
| 2019 | Su et al. (2019b) | WBPR-DST | ✓ | | | | ✓ | | | ✓ | | | | | ✓ | | |
| 2019 | Zeng et al. (2019) | RBMNMF | | | | | | | ✓ | | ✓ | | ✓ | | ✓ | | |

**Table A.5:** Performance for all the cities, only showing the best recommender for each family. Notation and cutoffs like in Table 6.7.

| | | | | Accuracy | | | Seq. Accuracy | | | Non Accuracy | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| City | Family | Rec | P | nDCG | TFP | $P_s$ | $nDCG_s$ | $FP_s$ | Dist | Gini | EPC |
| NYC | Basic | Pop | 0.144 | 0.417 | 0.326 | 0.139 | 0.402 | 0.284 | 43.9 | 0.001 | 0.904 |
| | Classic | BPRMF | 0.145 | 0.418 | 0.330 | 0.141 | 0.404 | 0.285 | 45.3 | 0.001 | 0.905 |
| | Temporal | BFsUB | 0.145 | 0.420 | 0.336 | 0.140 | 0.404 | 0.302 | †**42.4** | †**0.003** | †**0.920** |
| | Geo | IRenMF | †**0.147** | †**0.420** | †**0.345** | †**0.143** | †**0.405** | †**0.306** | 43.9 | 0.002 | 0.916 |
| | Tour | ItemMC | 0.132 | 0.404 | 0.295 | 0.129 | 0.391 | 0.279 | 44.9 | 0.001 | 0.911 |
| | Skylines | TestOrder | ▴**0.453** | ▴**0.928** | ▴**0.453** | ▴**0.453** | ▴**0.909** | ▴**0.453** | ▴**8.3** | ▴**0.023** | ▴**0.979** |
| TOK | Basic | Pop | 0.127 | 0.385 | 0.385 | 0.126 | 0.374 | 0.372 | 25.8 | 0.001 | 0.849 |
| | Classic | BPRMF | 0.128 | 0.386 | †**0.389** | 0.126 | 0.374 | †**0.375** | 22.7 | 0.001 | 0.852 |
| | Temporal | BFsUB | †**0.132** | 0.389 | 0.382 | †**0.129** | 0.377 | 0.365 | 25.4 | 0.001 | 0.866 |
| | Geo | IRenMF | 0.131 | †**0.389** | 0.376 | 0.129 | †**0.377** | 0.361 | 23.7 | 0.002 | 0.870 |
| | Tour | ItemMC | 0.128 | 0.388 | 0.366 | 0.127 | 0.376 | 0.351 | †**17.9** | †**0.002** | †**0.878** |
| | Skylines | TestOrder | ▴**0.443** | ▴**0.885** | ▴**0.443** | ▴**0.443** | ▴**0.865** | ▴**0.443** | ▴**8.1** | ▴**0.019** | ▴**0.965** |
| ROM | Basic | Pop | 0.227 | 0.508 | 0.517 | 0.202 | 0.447 | 0.464 | 5.0 | 0.050 | 0.848 |
| | Classic | BPRMF | 0.226 | 0.508 | 0.518 | 0.201 | 0.447 | 0.460 | 6.3 | 0.050 | 0.848 |
| | Temporal | FPMC | 0.227 | 0.508 | 0.516 | 0.201 | 0.447 | 0.469 | 4.9 | 0.052 | 0.849 |
| | Geo | RankGeoFM | 0.211 | 0.486 | 0.516 | 0.187 | 0.427 | 0.457 | 5.6 | †**0.082** | 0.865 |
| | Tour | ItemMC | †**0.231** | †**0.537** | ▴**0.519** | †**0.212** | †**0.477** | †**0.473** | ▴**2.0** | 0.076 | †**0.871** |
| | Skylines | TestOrder | ▴**0.481** | ▴**0.915** | 0.481 | ▴**0.481** | ▴**0.858** | ▴**0.481** | 2.1 | ▴**0.217** | ▴**0.915** |
| PJ | Basic | Pop | 0.128 | 0.413 | 0.249 | 0.126 | 0.404 | 0.245 | 35.0 | 0.001 | 0.915 |
| | Classic | BPRMF | 0.131 | 0.418 | 0.274 | 0.128 | 0.408 | 0.270 | 30.0 | 0.001 | 0.917 |
| | Temporal | BFsUB | †**0.135** | †**0.423** | †**0.296** | †**0.134** | †**0.416** | 0.285 | †**26.6** | 0.003 | †**0.933** |
| | Geo | PGN | 0.129 | 0.415 | 0.296 | 0.126 | 0.406 | †**0.286** | 30.0 | †**0.008** | 0.931 |
| | Tour | ItemMC | 0.127 | 0.412 | 0.244 | 0.125 | 0.403 | 0.240 | 28.4 | 0.002 | 0.918 |
| | Skylines | TestOrder | ▴**0.369** | ▴**0.864** | ▴**0.368** | ▴**0.369** | ▴**0.853** | ▴**0.368** | ▴**7.0** | ▴**0.023** | ▴**0.980** |

feedback captured by the system in a pairwise fashion (Rendle et al., 2009), which fits the POI recommendation scenario very well and, according to the results, could be a key technique to take advantage of the available data.

### A.2.2 Complete performance results of reranking strategies

In Tables A.6, A.7, A.8, A.9, and A.10 we present the complete results for the same reranker strategies as in Table 6.9 (see Section 6.3.6), but not limited to distance and item-level and category-level accuracy. We observe from these tables a similar behavior in every city although each one has different characteristics. Firstly, the feature-based Markov Chain reranker ($f_{seq}^{feat}$) is usually the worst (together with the random one, $f_{seq}^{rnd}$), especially in terms of $FP_s$, but also for $nDCG_s$, where it tends to decrease the performance with respect to the baseline (base recommender without reranking).

Secondly, it is interesting to observe that the baseline is often providing the longest routes to the users, hence, the reranking strategies allow to create more realistic and affordable routes to the final user. Another example of the varied range of improvements in different dimensions (below those already provided in the main part of Chapter 6) is the following: the distance of the route obtained by the recommender from the Basic family (Pop) in Petaling Jaya is 34.95 Km, while applying $f_{seq}^{dist}$ this distance is reduced to 7.21 Km – a decrease of 80%. We believe these outcomes (together with those already presented) are very positive and promising, as route recommenders normally require many different information sources and long execution times in order to work well, but using this kind of techniques may help to find simple solutions for these cases

**Table A.6:** Performance of the rerankers for the Basic family of recommenders. Same notation as in Table 6.7.

| | | Accuracy | | | Seq. Accuracy | | | Non-Accuracy | | |
|---|---|---|---|---|---|---|---|---|---|---|
| City | Reranker | P | nDCG | TFP | $P_s$ | $nDCG_s$ | $FP_s$ | Dist | Gini | EPC |
| New York | Baseline | 0.144 | 0.417 | 0.326 | 0.139 | 0.402 | 0.284 | 43.9 | 0.001 | 0.904 |
| | $f_{seq}^{rnd}$ | 0.132 | 0.396 | 0.324 | 0.128 | 0.383 | 0.297 | 28.0 | ▲**0.002** | 0.935 |
| | $f_{seq}^{dist}$ | 0.137 | 0.408 | †**0.331** | 0.134 | 0.396 | ▲**0.308** | ▲**4.1** | 0.001 | ▲**0.940** |
| | $f_{seq}^{feat}$ | 0.143 | 0.415 | 0.285 | 0.138 | 0.400 | 0.267 | 33.3 | 0.001 | 0.914 |
| | $f_{seq}^{item}$ | 0.137 | 0.412 | 0.309 | 0.134 | 0.399 | 0.279 | 37.8 | 0.001 | 0.908 |
| | $f_{seq}^{rec}$ | †**0.147** | †**0.422** | 0.331 | †**0.142** | †**0.406** | 0.298 | 42.4 | 0.001 | 0.913 |
| | $f_{seq}^{lcs}$ | 0.137 | 0.406 | 0.313 | 0.136 | 0.395 | 0.285 | 17.8 | 0.001 | 0.918 |
| | $f_{seq}^{stree}$ | 0.143 | 0.416 | 0.326 | 0.140 | 0.402 | 0.289 | 38.4 | 0.001 | 0.905 |
| | $f_{seq}^{oracle}$ | ▲**0.163** | ▲**0.479** | ▲**0.332** | ▲**0.163** | ▲**0.468** | 0.296 | 43.2 | 0.001 | 0.904 |
| Tokyo | Baseline | 0.127 | 0.385 | †**0.385** | 0.126 | 0.374 | 0.372 | 25.8 | 0.001 | 0.849 |
| | $f_{seq}^{rnd}$ | 0.118 | 0.368 | 0.379 | 0.116 | 0.356 | 0.365 | 27.5 | ▲**0.001** | ▲**0.894** |
| | $f_{seq}^{dist}$ | 0.122 | 0.378 | 0.383 | 0.121 | 0.367 | ▲**0.373** | ▲**6.9** | 0.001 | 0.893 |
| | $f_{seq}^{feat}$ | 0.126 | 0.384 | 0.315 | 0.124 | 0.372 | 0.315 | 26.0 | 0.001 | 0.860 |
| | $f_{seq}^{item}$ | 0.127 | 0.386 | 0.369 | 0.125 | 0.374 | 0.359 | 18.2 | 0.001 | 0.856 |
| | $f_{seq}^{rec}$ | †**0.129** | †**0.386** | 0.358 | †**0.127** | †**0.374** | 0.346 | 24.5 | 0.001 | 0.858 |
| | $f_{seq}^{lcs}$ | 0.123 | 0.378 | 0.351 | 0.122 | 0.367 | 0.344 | 10.4 | 0.001 | 0.880 |
| | $f_{seq}^{stree}$ | 0.127 | 0.383 | 0.372 | 0.126 | 0.371 | 0.359 | 15.5 | 0.001 | 0.862 |
| | $f_{seq}^{oracle}$ | ▲**0.138** | ▲**0.419** | ▲**0.386** | ▲**0.138** | ▲**0.408** | 0.372 | 25.1 | 0.001 | 0.849 |
| Rome | Baseline | 0.227 | 0.508 | 0.517 | 0.202 | 0.447 | 0.464 | 5.0 | 0.050 | 0.848 |
| | $f_{seq}^{rnd}$ | 0.186 | 0.449 | 0.511 | 0.171 | 0.402 | 0.452 | 5.9 | ▲**0.077** | ▲**0.885** |
| | $f_{seq}^{dist}$ | 0.225 | 0.523 | †**0.523** | 0.211 | 0.469 | †**0.474** | ▲**1.4** | 0.072 | 0.884 |
| | $f_{seq}^{feat}$ | 0.199 | 0.476 | 0.423 | 0.181 | 0.422 | 0.371 | 5.0 | 0.048 | 0.876 |
| | $f_{seq}^{item}$ | †**0.232** | †**0.534** | 0.520 | †**0.214** | †**0.473** | 0.469 | 1.8 | 0.067 | 0.872 |
| | $f_{seq}^{rec}$ | 0.207 | 0.477 | 0.510 | 0.187 | 0.422 | 0.452 | 6.0 | 0.058 | 0.857 |
| | $f_{seq}^{lcs}$ | 0.202 | 0.487 | 0.488 | 0.189 | 0.440 | 0.446 | 2.3 | 0.069 | 0.881 |
| | $f_{seq}^{stree}$ | 0.218 | 0.502 | 0.520 | 0.198 | 0.446 | 0.466 | 3.2 | 0.063 | 0.861 |
| | $f_{seq}^{oracle}$ | ▲**0.289** | ▲**0.659** | ▲**0.531** | ▲**0.289** | ▲**0.614** | ▲**0.482** | 4.2 | 0.052 | 0.850 |
| Petaling Jaya | Baseline | 0.128 | 0.413 | 0.249 | 0.126 | 0.404 | 0.245 | 35.0 | 0.001 | 0.915 |
| | $f_{seq}^{rnd}$ | 0.118 | 0.394 | 0.283 | 0.117 | 0.387 | 0.274 | 29.5 | ▲**0.002** | ▲**0.939** |
| | $f_{seq}^{dist}$ | †**0.132** | †**0.418** | ▲**0.311** | †**0.130** | †**0.409** | ▲**0.296** | ▲**7.2** | 0.002 | 0.937 |
| | $f_{seq}^{feat}$ | 0.129 | 0.412 | 0.296 | 0.126 | 0.402 | 0.267 | 33.2 | 0.001 | 0.917 |
| | $f_{seq}^{item}$ | 0.130 | 0.417 | 0.267 | 0.127 | 0.408 | 0.262 | 19.5 | 0.001 | 0.920 |
| | $f_{seq}^{rec}$ | 0.131 | 0.415 | 0.277 | 0.129 | 0.407 | 0.271 | 26.3 | 0.002 | 0.924 |
| | $f_{seq}^{lcs}$ | 0.130 | 0.413 | 0.293 | 0.128 | 0.403 | 0.274 | 14.8 | 0.002 | 0.926 |
| | $f_{seq}^{stree}$ | 0.130 | 0.413 | 0.283 | 0.126 | 0.403 | 0.263 | 25.9 | 0.001 | 0.919 |
| | $f_{seq}^{oracle}$ | ▲**0.144** | ▲**0.463** | 0.251 | ▲**0.144** | ▲**0.456** | 0.247 | 34.2 | 0.001 | 0.916 |

**Table A.7:** Performance of the rerankers for the Classic family of recommenders. Same notation as in Table 6.7.

| | | | Accuracy | | | Seq. Accuracy | | | Non-Accuracy | |
|---|---|---|---|---|---|---|---|---|---|---|
| City | Reranker | P | nDCG | TFP | $P_s$ | $nDCG_s$ | $FP_s$ | Dist | Gini | EPC |
| New York | Baseline | 0.145 | 0.418 | 0.330 | 0.141 | 0.404 | 0.285 | 45.3 | 0.001 | 0.905 |
| | $f_{seq}^{rnd}$ | 0.130 | 0.393 | 0.320 | 0.127 | 0.382 | 0.292 | 30.3 | ▲**0.002** | 0.935 |
| | $f_{seq}^{dist}$ | 0.136 | 0.407 | 0.332 | 0.133 | 0.395 | ▲**0.309** | ▲**4.2** | 0.002 | ▲**0.941** |
| | $f_{seq}^{feat}$ | 0.142 | 0.413 | 0.287 | 0.137 | 0.398 | 0.267 | 33.5 | 0.002 | 0.915 |
| | $f_{seq}^{item}$ | 0.138 | 0.413 | 0.307 | 0.136 | 0.400 | 0.276 | 38.0 | 0.001 | 0.908 |
| | $f_{seq}^{rec}$ | †**0.147** | †**0.422** | 0.332 | †**0.142** | †**0.406** | 0.300 | 42.4 | 0.002 | 0.913 |
| | $f_{seq}^{lcs}$ | 0.138 | 0.407 | 0.316 | 0.136 | 0.395 | 0.284 | 17.9 | 0.002 | 0.919 |
| | $f_{seq}^{stree}$ | 0.145 | 0.418 | ▲**0.335** | 0.141 | 0.404 | 0.294 | 38.6 | 0.001 | 0.906 |
| | $f_{seq}^{oracle}$ | ▲**0.163** | ▲**0.479** | 0.334 | ▲**0.163** | ▲**0.468** | 0.300 | 44.3 | 0.001 | 0.905 |
| Tokyo | Baseline | †**0.128** | †**0.386** | †**0.389** | †**0.126** | †**0.374** | ▲**0.375** | 22.7 | 0.001 | 0.852 |
| | $f_{seq}^{rnd}$ | 0.116 | 0.366 | 0.371 | 0.115 | 0.356 | 0.357 | 27.0 | ▲**0.001** | ▲**0.897** |
| | $f_{seq}^{dist}$ | 0.123 | 0.378 | 0.374 | 0.122 | 0.367 | 0.362 | ▲**6.8** | 0.001 | 0.896 |
| | $f_{seq}^{feat}$ | 0.125 | 0.384 | 0.315 | 0.124 | 0.372 | 0.315 | 26.0 | 0.001 | 0.860 |
| | $f_{seq}^{item}$ | 0.127 | 0.385 | 0.370 | 0.126 | 0.373 | 0.360 | 17.4 | 0.001 | 0.856 |
| | $f_{seq}^{rec}$ | 0.127 | 0.385 | 0.358 | 0.126 | 0.373 | 0.346 | 24.7 | 0.001 | 0.859 |
| | $f_{seq}^{lcs}$ | 0.123 | 0.377 | 0.349 | 0.122 | 0.365 | 0.343 | 10.3 | 0.001 | 0.885 |
| | $f_{seq}^{stree}$ | 0.127 | 0.382 | 0.371 | 0.124 | 0.369 | 0.360 | 13.9 | 0.001 | 0.866 |
| | $f_{seq}^{oracle}$ | ▲**0.137** | ▲**0.416** | ▲**0.390** | ▲**0.137** | ▲**0.405** | 0.375 | 22.4 | 0.001 | 0.852 |
| Rome | Baseline | 0.226 | 0.508 | 0.518 | 0.201 | 0.447 | 0.460 | 6.3 | 0.050 | 0.848 |
| | $f_{seq}^{rnd}$ | 0.187 | 0.451 | 0.512 | 0.171 | 0.403 | 0.450 | 5.9 | ▲**0.077** | ▲**0.886** |
| | $f_{seq}^{dist}$ | 0.223 | 0.521 | †**0.525** | 0.210 | 0.468 | †**0.475** | ▲**1.4** | 0.071 | 0.884 |
| | $f_{seq}^{feat}$ | 0.200 | 0.478 | 0.426 | 0.183 | 0.424 | 0.373 | 5.0 | 0.047 | 0.875 |
| | $f_{seq}^{item}$ | †**0.234** | †**0.536** | 0.519 | †**0.216** | †**0.476** | 0.468 | 1.8 | 0.067 | 0.871 |
| | $f_{seq}^{rec}$ | 0.207 | 0.477 | 0.509 | 0.187 | 0.422 | 0.452 | 6.0 | 0.058 | 0.857 |
| | $f_{seq}^{lcs}$ | 0.200 | 0.487 | 0.487 | 0.188 | 0.440 | 0.447 | 2.3 | 0.068 | 0.880 |
| | $f_{seq}^{stree}$ | 0.218 | 0.502 | 0.520 | 0.199 | 0.447 | 0.465 | 3.7 | 0.063 | 0.861 |
| | $f_{seq}^{oracle}$ | ▲**0.287** | ▲**0.657** | ▲**0.530** | ▲**0.287** | ▲**0.612** | ▲**0.482** | 4.9 | 0.052 | 0.850 |
| Petaling Jaya | Baseline | 0.131 | 0.418 | 0.274 | 0.128 | 0.408 | 0.270 | 30.0 | 0.001 | 0.917 |
| | $f_{seq}^{rnd}$ | 0.124 | 0.402 | 0.288 | 0.122 | 0.394 | 0.278 | 30.7 | ▲**0.002** | ▲**0.939** |
| | $f_{seq}^{dist}$ | 0.132 | 0.418 | ▲**0.309** | †**0.130** | †**0.410** | ▲**0.294** | ▲**7.4** | 0.002 | 0.938 |
| | $f_{seq}^{feat}$ | 0.128 | 0.411 | 0.298 | 0.125 | 0.402 | 0.269 | 34.0 | 0.001 | 0.918 |
| | $f_{seq}^{item}$ | 0.130 | †**0.419** | 0.272 | 0.127 | 0.409 | 0.268 | 18.5 | 0.001 | 0.920 |
| | $f_{seq}^{rec}$ | 0.131 | 0.415 | 0.279 | 0.129 | 0.407 | 0.273 | 26.5 | 0.002 | 0.924 |
| | $f_{seq}^{lcs}$ | †**0.132** | 0.416 | 0.302 | 0.128 | 0.405 | 0.279 | 13.2 | 0.002 | 0.926 |
| | $f_{seq}^{stree}$ | 0.131 | 0.415 | 0.297 | 0.128 | 0.405 | 0.275 | 22.1 | 0.001 | 0.919 |
| | $f_{seq}^{oracle}$ | ▲**0.144** | ▲**0.462** | 0.275 | ▲**0.144** | ▲**0.455** | 0.269 | 29.0 | 0.001 | 0.917 |

**Table A.8:** Performance of the rerankers for the Temporal family of recommenders. Same notation as in Table 6.7.

| | | Accuracy | | | Seq. Accuracy | | | Non Accuracy | | |
|---|---|---|---|---|---|---|---|---|---|---|
| City | Reranker | P | nDCG | TFP | $P_s$ | $nDCG_s$ | $FP_s$ | Dist | Gini | EPC |
| New York | Baseline | 0.145 | 0.420 | 0.336 | **†0.140** | **†0.404** | 0.302 | 42.4 | 0.003 | 0.920 |
| | $f_{seq}^{rnd}$ | 0.127 | 0.390 | 0.349 | 0.125 | 0.379 | 0.317 | 25.3 | **▲0.006** | 0.945 |
| | $f_{seq}^{dist}$ | 0.132 | 0.399 | **▲0.358** | 0.130 | 0.389 | **▲0.319** | **▲3.5** | 0.006 | **▲0.958** |
| | $f_{seq}^{feat}$ | 0.134 | 0.401 | 0.300 | 0.130 | 0.388 | 0.272 | 30.6 | 0.006 | 0.933 |
| | $f_{seq}^{item}$ | 0.142 | 0.414 | 0.328 | 0.137 | 0.400 | 0.293 | 37.2 | 0.002 | 0.916 |
| | $f_{seq}^{rec}$ | **†0.145** | **†0.420** | 0.344 | 0.139 | 0.403 | 0.309 | 41.5 | 0.003 | 0.921 |
| | $f_{seq}^{lcs}$ | 0.135 | 0.400 | 0.346 | 0.132 | 0.388 | 0.314 | 10.8 | 0.006 | 0.940 |
| | $f_{seq}^{stree}$ | 0.142 | 0.412 | 0.351 | 0.138 | 0.398 | 0.311 | 29.6 | 0.004 | 0.924 |
| | $f_{seq}^{oracle}$ | **▲0.160** | **▲0.472** | 0.340 | **▲0.160** | **▲0.462** | 0.308 | 39.9 | 0.003 | 0.920 |
| Tokyo | Baseline | **†0.132** | **†0.389** | 0.382 | **†0.129** | **†0.377** | 0.365 | 25.4 | 0.001 | 0.866 |
| | $f_{seq}^{rnd}$ | 0.120 | 0.372 | **▲0.390** | 0.118 | 0.361 | **▲0.372** | 26.4 | 0.002 | 0.910 |
| | $f_{seq}^{dist}$ | 0.127 | 0.383 | 0.389 | 0.126 | 0.371 | 0.367 | **▲6.5** | **▲0.002** | **▲0.914** |
| | $f_{seq}^{feat}$ | 0.127 | 0.386 | 0.317 | 0.126 | 0.374 | 0.316 | 26.3 | 0.001 | 0.866 |
| | $f_{seq}^{item}$ | 0.128 | 0.385 | 0.365 | 0.126 | 0.374 | 0.355 | 18.4 | 0.001 | 0.863 |
| | $f_{seq}^{rec}$ | 0.130 | 0.388 | 0.373 | 0.128 | 0.376 | 0.359 | 25.4 | 0.001 | 0.868 |
| | $f_{seq}^{lcs}$ | 0.128 | 0.383 | 0.357 | 0.126 | 0.370 | 0.347 | 9.7 | 0.002 | 0.898 |
| | $f_{seq}^{stree}$ | 0.130 | 0.385 | 0.373 | 0.128 | 0.373 | 0.360 | 14.5 | 0.002 | 0.880 |
| | $f_{seq}^{oracle}$ | **▲0.144** | **▲0.427** | 0.384 | **▲0.144** | **▲0.416** | 0.367 | 24.5 | 0.001 | 0.866 |
| Rome | Baseline | 0.227 | 0.508 | 0.516 | 0.201 | 0.447 | **†0.469** | 4.9 | 0.052 | 0.849 |
| | $f_{seq}^{rnd}$ | 0.188 | 0.453 | 0.507 | 0.175 | 0.409 | 0.449 | 6.0 | **▲0.080** | **▲0.887** |
| | $f_{seq}^{dist}$ | 0.221 | 0.518 | **†0.520** | 0.207 | 0.464 | 0.468 | **▲1.4** | 0.075 | 0.886 |
| | $f_{seq}^{feat}$ | 0.199 | 0.476 | 0.431 | 0.180 | 0.421 | 0.375 | 5.0 | 0.050 | 0.874 |
| | $f_{seq}^{item}$ | **†0.234** | **†0.535** | 0.518 | **†0.214** | **†0.474** | 0.465 | 1.9 | 0.068 | 0.871 |
| | $f_{seq}^{rec}$ | 0.207 | 0.476 | 0.508 | 0.187 | 0.422 | 0.452 | 6.1 | 0.060 | 0.857 |
| | $f_{seq}^{lcs}$ | 0.204 | 0.489 | 0.488 | 0.191 | 0.441 | 0.447 | 2.3 | 0.071 | 0.880 |
| | $f_{seq}^{stree}$ | 0.218 | 0.501 | 0.519 | 0.198 | 0.445 | 0.468 | 3.1 | 0.065 | 0.863 |
| | $f_{seq}^{oracle}$ | **▲0.285** | **▲0.654** | **▲0.529** | **▲0.285** | **▲0.608** | **▲0.482** | 4.1 | 0.054 | 0.851 |
| Petaling Jaya | Baseline | **†0.135** | **†0.423** | 0.296 | **†0.134** | **†0.416** | 0.285 | 26.6 | 0.003 | 0.933 |
| | $f_{seq}^{rnd}$ | 0.114 | 0.389 | 0.325 | 0.114 | 0.383 | 0.308 | 28.1 | 0.006 | 0.952 |
| | $f_{seq}^{dist}$ | 0.132 | 0.420 | **▲0.346** | 0.130 | 0.412 | **▲0.326** | **▲5.6** | **▲0.006** | **▲0.954** |
| | $f_{seq}^{feat}$ | 0.126 | 0.406 | 0.327 | 0.123 | 0.397 | 0.291 | 30.2 | 0.004 | 0.932 |
| | $f_{seq}^{item}$ | 0.131 | 0.420 | 0.294 | 0.130 | 0.412 | 0.283 | 17.5 | 0.002 | 0.928 |
| | $f_{seq}^{rec}$ | 0.131 | 0.416 | 0.304 | 0.129 | 0.407 | 0.292 | 26.1 | 0.003 | 0.933 |
| | $f_{seq}^{lcs}$ | 0.133 | 0.416 | 0.330 | 0.130 | 0.407 | 0.311 | 10.9 | 0.005 | 0.946 |
| | $f_{seq}^{stree}$ | 0.134 | 0.419 | 0.324 | 0.131 | 0.411 | 0.301 | 17.3 | 0.004 | 0.938 |
| | $f_{seq}^{oracle}$ | **▲0.145** | **▲0.463** | 0.299 | **▲0.145** | **▲0.457** | 0.287 | 25.8 | 0.003 | 0.933 |

**Table A.9:** Performance of the rerankers for the Geo family of recommenders. Same notation as in Table 6.7.

| | | Accuracy | | | Seq. Accuracy | | | Non-Accuracy | | |
|---|---|---|---|---|---|---|---|---|---|---|
| City | Reranker | P | nDCG | TFP | $P_s$ | $nDCG_s$ | $FP_s$ | Dist | Gini | EPC |
| New York | Baseline | **†0.147** | 0.420 | 0.345 | **†0.143** | 0.405 | 0.306 | 43.9 | 0.002 | 0.916 |
| | $f_{seq}^{rnd}$ | 0.126 | 0.388 | 0.346 | 0.125 | 0.378 | 0.307 | 22.5 | **▲0.004** | 0.946 |
| | $f_{seq}^{dist}$ | 0.131 | 0.397 | 0.353 | 0.128 | 0.385 | 0.315 | **▲3.6** | 0.004 | **▲0.959** |
| | $f_{seq}^{feat}$ | 0.137 | 0.406 | 0.311 | 0.133 | 0.393 | 0.281 | 32.8 | 0.004 | 0.930 |
| | $f_{seq}^{item}$ | 0.142 | 0.416 | 0.330 | 0.138 | 0.402 | 0.291 | 37.1 | 0.002 | 0.915 |
| | $f_{seq}^{rec}$ | 0.145 | **†0.421** | 0.348 | 0.141 | **†0.405** | 0.311 | 42.0 | 0.003 | 0.920 |
| | $f_{seq}^{lcs}$ | 0.138 | 0.402 | 0.346 | 0.134 | 0.390 | 0.311 | 11.9 | 0.004 | 0.938 |
| | $f_{seq}^{stree}$ | 0.146 | 0.416 | **▲0.362** | 0.142 | 0.402 | **▲0.321** | 31.3 | 0.003 | 0.921 |
| | $f_{seq}^{oracle}$ | **▲0.162** | **▲0.475** | 0.348 | **▲0.162** | **▲0.464** | 0.314 | 41.7 | 0.002 | 0.917 |
| Tokyo | Baseline | **†0.131** | **†0.389** | 0.376 | **†0.129** | **†0.377** | 0.361 | 23.7 | 0.002 | 0.870 |
| | $f_{seq}^{rnd}$ | 0.122 | 0.372 | 0.394 | 0.120 | 0.360 | 0.374 | 25.3 | 0.003 | 0.910 |
| | $f_{seq}^{dist}$ | 0.127 | 0.382 | **▲0.398** | 0.125 | 0.370 | **▲0.375** | **▲5.9** | **▲0.003** | **▲0.917** |
| | $f_{seq}^{feat}$ | 0.126 | 0.384 | 0.319 | 0.125 | 0.372 | 0.317 | 26.2 | 0.001 | 0.866 |
| | $f_{seq}^{item}$ | 0.128 | 0.386 | 0.366 | 0.126 | 0.374 | 0.354 | 18.2 | 0.002 | 0.866 |
| | $f_{seq}^{rec}$ | 0.130 | 0.388 | 0.365 | 0.128 | 0.376 | 0.354 | 25.2 | 0.002 | 0.870 |
| | $f_{seq}^{lcs}$ | 0.125 | 0.379 | 0.355 | 0.123 | 0.367 | 0.346 | 9.0 | 0.003 | 0.902 |
| | $f_{seq}^{stree}$ | 0.129 | 0.383 | 0.368 | 0.127 | 0.372 | 0.357 | 12.9 | 0.002 | 0.883 |
| | $f_{seq}^{oracle}$ | **▲0.144** | **▲0.428** | 0.380 | **▲0.144** | **▲0.417** | 0.363 | 23.0 | 0.002 | 0.870 |
| Rome | Baseline | 0.211 | 0.486 | 0.516 | 0.187 | 0.427 | 0.457 | 5.6 | 0.082 | 0.865 |
| | $f_{seq}^{rnd}$ | 0.178 | 0.440 | 0.507 | 0.167 | 0.397 | 0.447 | 5.9 | **▲0.127** | **▲0.900** |
| | $f_{seq}^{dist}$ | 0.213 | 0.507 | **†0.520** | 0.201 | 0.456 | **†0.468** | **▲1.4** | 0.114 | 0.899 |
| | $f_{seq}^{feat}$ | 0.189 | 0.464 | 0.411 | 0.174 | 0.414 | 0.364 | 5.3 | 0.077 | 0.890 |
| | $f_{seq}^{item}$ | **†0.225** | **†0.526** | 0.519 | **†0.207** | **†0.467** | 0.466 | 2.1 | 0.081 | 0.872 |
| | $f_{seq}^{rec}$ | 0.200 | 0.469 | 0.510 | 0.181 | 0.417 | 0.453 | 6.0 | 0.073 | 0.862 |
| | $f_{seq}^{lcs}$ | 0.189 | 0.472 | 0.483 | 0.178 | 0.426 | 0.440 | 2.2 | 0.115 | 0.896 |
| | $f_{seq}^{stree}$ | 0.202 | 0.481 | 0.514 | 0.186 | 0.431 | 0.458 | 3.5 | 0.103 | 0.880 |
| | $f_{seq}^{oracle}$ | **▲0.268** | **▲0.630** | **▲0.527** | **▲0.268** | **▲0.586** | **▲0.472** | 4.6 | 0.083 | 0.866 |
| Petaling Jaya | Baseline | 0.129 | 0.415 | 0.296 | 0.126 | 0.406 | 0.286 | 30.0 | 0.008 | 0.931 |
| | $f_{seq}^{rnd}$ | 0.121 | 0.397 | 0.322 | 0.119 | 0.390 | 0.307 | 25.1 | 0.015 | 0.949 |
| | $f_{seq}^{dist}$ | 0.127 | 0.413 | **▲0.337** | 0.125 | 0.405 | **▲0.315** | **▲5.8** | **▲0.023** | **▲0.960** |
| | $f_{seq}^{feat}$ | 0.126 | 0.406 | 0.316 | 0.123 | 0.397 | 0.282 | 26.8 | 0.007 | 0.929 |
| | $f_{seq}^{item}$ | 0.131 | **†0.420** | 0.279 | **†0.130** | **†0.412** | 0.270 | 18.8 | 0.003 | 0.922 |
| | $f_{seq}^{rec}$ | **†0.132** | 0.416 | 0.301 | 0.130 | 0.407 | 0.290 | 25.8 | 0.003 | 0.931 |
| | $f_{seq}^{lcs}$ | 0.128 | 0.410 | 0.331 | 0.126 | 0.401 | 0.308 | 10.5 | 0.013 | 0.943 |
| | $f_{seq}^{stree}$ | 0.130 | 0.413 | 0.326 | 0.127 | 0.404 | 0.302 | 19.4 | 0.010 | 0.934 |
| | $f_{seq}^{oracle}$ | **▲0.141** | **▲0.456** | 0.297 | **▲0.141** | **▲0.449** | 0.287 | 28.8 | 0.008 | 0.931 |

**Table A.10:** Performance of the rerankers for the Tour family of recommenders. Same notation as in Table 6.7.

| | | | Accuracy | | | Seq. Accuracy | | | Non-Accuracy | |
|---|---|---|---|---|---|---|---|---|---|---|
| City | Reranker | P | nDCG | TFP | $P_s$ | $nDCG_s$ | $FP_s$ | Dist | Gini | EPC |
| New York | Baseline | 0.132 | 0.404 | 0.295 | 0.129 | 0.391 | 0.279 | 44.9 | 0.001 | 0.911 |
| | $f_{seq}^{rnd}$ | 0.117 | 0.374 | 0.343 | 0.115 | 0.364 | 0.305 | 23.9 | **▴0.002** | 0.945 |
| | $f_{seq}^{dist}$ | 0.126 | 0.392 | **▴0.350** | 0.124 | 0.381 | 0.311 | **▴4.2** | 0.002 | **▴0.950** |
| | $f_{seq}^{feat}$ | 0.122 | 0.385 | 0.312 | 0.121 | 0.374 | 0.277 | 20.9 | 0.002 | 0.942 |
| | $f_{seq}^{item}$ | 0.137 | 0.410 | 0.316 | 0.134 | 0.397 | 0.283 | 38.1 | 0.002 | 0.912 |
| | $f_{seq}^{rec}$ | †**0.143** | †**0.417** | 0.316 | †**0.140** | †**0.403** | 0.289 | 41.4 | 0.001 | 0.911 |
| | $f_{seq}^{lcs}$ | 0.127 | 0.392 | 0.349 | 0.126 | 0.382 | **▴0.312** | 12.0 | 0.002 | 0.934 |
| | $f_{seq}^{stree}$ | 0.131 | 0.397 | 0.323 | 0.129 | 0.386 | 0.295 | 32.4 | 0.002 | 0.914 |
| | $f_{seq}^{oracle}$ | **▴0.149** | **▴0.452** | 0.301 | **▴0.149** | **▴0.442** | 0.285 | 44.4 | 0.002 | 0.910 |
| Tokyo | Baseline | 0.128 | †**0.388** | 0.366 | 0.127 | †**0.376** | 0.351 | 17.9 | 0.002 | 0.878 |
| | $f_{seq}^{rnd}$ | 0.115 | 0.365 | 0.380 | 0.115 | 0.355 | 0.359 | 25.7 | 0.002 | 0.924 |
| | $f_{seq}^{dist}$ | 0.120 | 0.374 | **▴0.385** | 0.119 | 0.364 | **▴0.364** | **▴6.7** | 0.002 | **▴0.927** |
| | $f_{seq}^{feat}$ | 0.126 | 0.384 | 0.319 | 0.124 | 0.372 | 0.318 | 26.1 | 0.001 | 0.869 |
| | $f_{seq}^{item}$ | 0.127 | 0.386 | 0.369 | 0.126 | 0.375 | 0.352 | 17.0 | **▴0.002** | 0.882 |
| | $f_{seq}^{rec}$ | †**0.129** | 0.387 | 0.353 | †**0.127** | 0.375 | 0.342 | 24.7 | 0.001 | 0.868 |
| | $f_{seq}^{lcs}$ | 0.123 | 0.376 | 0.353 | 0.121 | 0.365 | 0.343 | 11.4 | 0.002 | 0.896 |
| | $f_{seq}^{stree}$ | 0.126 | 0.381 | 0.365 | 0.125 | 0.370 | 0.353 | 13.8 | 0.002 | 0.884 |
| | $f_{seq}^{oracle}$ | **▴0.131** | **▴0.406** | 0.367 | **▴0.131** | **▴0.395** | 0.352 | 18.5 | 0.002 | 0.878 |
| Rome | Baseline | 0.231 | †**0.537** | 0.519 | 0.212 | †**0.477** | 0.473 | 2.0 | 0.076 | 0.871 |
| | $f_{seq}^{rnd}$ | 0.182 | 0.445 | 0.507 | 0.168 | 0.400 | 0.448 | 5.7 | **▴0.100** | **▴0.896** |
| | $f_{seq}^{dist}$ | 0.221 | 0.519 | †**0.521** | 0.208 | 0.467 | †**0.474** | **▴1.4** | 0.097 | 0.892 |
| | $f_{seq}^{feat}$ | 0.193 | 0.470 | 0.402 | 0.178 | 0.420 | 0.359 | 5.0 | 0.063 | 0.888 |
| | $f_{seq}^{item}$ | †**0.232** | 0.536 | 0.521 | †**0.213** | 0.477 | 0.470 | 1.8 | 0.081 | 0.875 |
| | $f_{seq}^{rec}$ | 0.213 | 0.485 | 0.509 | 0.190 | 0.427 | 0.451 | 5.8 | 0.058 | 0.856 |
| | $f_{seq}^{lcs}$ | 0.198 | 0.487 | 0.487 | 0.186 | 0.438 | 0.446 | 2.1 | 0.097 | 0.894 |
| | $f_{seq}^{stree}$ | 0.215 | 0.510 | 0.515 | 0.201 | 0.457 | 0.466 | 2.4 | 0.090 | 0.882 |
| | $f_{seq}^{oracle}$ | **▴0.279** | **▴0.644** | **▴0.529** | **▴0.279** | **▴0.600** | **▴0.482** | 3.0 | 0.076 | 0.871 |
| Petaling Jaya | Baseline | 0.127 | 0.412 | 0.244 | 0.125 | 0.403 | 0.240 | 28.4 | 0.002 | 0.918 |
| | $f_{seq}^{rnd}$ | 0.123 | 0.399 | 0.306 | 0.120 | 0.390 | 0.291 | 30.8 | **▴0.002** | **▴0.941** |
| | $f_{seq}^{dist}$ | †**0.134** | †**0.422** | **▴0.325** | †**0.132** | †**0.412** | **▴0.309** | **▴7.1** | 0.002 | 0.938 |
| | $f_{seq}^{feat}$ | 0.128 | 0.411 | 0.312 | 0.125 | 0.401 | 0.278 | 31.6 | 0.001 | 0.921 |
| | $f_{seq}^{item}$ | 0.129 | 0.416 | 0.277 | 0.126 | 0.406 | 0.271 | 16.9 | 0.002 | 0.923 |
| | $f_{seq}^{rec}$ | 0.132 | 0.416 | 0.280 | 0.130 | 0.408 | 0.273 | 26.6 | 0.002 | 0.923 |
| | $f_{seq}^{lcs}$ | 0.132 | 0.417 | 0.315 | 0.129 | 0.406 | 0.290 | 13.9 | 0.002 | 0.929 |
| | $f_{seq}^{stree}$ | 0.128 | 0.412 | 0.291 | 0.126 | 0.403 | 0.272 | 21.5 | 0.002 | 0.921 |
| | $f_{seq}^{oracle}$ | **▴0.143** | **▴0.462** | 0.250 | **▴0.143** | **▴0.455** | 0.244 | 28.0 | 0.002 | 0.918 |

**Table A.11:** Performance for all the cities when items already seen by the user are also allowed in the recommendations. Notation as in Table A.5.

| | | | | Accuracy | | | Seq. Accuracy | | | Non Accuracy | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| City | Family | Rec | P | nDCG | TFP | $P_s$ | $nDCG_s$ | $FP_s$ | Dist | Gini | EPC |
| NYC | Basic | Pop | 0.152 | 0.431 | 0.326 | 0.146 | 0.414 | 0.286 | 47.6 | 0.001 | 0.896 |
| | Classic | UB | 0.155 | 0.433 | †**0.354** | 0.148 | 0.416 | †**0.320** | †**27.6** | †**0.011** | †**0.939** |
| | Temporal | BFsUB | 0.161 | 0.444 | 0.344 | 0.153 | 0.425 | 0.314 | 39.4 | 0.004 | 0.918 |
| | Geo | IRenMF | †**0.163** | †**0.445** | 0.353 | †**0.155** | †**0.427** | 0.319 | 37.0 | 0.003 | 0.918 |
| | Tour | ItemMC | 0.143 | 0.422 | 0.291 | 0.138 | 0.406 | 0.281 | 48.3 | 0.001 | 0.902 |
| | Skylines | TestOrder | ▴**0.497** | ▴**0.990** | ▴**0.497** | ▴**0.497** | ▴**0.970** | ▴**0.497** | ▴**9.3** | ▴**0.024** | ▴**0.976** |
| TOK | Basic | Train | 0.166 | 0.448 | †**0.403** | 0.157 | 0.424 | †**0.376** | 22.2 | †**0.016** | †**0.940** |
| | Classic | UB | 0.162 | 0.439 | 0.371 | 0.155 | 0.420 | 0.359 | 23.2 | 0.003 | 0.864 |
| | Temporal | BFUB | 0.164 | 0.442 | 0.373 | 0.158 | 0.422 | 0.359 | 21.1 | 0.002 | 0.858 |
| | Geo | IRenMF | †**0.172** | †**0.449** | 0.376 | †**0.162** | †**0.426** | 0.364 | 21.7 | 0.002 | 0.855 |
| | Tour | ItemMC | 0.156 | 0.429 | 0.386 | 0.151 | 0.413 | 0.375 | †**12.4** | 0.002 | 0.846 |
| | Skylines | TestOrder | ▴**0.518** | ▴**0.994** | ▴**0.518** | ▴**0.518** | ▴**0.970** | ▴**0.518** | ▴**9.1** | ▴**0.020** | ▴**0.955** |
| ROM | Basic | Pop | 0.233 | 0.514 | 0.532 | 0.204 | 0.450 | 0.477 | 4.0 | 0.035 | 0.819 |
| | Classic | BPRMF | 0.235 | 0.516 | †**0.534** | 0.206 | 0.452 | 0.467 | 4.7 | 0.035 | 0.821 |
| | Temporal | Fossil | 0.232 | 0.515 | 0.526 | 0.206 | 0.453 | 0.442 | 5.1 | 0.036 | 0.829 |
| | Geo | RankGeoFM | 0.210 | 0.482 | 0.513 | 0.190 | 0.429 | 0.458 | 5.3 | 0.068 | 0.841 |
| | Tour | ItemMC | †**0.255** | †**0.567** | 0.533 | †**0.234** | †**0.501** | †**0.483** | ▴**1.3** | †**0.082** | †**0.886** |
| | Skylines | TestOrder | ▴**0.546** | ▴**1.000** | ▴**0.546** | ▴**0.546** | ▴**0.936** | ▴**0.546** | 2.3 | ▴**0.202** | ▴**0.907** |
| PJ | Basic | Train | †**0.160** | †**0.473** | 0.355 | 0.151 | †**0.454** | 0.325 | 21.5 | †**0.029** | †**0.972** |
| | Classic | UB | 0.158 | 0.468 | 0.330 | 0.150 | 0.449 | 0.313 | 21.5 | 0.006 | 0.939 |
| | Temporal | BFsUB | †**0.160** | 0.464 | 0.288 | †**0.154** | 0.450 | 0.276 | 22.6 | 0.003 | 0.928 |
| | Geo | IRenMF | 0.155 | 0.457 | †**0.367** | 0.149 | 0.444 | 0.343 | 22.0 | 0.010 | 0.964 |
| | Tour | ItemMC | 0.143 | 0.443 | 0.356 | 0.141 | 0.433 | †**0.345** | †**11.0** | 0.002 | 0.942 |
| | Skylines | TestOrder | ▴**0.436** | ▴**0.967** | ▴**0.434** | ▴**0.436** | ▴**0.954** | ▴**0.434** | ▴**8.5** | ▴**0.025** | ▴**0.977** |

by balancing a tradeoff between relevance and the rest of the dimensions.

Regarding specific rerankers, the distance-based reranker ($f^{dist}_{seq}$) is, by definition, the one that reduces the most the distance of the recommended route, but what is more important is that, in some situations, it is able to improve the performance in both $nDCG_s$ and $FP_s$ (see for example the results in Petaling Jaya for most of the families, but especially for Tour) or at least in one of the metrics (for example, in New York this reranker is the best in terms of $FP_s$). The performance of the rerankers based on subsequences ($f^{lcs}_{seq}$ and $f^{stree}_{seq}$) actually depends on each dataset, although it seems they work better in Petaling Jaya; nonetheless, they tend to decrease the distance and improve $FP_s$, but these results also depend on the recommender family, so they are not conclusive except in Petaling Jaya, where these strategies work better than the baseline in every case.

Now, if we analyze these results from the perspective of the family of the recommenders, we observe that the behavior is pretty stable in each city, independently of the origin of the recommendations being reranked; however, we observe how the oracle reranker obtains different values depending on the family, which tends to be lower for those in the Tour family except in Rome, and higher in the rest, evidencing that the recommended items being reranked (those in the top-n) have a lot of potential (except, to a lower extent, in the Tour family) and there is room for improvement.

### A.2.3 Extended analysis on repeated interactions

In this section, we present additional results where recommenders are allowed to return items previously interacted by the user, a situation very common in the venue recommendation task. In the previous results in Chapter 6 (either those shown in the main part or in the appendix), we have not processed in any way the routes included in the test sets, hence, they may include repeated items or items previously visited by the user; we decided not to do anything with these cases to not "break" the sequentiality inherent in the routes followed by the users. However, as we show in Table A.11, by allowing the algorithms to recommend items the user has previously interacted with – we denote this methodology as "ItemsInTraining" (already shown in Section 4.5.2), since every candidate item needs to appear in the training set with no further restrictions –, the behavior of the recommenders is markedly different, in particular for those in the Basic, Classic, and Temporal families.

More specifically, in Petaling Jaya and Tokyo, the best recommender from the Basic family is the one returning just the training set of the user (Train); moreover, in Petaling Jaya, it is actually the best recommender after the skylines. Thus, this is a strong baseline to beat, where some of the more complex algorithms such as UB, Caser, or IRenMF obtain worse performance values or very close to the ones from this method. Furthermore, in this scenario, the popularity bias found in the TrainItems methodology and well-known in classical recommendation (Bellogín et al., 2017) is strongly reduced, favoring another type of baseline, evidencing that in this domain, well-known, popular venues are not as important as previously visited venues by each user, confirming that these two scenarios (TrainItems against ItemsInTraining) are actually modeling two different recommendation situations and hypotheses. This change of behavior could be the reason for the change in the optimal recommenders of the other families, instead of BPRMF in Tokyo, UB is the best algorithm in the Classic family, as in Petaling Jaya.

At the end, we argue that, by evaluating with items already interacted by the user we are aiming at a different kind of algorithm than when those items are removed; in other terms, a recommender system that performs very well with known items (ItemsInTraining) is expected to distinguish well which of the previously visited venues the user will visit next, hence, its final goal is to generate recommendations already known by the user, probably the opposite of a recommender evaluated with only new items in the test set (TrainItems), thus aiming at recommending new, novel venues for each particular user – in fact, some authors define explicitly such a task as *recommending new places* (Bothorel et al., 2018).

## A.3 Information about N-MCA and C-MCA cities

In this section we include the information of the selected cities in Chapter 7 according to N-MCA and C-MCA strategies (see Section 7.4.4). Here we include the 7 closest cities (N-MCA strategy) with respect to each target city:

- Istanbul: Kutahya, Bursa, Eskisehir, Tekirdag, Kocaeli, Balikesir, Sakarya.

- Jakarta: Palembang, Tanjungkarang-Telukbetung, Pontianak, Bandung, Surabaja, Semarang, Yogyakarta.
- Kuala Lumpur: Ipoh, Seremban, Pinang, Kuantan New Port, Shah Alam, Kuala Terengganu, Melaka.
- Mexico City: Queretaro, Jalapa, Morelia, Puebla, Pachuca, Toluca, Cuernavaca.
- Moscow: Tver, Yaroslavl, Ivanovo, Gor'kiy, Voronezh, Ceboksary, Smolensk.
- Santiago: Valparaiso, Coquimbo, Talca, Cordoba, Concepcion, Temuco, La Serena.
- São Paulo: Florianopolis, Curitiba, Santos, Vitoria, Belo Horizonte, Niteroi, Rio de Janeiro.
- Tokyo: Sendai, Kawasaki, Osaka, Gifu, Yokohama, Kyoto, Nagoya.

To properly compare the results from N-MCA and C-MCA, we also state all cities belonging to the same country (C-MCA) with respect to each target city:

- Turkey: Istanbul, Sakarya, Balikesir, Canakkale, Zonguldak, Kutahya, Kayseri, Ordu, Trabzon, Mersin, Isparta, Mugla, Denizli, Sanhurfa, Aydin, Ankara, Eskisehir, Malatya, Kocaeli, Seyhan, Tekirdag, Afyon, Samsun, Rize, Izmir, Bursa, Antalya, Giresun, Antioch, Manisa, Kahramanmaras, Bolu, Edirne, Konya, Aintab.
- Indonesia: Jakarta, Palembang, Tanjungkarang-Telukbetung, Semarang, Samarinda, Balikpapan, Surabaja, Bandung, Denpasar, Bandjermasin, Mataram, Yogyakarta, Padang, Pontianak, Medan, Manado, BandaAceh, Pekanbaru.
- Myanmar: Kuala Lumpur Ipoh, Alor Setar, Melaka, Kangar, Kuantan NewPort, Pinang, Kota Baharu, Kuala Terengganu, Kota Kinabalu, Kuching, Johor Baharu, Seremban, ShahAlam.
- Mexico: Mexico City, Villahermosa, Queretaro, Tampico, Jalapa, Morelia, Puebla, Pachuca, Toluca, Cuernavaca, Guadalajara, Aguascalientes, La Paz, Oaxaca, Tuxtla Gutierrez, San Luis Potosi, Campeche, Colima, Veracruz, Merida, Monterrey, Hermosillo.
- Russia: Moscow, Irkutsk, Kuybyskev, Kaliningrad, Ivanovo, Voronezh, Vladivostok, Gor'kiy, Chelyabinsk, Rostov-on-Don, Omsk, Krasnodar, Perm, Novosibirsk, Vyatka, Saint Petersburg, Tver, Ufa, Tomsk, Smolensk, Sverdlovsk, Krasnoyarsk, Volgograd, Kazan, Izevsk, Ceboksary, Ulyanovsk, Yakutsk, Khabarovsk, Yaroslavl, Saratov.
- Chile: Santiago, Puerto Montt, Valparaiso, Coquimbo, Talca, Antofagasta, Concepcion, Temuco, La Serena, Iquique.

- Brazil: São Paulo, Joao Pessoa, Porto Velho, Natal, Palmas, Belem, Manaus, Maceio, Aracaju, Boa Vista, Vitoria, Niteroi, Brasilia, Belo Horizonte, Cuiaba, Sao Luis, Macapa, Curitiba, Rio de Janeiro, Rio Branco, Goiania, Florianopolis, Teresina, Fortaleza, Santos, Campo Grande, Recife, Porto Alegre, Santarem, Salvador.

- Japan: Tokyo, Hiroshima, Naha, Fukuoka, Kobe, Kawasaki, Sendai, Kawasaki, Osaka, Gifu, Yokohama, Kyoto, Nagoya, Shimonoseki, Sapporo.

# Appendix B

# Introducción

## B.1 Motivación

En 2018, la empresa DOMO estimó que durante el año 2020 cada persona del planeta Tierra generaría una media de 1,7MB de datos cada segundo (Ahmad, 2018). En la actualidad, es muy probable que esta cantidad de datos se haya incrementado sustancialmente debido a que la crisis sanitaria de la COVID-19 ha obligado a un gran número de personas a pasar más tiempo en casa y a hacer un mayor uso de Internet, tanto generando como consumiendo contenidos ya sea por motivos de ocio o de trabajo.

En este contexto de crecimiento exponencial de la información disponible en la web, se hace aún más evidente el problema de la sobrecarga de información (Maes, 1994), en el que los usuarios pueden dedicar un tiempo excesivo a la búsqueda de la información que necesitan. Para solucionar este problema surgen los Sistemas de Recomendación (RS, en inglés). Estas herramientas de software están orientadas a filtrar los innumerables artículos disponibles en un sistema para recomendar a los usuarios aquellos elementos que se adaptan mejor a sus necesidades en función de sus experiencias previas. Aunque los primeros buscadores y Sistemas de Recomendación surgieron en los años 90, su uso se ha extendido de forma imparable en los últimos años.

Hoy en día, empresas como Google, Amazon, Youtube, Netflix, y muchas más hacen uso de estas tecnologías para aumentar el número de usuarios de sus plataformas a la vez que ofrecen contenidos personalizados a los clientes existentes. En concreto, los Sistemas de Recomendación han demostrado ser un área de investigación muy demandada, especialmente desde la aparición del premio Netflix entre 2006-2009 (Bell and Koren, 2007), donde se ofrecía 1M de dólares al grupo de investigación que consiguiera mejorar la predicción de su algoritmo base en un 10%. Al mismo tiempo, las conferencias internacionales dedicadas a este tema (entre las que destaca la conferencia ACM on Recommender Systems[1]) aumentan cada año el número de asistentes, así como las empresas interesadas en patrocinar estos congresos. Sin embargo, los Sistemas de Recomendación no son perfectos ni mucho menos. Debido al análisis masivo de datos que realizan estos algoritmos, los usuarios están cada vez más sensibilizados con aspectos

[1]ACM Conference on Recommender Systems, RecSys, https://recsys.acm.org/

como la privacidad, el intrusismo o la explicación de las recomendaciones (Ricci et al., 2015). Algunos de estos problemas son, de hecho, retos importantes dentro de la comunidad de los Sistemas de Recomendación, aunque están fuera del alcance de esta tesis.

El alto grado de adaptabilidad de los Sistemas de Recomendación permite que se apliquen en muchos ámbitos diferentes como el cine, los libros, la música, el turismo, o incluso en las aplicaciones de citas (Ricci et al., 2015). Sin embargo, es importante mencionar que cada dominio tiene sus propias particularidades. Por ejemplo, la música y el cine, que son dominios de recomendación muy conocidos, tienen diferencias sustanciales: el catálogo de películas es normalmente más reducido que el de canciones, el dominio de la música tiene un fuerte componente secuencial, mientras que fallar una recomendación en el dominio de la música no es demasiado crítico, ya que las canciones suelen durar menos de 5 minutos, fallar una recomendación de películas o vídeos puede afectar más a los usuarios, ya que tienden a frustrarse más fácilmente, etc. (Schedl et al., 2018, Jannach et al., 2018). Además, hay dominios que necesitan incorporar fuentes de información adicionales para hacer recomendaciones útiles. En este sentido, asistentes como Alexa (Amazon), Google Assistant o Siri (Apple) podrían ser especialmente útiles ya que almacenan más información sobre los usuarios como sus gustos, su ubicación actual, la hora o incluso el tiempo, entre otros datos.

Un ejemplo importante en el que se puede aplicar todo este tipo de información es el ámbito del turismo, que tiene un gran impacto económico tanto en los turistas como en las regiones que visitan. Por ejemplo, en países como España, Islandia o México, el porcentaje del Producto Interior Bruto (PIB) total asociado al turismo es superior al 8%[2]. Hay un gran número de tareas de recomendación relacionadas con el turismo, incluyendo la recomendación de rutas o trayectorias y de grupos, pero quizás la más conocida y estudiada sea el problema de recomendación de Puntos de Interés (POI, del inglés *Points-of-Interest*), donde los elementos a recomendar son locales o lugares interesantes para que el usuario visite cuando llegue a una ciudad (hoteles, bares, restaurantes, museos, etc.) (Ye et al., 2011). En este tipo de recomendación, las Redes Sociales basadas en localización (LBSNs, del inglés *Location-Based Social Networks*) como Foursquare, Yelp, o Gowalla (ver Figura B.1), son especialmente relevantes ya que en estas redes sociales los usuarios pueden registrar los check-ins que realizan sobre los locales que visitan e intercambiar información con el resto de usuarios del sistema (Wang et al., 2013). De hecho, la investigación en este campo se ha incrementado en los últimos años debido a varias razones, entre ellas el creciente número de personas que pueden permitirse hacer viajes a diferentes ciudades, la mejora en las infraestructuras de transporte o la facilidad para acceder a tecnología de alto nivel (por ejemplo, redes de alta velocidad o teléfonos móviles más avanzados).

Al principio, las primeras estrategias de recomendación sólo utilizaban las interacciones entre los usuarios y los artículos para hacer recomendaciones. Sin embargo, en los últimos años, el uso de información contextual se ha vuelto especialmente útil, ya

[2]Organización para la Cooperación y el Desarrollo Económico, OCDE, `https://www.oecd.org/cfe/tourism/OECD-Tourism-Trends-Policies2020-Highlights-ENG.pdf`

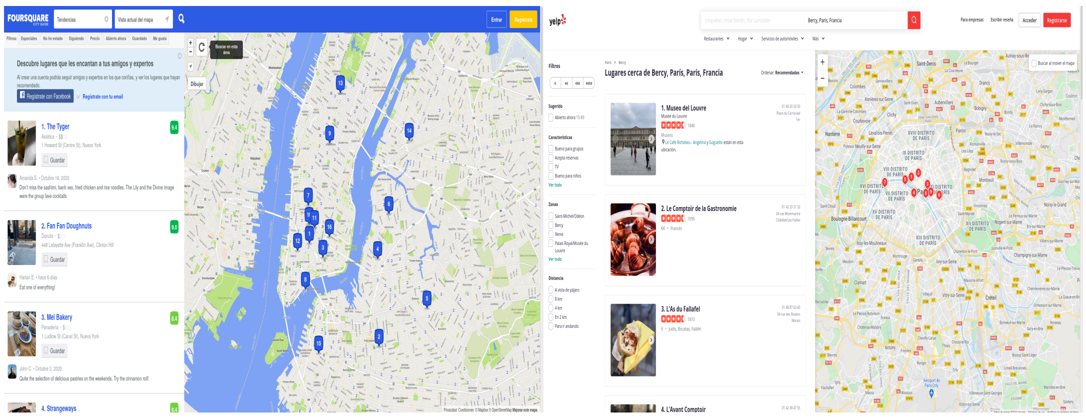

FOURSQUARE
Descubre lugares que les encantan a tus amigos y expertos
1. The Tyger
2. Fan Fan Doughnuts
3. Mei Bakery
4. Strangeways
yelp
Lugares cerca de Bercy, Paris, Paris, Francia
1. Museo del Louvre
2. Le Comptoir de la Gastronomie
3. L'As du Fallafel
4. L'Avant Comptoir

Figure B.1: Ejemplo de dos LBSNs. Foursquare (izquierda) y Yelp (derecha).

que permite a los algoritmos adaptarse mejor a los intereses de los usuarios en determinadas situaciones. Esta información contextual puede ser muy variada, incluyendo información temporal y/o secuencial, meteorológica, tendencias populares, etc. (Adomavicius and Tuzhilin, 2015, Villegas et al., 2018). Es importante tener en cuenta que los usuarios pueden consumir determinados elementos en función de la situación actual. Por ejemplo, un usuario puede ver diferentes tipos de películas dependiendo de si está solo o acompañado. El mismo razonamiento se aplica en el ámbito de la música, ya que un usuario puede escuchar diferentes canciones dependiendo de si está en el trabajo o con amigos un viernes por la noche. En el ámbito del turismo, esta información contextual también es importante, ya que los tipos de POIs que visita un usuario pueden verse afectados por el clima (por ejemplo, en verano es más factible realizar actividades al aire libre), la hora del día (por ejemplo, no recomendar un bar a primera hora de la mañana) o la distancia geográfica (Laß et al., 2017).

Por otra parte, si bien es importante investigar cómo generar mejores recomendaciones, también es esencial diseñar mecanismos para evaluar los modelos. De hecho, la evaluación de los Sistemas de Recomendación ha cambiado a lo largo de los años. Las métricas de predicción de error orientadas a medir la diferencia entre las valoraciones predichas y las reales se utilizaron inicialmente para analizar el rendimiento de los recomendadores pero, en los últimos años, la evaluación de los algoritmos ha evolucionado mediante el uso de métricas de ranking procedentes del área de la Recuperación de Información (IR, del inglés *Information Retrieval*), donde el objetivo es predecir una lista de elementos hipotéticamente interesantes para el usuario (Steck, 2013, Gunawardana and Shani, 2015). Sin embargo, este tipo de análisis, aunque útil, es incompleto, ya que es importante desde el punto de vista del usuario medir también la calidad de las recomendaciones en términos de otras dimensiones complementarias como la novedad, la diversidad o la serendipia (Castells et al., 2015). Además, en los últimos tiempos ha aumentado la concienciación sobre la necesidad de ofrecer recomendaciones justas a los usuarios, evitando posibles sesgos contraproducentes en los algoritmos, como por ejemplo realizar recomendaciones de mayor calidad a grupos sociales específicos por su edad, género, nacionalidad, etc (Steck, 2018, Abdollahpouri et al., 2019a). Por estas razones, uno de los objetivos de la comunidad es desarrollar algoritmos con un buen equilibrio entre estas dimensiones y el acierto – tratando de obtener técnicas precisas y a la vez lo más justas posible –, aunque generalmente se asume que es difícil, o incluso imposible, desarrollar un algoritmo que pueda superar a cualquier otra técnica en todos los aspectos posibles. Además, actualmente existe una creciente preocupación por la reproducibilidad del rendimiento obtenido por los modelos, ya que a menudo no es posible replicar los resultados comunicados por los autores originales de los trabajos (Beel et al., 2016, Dacrema et al., 2019). En consecuencia, desde hace algunos años, se ha intentado promover la publicación del código de los modelos desarrollados y, al mismo tiempo, se ha prestado más atención a diferentes aspectos del proceso de evaluación.

Así, en esta tesis analizamos algunas de las cuestiones mencionadas proponiendo soluciones en forma de algoritmos y métricas para profundizar en el uso de la información contextual en las recomendaciones. Aunque las propuestas desarrolladas pueden ser utilizadas en varios dominios de recomendación, haremos especial hincapié en la recomendación de POIs, ya que es un área en crecimiento que puede beneficiarse del uso de este tipo de información para paliar algunos de sus problemas fundamentales (como la gran escasez de datos). En nuestro trabajo experimental demostramos la utilidad de nuestros enfoques propuestos ya que, por un lado, nuestras métricas derivadas nos permiten obtener un análisis más completo de por qué los algoritmos hacen determinadas recomendaciones, y por otro lado, nuestros modelos de recomendación contextual nos permiten mejorar el rendimiento de los algoritmos tanto en precisión como en dimensiones complementarias como la novedad, la diversidad o la frescura.

## B.2 Objetivos

Esta tesis tiene dos objetivos principales. Por un lado, afirmamos que el modelado de la información contextual es importante para mejorar el rendimiento de los algoritmos y, por lo tanto, investigamos cómo incorporar contextos como la secuencialidad o el tiempo en los Sistemas de Recomendación clásicos. Además, argumentamos que estos contextos también pueden integrarse en la fase de evaluación mediante la creación de nuevas métricas para detectar posibles sesgos en las recomendaciones producidas. Por otra parte, nos centramos en el problema de la recomendación de Puntos de Interés, estudiando los principales problemas y retos del área y proponiendo soluciones para paliarlos. En este sentido, demostramos que podemos generar rutas coherentes para los usuarios explotando los enfoques de reranking, mientras que también podemos utilizar técnicas de dominio cruzado (del inglés *cross-domain*) para mejorar el rendimiento de los recomendadores. Por tanto, tomando estas ideas como punto de partida, en esta tesis proponemos los siguientes objetivos de investigación (OBJ):

**OBJ1: Revisar el estado del arte sobre Sistemas de Recomendación de Puntos de Interés con la intención de caracterizar los trabajos más importantes en el área.** La mayoría de las revisiones bibliográficas (o *surveys*, en inglés) en el ámbito de POIs se centran en analizar el tipo de algoritmo que se utiliza en los modelos y el tipo de información que explotan. Sin embargo, un aspecto importante que se suele pasar por alto es la metodología de evaluación y/o los conjuntos de datos y métricas utilizados. Por lo tanto, pretendemos realizar una *survey* para analizar todos estos aspectos en profundidad con el fin de detectar hasta qué punto son comparables los algoritmos orientados o enfocados a POI del estado del arte.

**OBJ2: Estudiar las métricas de evaluación de la recomendación clásica para adaptar e integrar dimensiones adicionales más allá de la relevancia.** Cuando se analiza el rendimiento de un recomendador, la mayoría de los investigadores analizan lo bien que funciona el sistema de recomendación en términos de precisión. Aunque se han tenido en cuenta dimensiones adicionales como la novedad y la diversidad a la hora de evaluar los recomendadores, creemos que es importante incorporar a las métricas otras dimensiones como la secuencialidad, el tiempo y la anti-relevancia (elementos que los usuarios han indicado específicamente que no les gustan). Por ello, tenemos previsto desarrollar diferentes métricas que tengan en cuenta estas dimensiones adicionales y analizar los resultados de los algoritmos en estas nuevas métricas.

**OBJ3: Desarrollar un mecanismo para añadir la secuencialidad en los sistemas de recomendación basados en vecinos próximos.** Actualmente, en el ámbito de los Sistemas de Recomendación, se han propuesto un gran número de modelos para incorporar contextos como la secuencialidad o el tiempo en las recomendaciones producidas. Hoy en día se utilizan métodos populares, como las redes neuronales y diferentes derivaciones de las cadenas de Markov, aunque en algunos casos resulta difícil interpretar correctamente las recomendaciones que proporcionan. Por lo tanto, investigamos la viabilidad de incorporar el contexto secuencial en los algoritmos basados en los vecinos próximos, ya que son más fáciles de entender e interpretar que los modelos mencionados.

**OBJ4: Explorar los datos de las LBSNs utilizados en la recomendación de POI para explorar nuevas formas de hacer recomendaciones y permitir la recomendación de rutas completas.** La recomendación de POIs se ha modelado tradicionalmente como una recomendación de lugares al usuario que son relevantes para él, sin tener en cuenta ningún tipo de relación secuencial entre ellos (por ejemplo, si en realidad están siguiendo algún tipo de ruta o trayectoria). En este sentido, pretendemos explotar la información almacenada en las LBSNs e investigar cómo obtener rutas a partir de los datos del usuario utilizando técnicas sencillas ya exploradas en el ámbito de los Sistemas de Recomendación.

**OBJ5: Mejorar el rendimiento de los algoritmos en la recomendación de POIs.** El problema de la recomendación de POIs tiene consideraciones específicas que deben ser tenidas en cuenta a la hora de realizar recomendaciones. El estudio que realizaremos en el OBJ1 nos permitirá establecer claramente estas consideraciones para poder proponer nuevos mecanismos que mejoren las recomendaciones producidas.

## B.3 Contribuciones

El trabajo realizado en esta tesis ha contribuido al actual estado del arte tanto de la recomendación clásica como de la recomendación de POIs. Las contribuciones incluyen: una categorización sistemática de los diferentes tipos de algoritmos de recomendación de Puntos de Interés, la definición de nuevas métricas para analizar el rendimiento de los modelos de recomendación en varias dimensiones, la definición de nuevos algoritmos para la recomendación tradicional, y el análisis de las rutas y los sesgos en los datos obtenidos en las Redes Sociales basadas en localización.

En primer lugar, en el **Capítulo** 3 nos centramos en el dominio de la recomendación de POIs y realizamos una revisión en la que analizamos los algoritmos más importantes del estado del arte entre 2011 y 2019. En esta revisión, además de analizar el tipo de información (geográfica, de contenido, colaborativa, etc.) y los algoritmos (sociales, de aprendizaje profundo, de factorización, etc.) utilizados en este ámbito, también examinamos los procedimientos más habituales a la hora de evaluar las propuestas. En este aspecto, determinamos los conjuntos de datos más utilizados, así como las métricas y los tipos de particionamiento de datos, con el fin de determinar cómo de comparables son todos estos trabajos entre sí. Nuestro análisis sobre los enfoques actuales de recomendación de POIs se ha presentado en la siguiente revista:

- **Pablo Sánchez** and Alejandro Bellogín. (2020). Point-of-Interest Recommender Systems: A Survey from an Experimental Perspective. Submitted to *ACM Computing Surveys*. Under Review (1st round of review). **Factor de Impacto 2019:** 7.990. **JCR 2019:** Q1:4/108. Computer Science Theory & Methods.

En segundo lugar, en el **Capítulo** 4 proponemos nuevas métricas que incorporan diferentes contextos para evaluar los recomendadores. Primero, explotamos la información temporal para ver si un elemento es novedoso o no en función de los momentos concretos en los que fue consumido por los usuarios del sistema. Después, adaptamos el Principio de Ranking Probabilístico (del inglés *Probabilistic Ranking Principle*) para definir nuevas métricas que midan cuántas recomendaciones "anti-relevantes" (ítems con una valoración muy baja) hacen los algoritmos. Se trata de una propuesta novedosa porque, aunque es importante que un recomendador haga buenas recomendaciones, es igualmente importante que no sugiera artículos que los usuarios hayan categorizado específicamente como "malos". A continuación, también definimos métricas que explotan los atributos de los usuarios y los artículos. Los atributos de los artículos nos permiten determinar si una lista de recomendaciones es mejor que otra en función de cómo coinciden los atributos de los artículos recomendados con los del conjunto de pruebas. Por otro lado, los atributos de los usuarios nos permiten determinar si los recomendadores proporcionan recomendaciones de la misma calidad a diferentes grupos de usuarios. Por último, también modificamos las métricas de precisión tradicionales basadas en la clasificación de la recuperación de información para tener en cuenta no sólo la relevancia de las recomendaciones, sino también el orden con respecto al conjunto de test del usuario, para comprobar si los elementos recomendados siguen una secuencia determinada. Las publicaciones relacionadas con este capítulo son las siguientes:

- **Pablo Sánchez** and Alejandro Bellogín. Time-aware novelty metrics for recommender systems. In Gabriella Pasi, Benjamin Piwowarski, Leif Azzopardi, and Allan Hanbury, editors, *Advances in Information Retrieval - 40th European Conference on IR Research*, ECIR 2018, Grenoble, France, March 26-29, 2018, Proceedings, volume 10772 of Lecture Notes in Computer Science, pages 357-370. Springer, 2018. **CORE 2018:** A. **Tasa de aceptación (long papers):** 23%

- **Pablo Sánchez** and Alejandro Bellogín. Measuring anti-relevance: a study on when recommendation algorithms produce bad suggestions. In Sole Pera, Michael D. Ekstrand, Xavier Amatriain, and John O'Donovan, editors, *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018, pages 367-371. ACM, 2018. **CORE 2018:** B. **Tasa de aceptación (short papers):** 25%.

- **Pablo Sánchez** and Alejandro Bellogín. Attribute-based evaluation for recommender systems: incorporating user and item attributes in evaluation metrics. In Toine Bogers, Alan Said, Peter Brusilovsky, and Domonkos Tikk, editors, *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys 2017, Copenhagen, Denmark, September 16-20, 2019., pages 378-382. ACM, 2019. **CORE 2018:** B. **Tasa de aceptación (short papers):** 24%.

Después de eso, en el **Capítulo** 5 adaptamos el algoritmo de la subsecuencia común más larga (del inglés, *Longest Common Subsequence*) para medir la similitud entre los usuarios de un sistema e integrar esta similitud en un modelo de recomendación. Como este algoritmo está diseñado para trabajar con secuencias, nos permite incorporar tanto información temporal como de contenido de forma sencilla y también es adecuado para trabajar con interacciones repetidas (es decir, cuando el usuario consume el mismo elemento más de una vez). Esto es algo útil en dominios como la música o la recomendación de POIs, donde los usuarios tienden a consumir o visitar el mismo elemento más de una vez. Además, hemos propuesto un nuevo algoritmo de recomendación basado en vecinos en el que los ítems candidatos se seleccionan utilizando la última interacción que el vecino tiene con el usuario objetivo, con el fin de generar recomendaciones que tengan en cuenta la información temporal y secuencial. La principal publicación relacionada con este capítulo es:

- **Pablo Sánchez** and Alejandro Bellogín. Time and sequence awareness in similarity metrics for recommendation. *Information Processing Management*, 57(3):102228, 2020. **Factor de Impacto 2019:** 4.787. **JCR 2019:** Q1: 22/156. Computer Science, Information Systems.

Este artículo está influenciado y puede entenderse como el trabajo futuro de las estas publicaciones anteriores:

- **Pablo Sánchez** and Alejandro Bellogín. Building user profiles based on sequences for content and collaborative filtering. *Information Processing Management*, 56(1):192-211, 2019. **JCR 2019:** Q1: 22/156. Computer Science, Information Systems.

- Alejandro Bellogín and **Pablo Sánchez**. Collaborative filtering based on subsequence matching: A new approach. *Information Sciences*, 418:432-446, 2017. **Factor de Impacto 2017:** 4.305. **JCR 2017:** Q1:12/148. Computer Science, Information Systems.

Continuando con el trabajo de las secuencias, en el **Capítulo** 6 definimos también mecanismos para obtener rutas o trayectorias de POIs, no sólo independientes, utilizando los datos de LBSNs. Así, definimos un método para obtener y filtrar rutas a partir de interacciones de POIs en estos conjuntos de datos y luego exploramos cómo recomendar rutas aplicando y adaptando técnicas de IR como reranking en el ámbito de la recomendación de POIs. La publicación relacionada con este capítulo es:

- **Pablo Sánchez** and Alejandro Bellogín. Applying reranking strategies to route recommendation using sequence-aware evaluation. *User Modeling and User-Adapted Interaction*, 30(4):659-725, 2020. **Impact Factor 2019:** 4.682. **JCR 2019:** Q1:4/22. Computer Science, Cybernetics.

Por último, debido a la gran dispersión de datos prevalente en el área de recomendación de POIs, en el **Capítulo** 7 también aplicaremos técnicas basadas en dominio cruzado para ver si es posible mejorar las recomendaciones producidas por los algoritmos tanto en términos de precisión como en otras dimensiones tales como novedad, diversidad y cobertura. Al mismo tiempo, clasificamos a los usuarios en dos grupos diferentes, turistas y locales, y analizamos el efecto de estas técnicas en ambos grupos por separado. Las publicaciones relacionadas con este capítulo son:

- **Pablo Sánchez** and Alejandro Bellogín. On the effects of aggregation strategies for different groups of users in venue recommendation. Submitted to *Information Processing and Management*. Under Review. **Factor de impacto 2019:** 4.787. **JCR 2019:** Q1: 22/156. Computer Science, Information Systems.

- **Pablo Sánchez** and Alejandro Bellogín. A novel approach for venue recommendation using cross-domain techniques. In the 2nd Workshop on Intelligent Recommender Systems by Knowledge Transfer & Learning (RecSysKTL), held in conjunction with the 12th ACM Conference on Recommender Systems, Vancouver, Canada, Oct, 2018.

## B.4 Estructura de la tesis

Esta tesis está estructurada de la siguiente manera:

- En el Capítulo 2 se presenta el estado del arte de los Sistemas de Recomendación clásicos. En primer lugar, definimos el problema de la recomendación y categorizamos los tipos más populares de Sistemas de Recomendación destacando sus ventajas, desventajas y los enfoques más representativos. Además, mostramos la evolución de los métodos y protocolos de evaluación que se han utilizado en el área y definimos las metodologías y métricas de evaluación más conocidas actualmente.

- En el Capítulo 3 se analiza el problema de la recomendación de POIs y se clasifican los enfoques más importantes en este ámbito entre 2011 y 2019. En nuestro estudio, clasificamos los modelos según el tipo de información utilizada (temporal, secuencial, de contenido, etc.), el tipo de algoritmo (similitudes, factorización, redes neuronales, etc.), y también hacemos especial hincapié en el aspecto de la reproducibilidad, mostrando las principales estrategias de evaluación utilizadas en el área (métricas, conjuntos de datos, tipo de división, etc.).

- En el Capítulo 4 se propone un conjunto de nuevas métricas para incorporar contextos adicionales a la hora de evaluar los Sistemas de Recomendación. En primer lugar, ampliamos un framework existente de novedad y diversidad para incorporar información temporal. En segundo lugar, mostramos cómo tener en cuenta en la evaluación los artículos que no gustan específicamente al usuario. Posteriormente, mostramos cómo podemos utilizar los atributos de los artículos en la evaluación y, por último, mostramos cómo incorporar la secuencialidad en las métricas de clasificación tradicionales.

- En el Capítulo 5 se propone una nueva métrica de similitud basada en el algoritmo de la Subsecuencia Común más Larga para ser utilizada en los recomendadores basados en la secuencialidad. Además, también se propone una reformulación de los algoritmos basados en vecinos aportando ideas de las técnicas de Fusión de Ranking.

- En el Capítulo 6 se siguen explorando los datos de las LBSNs y se proponen mecanismos para obtener rutas completas a partir de ellas. Además, también se proponen mecanismos para obtener rutas a partir de recomendaciones independientes utilizando técnicas de reranking.

- En el Capítulo 7 continuamos investigando el problema de la recomendación de POIs con especial énfasis en la alta escasez de datos de este dominio. Para tratar de paliar parcialmente este efecto, se propone el uso de técnicas de agregación de datos (basadas en el ámbito de cross-domain) para mejorar las recomendaciones en ciudades independientes utilizando información de otras ciudades en base a diferentes estrategias.

- En el Capítulo 8 se resumen las principales conclusiones de esta tesis y se discuten sus principales limitaciones y el trabajo futuro de investigación.

# Appendix C

# Conclusiones y Trabajo Futuro

Los Sistemas de Recomendación se han ido integrando progresivamente en un número cada vez mayor de aplicaciones, hasta el punto de que ya son imprescindibles para manejar la gran cantidad de datos disponibles en un gran número de empresas tecnológicas. Por ello, la investigación de estos sistemas es crucial para mejorar la experiencia de los usuarios proporcionándoles mejores recomendaciones, adaptándolas al contexto actual de los mismos.

Esta tesis se ha centrado en dos temas principales: la integración de la información temporal y secuencial en los Sistemas de Recomendación (tanto en la etapa de recomendación como en la de evaluación) y el estudio detallado del problema de la recomendación de Puntos de Interés analizando los principales problemas y retos, al tiempo que se proponen soluciones para paliarlos.

Para ello, hemos definido en primer lugar nuevas métricas que incorporan estos contextos para analizar las recomendaciones producidas por los algoritmos en los siguientes aspectos: frescura, anti-relevancia, secuencialidad, y sesgos en función de los atributos del usuario y del ítem, mostrando que toda esta información puede ser utilizada también para evaluar a los recomendadores. En segundo lugar, hemos propuesto una nueva similitud entre usuarios basada en el algoritmo de la Subsecuencia Común más Larga para integrarlo en los sistemas de recomendación basados en vecinos, que tiene en cuenta tanto la información temporal como la secuencial. Además, hemos propuesto una redefinición de los algoritmos basados en vecinos para hacer recomendaciones explotando la última interacción común entre el usuario objetivo y el resto de sus vecinos, produciendo recomendaciones mejores y más novedosas temporalmente que otros algoritmos del estado del arte, demostrando que los modelos simples siguen siendo aplicables en el área.

Además, hemos analizado en detalle el problema de recomendación de Puntos de Interés (POI en inglés) clasificando un gran número de propuestas recientes según el tipo de información, los algoritmos y la metodología de evaluación utilizada. Con esta revisión hemos podido constatar que el problema de la recomendación de POIs sigue siendo relevante en la actualidad, ya que cada año se proponen un mayor número de nuevos trabajos. Sin embargo, también hemos detectado un problema en cuanto a la reproducibilidad de los resultados obtenidos por los modelos, ya que la mayoría de los algoritmos no son comparables entre sí porque presentan importantes discrepancias en la forma de realizar las recomendaciones y sus evaluaciones. Posteriormente, también hemos explorado la recomendación de rutas a partir de POIs independientes, utilizando información categórica y secuencial para generar secuencias a partir de POIs independientes recomendados explotando técnicas de reranking. Aunque nuestros enfoques de reranking no han mostrado una superioridad notable con respecto a otros algoritmos en cuanto al acierto en el ránking de las recomendaciones, sí han mostrado resultados prometedores en otras dimensiones, como la mejora de la precisión de las categorías y la reducción de la distancia que deben seguir los usuarios en función de las recomendaciones recibidas. Por último, dado que en la recomendación de Puntos de Interés la escasez de información es un problema severo (más que en la recomendación clásica), hemos aplicado estrategias de agregación de datos basadas en técnicas de dominio cruzado para mejorar el rendimiento de los recomendadores de POIs en diferentes regiones. Utilizando estas sencillas estrategias hemos podido mejorar el rendimiento de algunos recomendadores en varias dimensiones, como el acierto y la cobertura de usuarios.

En este capítulo presentamos las principales conclusiones obtenidas en esta tesis. En la Sección C.1 proporcionamos más detalles sobre las contribuciones de esta investigación y en la Sección C.2 presentamos algunas direcciones de investigación que podrían abordarse en futuros trabajos.

## C.1 Resumen de contribuciones

En las siguientes subsecciones discutimos y resumimos las principales contribuciones de esta tesis abordando los objetivos de investigación expuestos en el Apéndice B. En primer lugar, para el OBJ1, revisamos el estado del arte de los enfoques de recomendación de POIs y los caracterizamos en términos de información, tipo de algoritmos y metodologías de evaluación utilizadas. Con respecto al OBJ2, desarrollamos un conjunto de métricas que incorporan información adicional como el tiempo, las secuencias y los atributos de usuarios y artículos, y las probamos en dos conocidos conjuntos de datos. Para el OBJ3, definimos una métrica de similitud entre usuarios explotando la Subsecuencia Común más Larga entre sus interacciones; también redefinimos la formulación de los sistemas de recomendación basados en vecinos generando un ranking para el usuario objetivo explotando sus últimas interacciones con otros usuarios en el sistema. Para el OBJ4, demostramos que podemos generar trayectorias completas a partir de los datos de Redes Sociales basadas en localización utilizando técnicas de reranking. Por último, para el OBJ5 estudiamos cómo las técnicas de agregación derivadas del área de dominio cruzado nos ayudan a mejorar el rendimiento y la cobertura de usuarios de los recomendadores de Puntos de Interés.

### C.1.1 Métricas alternativas para los Sistemas de Recomendación

En el Capítulo 4 presentamos un conjunto de nuevas métricas para ser utilizadas en los Sistemas de Recomendación que explotan la información temporal, secuencial y categórica. También proponemos variaciones de las métricas para tener en cuenta los casos en los que se recomiendan artículos con puntuaciones bajas. Demostramos la importancia de utilizar este tipo de información en la etapa de evaluación para seguir analizando los resultados de los recomendadores.

Con nuestras métricas de novedad temporal, hemos demostrado que existe una clara relación entre los elementos relevantes y su novedad temporal. Esto es realmente útil, ya que estas métricas nos permiten detectar posibles sesgos y estallidos temporales de las interacciones dentro del sistema. Además, nuestras métricas de novedad temporal nos permiten construir perfiles de artículos con repeticiones, lo que las hace aptas para otros dominios de recomendación como la música.

Utilizando nuestros modelos de anti-relevancia hemos determinado que, a veces, aunque los recomendadores sugieran artículos relevantes a los usuarios, también devuelven artículos que los usuarios han valorado negativamente. Por ello, creemos que cuando se dispone de esta información (es decir, cuando se utilizan conjuntos de datos con calificaciones explícitas), los artículos recomendados con una calificación muy baja deberían ser penalizados aún más al evaluar los recomendadores. Además, consideramos que es un aspecto importante a tener en cuenta, ya que en ocasiones una recomendación realmente mala puede provocar una gran desconfianza en los usuarios del sistema. De hecho, aunque pensamos que este aspecto debería ser analizado con mayor profundidad, no hemos encontrado muchos estudios que analicen las "malas" recomendaciones producidas por los algoritmos, por lo que consideramos que es una dimensión novedosa que este trabajo ha aportado al campo de la evaluación de los Sistemas de Recomendación.

En cuanto a los atributos de los usuarios y los artículos en las recomendaciones, hemos observado que los usuarios suelen clasificarse en grupos, que a su vez pueden obtener resultados muy diferentes según los grupos a los que pertenezcan. Esta observación conecta con el análisis de la equidad en la recomendación (ya que en esos casos, algunos usuarios obtendrán mejores o peores recomendaciones sólo por poseer algunas características inherentes), lo que implica que algunos modelos pueden no estar devolviendo recomendaciones justas. Con los atributos de los artículos podemos distinguir mejor los resultados devueltos por los algoritmos y aumentar el rendimiento obtenido por los algoritmos en conjuntos de datos muy dispersos. Sin embargo, creemos que permitir la evaluación con atributos debe hacerse con cuidado, ya que podemos acabar aumentando de forma poco realista el rendimiento de los recomendadores si no aplicamos las penalizaciones adecuadas.

Las métricas desarrolladas en el Capítulo 4 nos han permitido, por tanto, obtener una visión más completa de los resultados obtenidos por los recomendadores. Hemos comprobado la utilidad de nuestras métricas en dos conjuntos de datos conocidos: Movielens1M y Foursquare, con el fin de analizar las diferencias y similitudes en ambos dominios utilizando dos metodologías de evaluación diferentes (divisiones aleatorias y conscientes del tiempo). Por último, aunque se trata de un resultado ya conocido por la comunidad, hemos encontrado, de acuerdo con otros investigadores en la materia, diferencias en los resultados obtenidos por los algoritmos en función del escenario de recomendación o de la metodología de evaluación utilizada (tipo de partición de datos, ajuste de parámetros, optimización de los algoritmos, etc.), véase (Beel et al., 2016, Dacrema et al., 2019). Por este motivo, volvemos a insistir en la importancia de ser lo más transparente posible a la hora de evaluar un algoritmo de recomendación – o, de hecho, cualquier algoritmo de Machine Learning.

### C.1.2 Información Secuencial en recomendadores $k$-NN

En el Capítulo 5 presentamos dos propuestas complementarias para los sistemas de recomendación basado en vecinos. En primer lugar, propusimos adaptar el uso del algoritmo de la subsecuencia común más larga para considerarlo como una métrica de similitud clásica, como la similitud del coseno o la correlación de Pearson. Esta similitud, aunque es computacionalmente más costosa que las anteriores, tiene dos grandes ventajas. Por un lado, permite tener en cuenta los componentes secuenciales y, por otro, es lo suficientemente flexible como para operar con información adicional de los elementos, como sus atributos, para crear un algoritmo híbrido (con la ventaja adicional de que es fácil de explicar e implementar).

En segundo lugar, redefinimos los algoritmos de vecinos próximos para generar un ranking para el usuario objetivo basado en las últimas interacciones que tienen en común con sus vecinos correspondientes. Nuestra principal propuesta aquí fue utilizar técnicas de fusión de rankings, lo que nos permitió generar recomendaciones que integran información secuencial. Además, esta nueva formulación es capaz de operar con cualquier métrica de similitud, tanto las clásicas como Pearson o coseno, como la métrica de similitud secuencial propuesta anteriormente en nuestro trabajo. Estas aproximaciones han sido evaluadas frente a otros algoritmos del estado del arte mostrando que en algunas circunstancias nuestra propuesta es más competitiva que el resto de los algoritmos, mientras que en otras situaciones siguen siendo eficaces. Con esta nueva formulación de los algoritmos más cercanos, demostramos que este tipo de propuestas siguen siendo aplicables y adaptables en el campo porque son más interpretables, eficientes (ya que se pueden paralelizar fácilmente) y más simples que otros algoritmos como las redes neuronales. Hemos probado nuestros recomendadores en dos conjuntos de datos diferentes con marcas de tiempo realistas: un subconjunto de MovieTweetings (del dominio de las películas) y un subconjunto del conjunto de datos Foursquare (del dominio de los Puntos de Interés) bajo dos metodologías de evaluación sensibles al tiempo, una más realista que considera una división temporal a nivel de sistema y otra división temporal por usuario para demostrar que nuestra propuesta puede obtener resultados competitivos en ambas metodologías.

### C.1.3 Nuevas perspectivas relativas a los sistemas de recomendación de Puntos de Interés

En el Capítulo 3 hemos realizado un estudio sobre el problema Puntos de Interés, mostrando que esta área sigue siendo relevante para los investigadores. Aunque hemos podido identificar que los algoritmos que explotan la información geográfica y temporal son ampliamente utilizados, también hemos detectado que no existe un protocolo de evaluación común para analizar el rendimiento de los recomendadores. En este sentido, hemos observado que la mayoría de los modelos trabajan con conjuntos de datos muy diferentes y utilizan metodologías de evaluación diversas (diferentes tipos de divisiones, diferencias en el filtrado de los datos, etc.), por lo que la comparación entre ellos resulta a veces inviable.

En el Capítulo 6 hemos explorado el problema de la recomendación de rutas o trayectorias a partir de la recomendación de POIs independientes. En primer lugar, definimos un framework para generar rutas a partir de los datos de las LBSNs y, a continuación, proponemos el uso de técnicas de reranking para generar rutas a partir de las recomendaciones producidas por los algoritmos de recomendación clásicos. En concreto, utilizamos tres técnicas diferentes de reranking: independiente, donde la puntuación de los ítems rerankeados sólo depende del par usuario-ítem, dependiente del último ítem, donde la puntuación de cada ítem rerankeado depende del ítem anterior, y finalmente los rerankers que dependen de toda la secuencia, donde los ítems son rerankeados para optimizar la ruta recomendada completa de acuerdo a diferentes condiciones (por ejemplo, distancia, probabilidad del ítem, o probabilidad de la categoría). Aunque nuestras técnicas de reranking no obtienen mejoras sustanciales en términos de relevancia, sí permiten mejoras en términos de aciertos de categorías de POIs al tiempo que consiguen niveles de precisión similares partiendo de métodos muy sencillos. Esto demuestra, en particular, que a veces se pueden generar rutas de forma sencilla aprovechando las recomendaciones producidas previamente por otros algoritmos. Para evaluar nuestras propuestas, hemos utilizado cuatro conjuntos de datos diferentes del mundo real. Nuestros métodos funcionaron bien en todos los casos, especialmente en el que contenía más información turística.

Por último, en el Capítulo 7 hemos desarrollado un conjunto de técnicas de agregación de varias ciudades con el fin de mejorar el rendimiento de los algoritmos de recomendación tanto clásicos como específicos de POIs. En concreto, hemos realizado recomendaciones a un subconjunto de ciudades independientes con más check-ins del conjunto de datos de Foursquare utilizando tres estrategias diferentes: seleccionando las ciudades geográficamente más cercanas de cada una de ellas, seleccionando el resto de ciudades de cada país de cada ciudad objetivo, y utilizando la información de las ciudades con más check-ins para realizar recomendaciones en cada ciudad de forma independiente. A través de nuestros experimentos, hemos podido comprobar que la precisión y la cobertura a nivel de usuario de la mayoría de los modelos puede mejorarse más utilizando las estrategias basadas en la proximidad o por país que utilizando las ciudades más populares, demostrando que los algoritmos dependen más de la calidad que de la cantidad de los datos. Además, hemos podido comprobar que mientras que el componente geográfico es útil si realizamos recomendaciones para cada ciudad de forma independiente, cuando se utilizan estrategias de agregación el rendimiento no siempre mejora.

## C.2 Trabajo futuro

A lo largo de esta tesis hemos mostrado sólo una pequeña fracción del área de los Sistemas de Recomendación. De hecho, aunque hemos propuesto soluciones y avanzado en el estado del arte tanto en la recomendación clásica como en la de Puntos de Interés, creemos que algunas de estas aportaciones pueden ser ampliadas en el futuro. Por ello, en esta sección resumimos las principales líneas de investigación que pueden seguir desarrollándose.

### C.2.1 Sobre la evaluación en los Sistemas de Recomendación

En primer lugar, creemos que las métricas desarrolladas que se muestran en el Capítulo 4 tienen suficiente potencial para ser aplicadas en otras áreas de recomendación. Por ejemplo, creemos que las métricas de novedad dependientes del tiempo tienen un interés especial en los sistemas de recomendación de streaming, donde el modelo temporal del recomendador podría no ser el mismo que el de la métrica: por ejemplo, mientras que la métrica podría recalcularse cada día, ya que la novedad temporal es crucial en este dominio, el recomendador podría entrenarse una vez a la semana. De este modo, se puede realizar un análisis más detallado para, por ejemplo, calcular el periodo óptimo para entrenar al recomendador, o explorar la sensibilidad día a día de diferentes recomendadores. Vale la pena mencionar que, aunque esto también es posible lograrlo con datos offline, las conclusiones no serán tan significativas porque la mayoría de los conjuntos de datos son muy escasos, por lo que es necesario utilizar datos reales y de forma online.

En el caso del framework de anti-relevancia, nos proponemos ampliar el análisis de estas métricas a más familias de algoritmos y también en tareas de recomendación especialmente difíciles, como el arranque en frío o los dominios cruzados, para comprender el comportamiento de las técnicas de recomendación en esos escenarios. Y lo que es más importante, nos gustaría analizar cómo extender nuestro framework a situaciones en las que no se dispone de valoraciones explícitas, pero donde se puedan inferir otras formas de anti-relevancia, ya sea directamente o a través de la interacción del usuario con el sistema.

En cuanto a las características de los usuarios y los artículos, explotando los atributos de los usuarios podemos extender el análisis realizado en esta tesis a otros experimentos, como discriminar entre usuarios activos o influyentes, o entre bots o cualquier otro tipo de atacante o diferentes grupos de usuarios más allá de los turistas y los locales en el ámbito del turismo. Por otro lado, el análisis de los atributos de los ítems puede ser interesante para detectar sesgos en las recomendaciones (por ejemplo, si las recomendaciones están sesgadas hacia categorías específicas de los ítems o si los usuarios que pertenecen a un grupo concreto tienden a consumir ítems con una categoría distinta), así como para aplicar métricas de relevancia para emparejar las categorías de los ítems recomendados con los ítems de prueba, en dominios donde hay un gran número de ítems, como en POI o la recomendación de música.

### C.2.2 Sobre la secuencialidad en los sistemas de recomedación basados en vecinos

Los resultados obtenidos por nuestra novedosa métrica de similitud basada en la Subsecuencia Común Más Larga y los resultados de nuestra reformulación de los algoritmos basados en vecinos próximos muestran que estos modelos simples siguen siendo competitivos y pueden adaptarse para incluir información temporal y/o secuencial. Como trabajo futuro, hemos planeado explorar el uso de funciones de agregación alternativas – como las basadas en la distribución de puntuación (Manmatha et al., 2001) – cuando se integren en nuestra propuesta. Además, debería realizarse un análisis exhaustivo – con más conjuntos de datos, algoritmos base como SVD++ con información temporal (Koren and Bell, 2015) u otras técnicas de Redes Neuronales (Hidasi et al., 2016, Donkers et al., 2017, He et al., 2017b), y otras metodologías de evaluación – para comprender mejor cada componente de los modelos propuestos. Por ejemplo, el número de elementos que se permite seleccionar antes y después de la última interacción común, junto con definiciones alternativas para las métricas de similitud sensibles a las secuencia. Como ejemplo, pretendemos ampliar la similitud propuesta basada en LCS explotando otras dimensiones sobre las que construir la secuencia (como las características de los ítems, las valoraciones, o combinaciones de las mismas) o incluso aplicando filtros para seleccionar aquellos ítems que han sido valorados con un valor superior a un umbral específico para crear las secuencias de usuario, como hemos analizado recientemente en Sánchez and Bellogín (2019). Sin embargo, para el dominio de POIs esto podría ser contraproducente, ya que podríamos acabar añadiendo más dispersión en los datos al filtrar demasiada información.

Además, sería interesante observar el impacto en el comportamiento online de los usuarios una vez que reciben las recomendaciones, como se analizó recientemente para las redes sociales en Falavarjani et al. (2019). También se podría realizar un análisis completo sobre el impacto de la optimización de los parámetros utilizando (o no) un subconjunto de validación. De hecho, hemos observado lo difícil que fue encontrar resultados consistentes (desde el subconjunto de validación hasta el conjunto de datos completo) en el Capítulo 4. Por ello, nos proponemos analizar estas cuestiones con más detalle en el futuro para obtener directrices o garantías teóricas de que los parámetros aprendidos utilizando una división temporal de validación se ajustarían bien utilizando los datos completos, ya que creemos que es la forma más realista de ajustar los parámetros, como si se hiciera en un sistema del mundo real.

### C.2.3 Sobre el problema de recomendación de Puntos de Interés

En cuanto al problema de la recomendación de POIs, gracias a la revisión bibliográfica realizada en el Capítulo 3, hemos podido identificar un gran número de propuestas que no son comparables entre sí, sobre todo porque la metodología de evaluación llevada a cabo en la mayoría de esos trabajos varía sustancialmente (por ejemplo, diferentes tipos de particiones de datos, datasets, ajuste de parámetros, etc.). Por esta razón, como trabajo futuro, creemos que es necesario llevar a cabo más estudios experimentales (como en Liu et al. (2017)) comparando los enfoques más recientes y más citados de la recomendación POIs mientras se realizan diferentes tipos de splits en al menos las siguientes condiciones: recomendación en la misma ciudad, recomendación en diferentes ciudades del mundo, y recomendación en regiones específicas. Esto nos ayudará a identificar si los algoritmos pueden explotar correctamente la influencia geográfica en cada una de estas situaciones. Además, este análisis puede servir para entender con más detalle si hay más algoritmos que puedan beneficiarse de las estrategias entre dominios mencionadas en el Capítulo 7.

Con respecto a estas estrategias, consideramos que es importante utilizar otras estrategias de agregación que, en lugar de maximizar el número de usuarios coincidentes, se basen en el establecimiento de similitudes entre elementos. En este aspecto, consideramos que podría ser interesante ver cómo se comportan algoritmos como SLIM, FISM o embeddings de ítems en estas circunstancias.

Por último, nos gustaría insistir una vez más en que cualquier trabajo de investigación que se lleve a cabo debería publicar también el framework utilizado para realizar los experimentos, ya que es generalmente reconocido que el mismo modelo en diferentes frameworks podría producir resultados completamente diferentes (Said and Bellogín, 2014). Del mismo modo, todos los algoritmos básicos utilizados deberían ajustarse adecuadamente para encontrar un rendimiento competitivo frente a los algoritmos propuestos.

# References

Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In Yi Chang, Chengxiang Zhai, Yan Liu, and Yoelle Maarek, editors, *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*, pages 565–573. ACM, 2018. doi: 10.1145/3159652.3159656. URL https://doi.org/10.1145/3159652.3159656. vii, 19, 25, 26, 29, 83, 117, 119, 146

Irfan Ahmad. How much data is generated every minute? [infographic], Jun 2018. URL https://www.socialmediatoday.com/news/how-much-data-is-generated-every-minute-infographic-1/525692/. 3, 217

Pattie Maes. Agents that reduce work and information overload. *Commun. ACM*, 37(7):30–40, 1994. doi: 10.1145/176789.176792. URL https://doi.org/10.1145/176789.176792. 3, 217

Robert M. Bell and Yehuda Koren. Lessons from the netflix prize challenge. *SIGKDD Explorations*, 9(2):75–79, 2007. doi: 10.1145/1345448.1345465. URL https://doi.org/10.1145/1345448.1345465. 3, 19, 55, 217

Francesco Ricci, Lior Rokach, and Bracha Shapira. Recommender systems: Introduction and challenges. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 1–34. Springer, 2015. doi: 10.1007/978-1-4899-7637-6_1. URL https://doi.org/10.1007/978-1-4899-7637-6_1. 3, 4, 13, 14, 21, 218

Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. Current challenges and visions in music recommender systems research. *IJMIR*, 7(2):95–116, 2018. doi: 10.1007/s13735-018-0154-2. URL https://doi.org/10.1007/s13735-018-0154-2. 4, 77, 218

Dietmar Jannach, Iman Kamehkhosh, and Geoffray Bonnin. Music recommendations. In Shlomo Berkovsky, Iván Cantador, and Domonkos Tikk, editors, *Collaborative Recommendations - Algorithms, Practical Challenges and Applications*, pages 481–518. WorldScientific, 2018. doi: 10.1142/9789813275355_0015. URL https://doi.org/10.1142/9789813275355_0015. 4, 218

Mao Ye, Peifeng Yin, Wang-Chien Lee, and Dik Lun Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In Wei-Ying Ma, Jian-Yun Nie, Ricardo A. Baeza-Yates, Tat-Seng Chua, and W. Bruce Croft, editors, *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*, pages 325–334. ACM, 2011. doi: 10.1145/2009916.2009962. URL http://doi.acm.org/10.1145/2009916.2009962. 4, 36, 38, 48, 52, 53, 59, 167, 202, 218

Hao Wang, Manolis Terrovitis, and Nikos Mamoulis. Location recommendation in location-based social networks using user check-in data. In Craig A. Knoblock, Markus Schneider, Peer Kröger, John Krumm, and Peter Widmayer, editors, *21st SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2013, Orlando, FL, USA, November 5-8, 2013*, pages 364–373. ACM, 2013. doi: 10.1145/2525314.2525357. URL http://doi.acm.org/10.1145/2525314.2525357. 4, 36, 38, 58, 202, 218

Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 191–226. Springer, 2015. doi: 10.1007/978-1-4899-7637-6_6. URL https://doi.org/10.1007/978-1-4899-7637-6_6. 4, 22, 219

Norha M. Villegas, Cristian Sánchez, Javier Díaz-Cely, and Gabriel Tamura. Characterizing context-aware recommender systems: A systematic literature review. *Knowl.-Based Syst.*, 140:173–200, 2018. doi: 10.1016/j.knosys.2017.11.003. URL https://doi.org/10.1016/j.knosys.2017.11.003. 4, 22, 219

Christopher Laß, Daniel Herzog, and Wolfgang Wörndl. Context-aware tourist trip recommendations. In Julia Neidhardt, Daniel R. Fesenmaier, Tsvi Kuflik, and Wolfgang Wörndl, editors, *Proceedings of the 2nd Workshop on Recommenders in Tourism co-located with 11th ACM Conference on Recommender Systems (RecSys 2017), Como, Italy, August 27, 2017.*, volume 1906 of *CEUR Workshop Proceedings*, pages 18–25. CEUR-WS.org, 2017. URL http://ceur-ws.org/Vol-1906/paper3.pdf. 5, 219

Harald Steck. Evaluation of recommendations: rating-prediction and ranking. In Qiang Yang, Irwin King, Qing Li, Pearl Pu, and George Karypis, editors, *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013*, pages 213–220. ACM, 2013. doi: 10.1145/2507157.2507160. URL http://doi.acm.org/10.1145/2507157.2507160. 5, 109, 219

Asela Gunawardana and Guy Shani. Evaluating recommender systems. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 265–308. Springer, 2015. doi: 10.1007/978-1-4899-7637-6_8. URL https://doi.org/10.1007/978-1-4899-7637-6_8. 5, 26, 80, 128, 171, 180, 219

Pablo Castells, Neil J. Hurley, and Saul Vargas. Novelty and diversity in recommender systems. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 881–918. Springer, 2015. doi: 10.1007/978-1-4899-7637-6_26. URL https://doi.org/10.1007/978-1-4899-7637-6_26. 5, 31, 60, 75, 132, 143, 170, 219

Harald Steck. Calibrated recommendations. In Sole Pera, Michael D. Ekstrand, Xavier Amatriain, and John O'Donovan, editors, *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, pages 154–162. ACM, 2018. doi: 10.1145/3240323.3240372. URL https://doi.org/10.1145/3240323.3240372. 5, 220

Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. The unfairness of popularity bias in recommendation. In Robin Burke, Himan Abdollahpouri, Edward C. Malthouse, K. P. Thai, and Yongfeng Zhang, editors, *Proceedings of the Workshop on Recommendation in Multi-stakeholder Environments co-located with the 13th ACM Conference on Recommender Systems (RecSys 2019), Copenhagen, Denmark, September 20, 2019*, volume 2440 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019a. URL http://ceur-ws.org/Vol-2440/paper4.pdf. 5, 86, 220

Jöran Beel, Corinna Breitinger, Stefan Langer, Andreas Lommatzsch, and Bela Gipp. Towards reproducibility in recommender-systems research. *User Model. User Adapt. Interact.*, 26(1):69–101, 2016. doi: 10.1007/s11257-016-9174-x. URL https://doi.org/10.1007/s11257-016-9174-x. 5, 193, 220, 230

Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In Toine Bogers, Alan Said, Peter Brusilovsky, and Domonkos Tikk, editors, *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*, pages 101–109. ACM, 2019. doi: 10.1145/3298689.3347058. URL https://doi.org/10.1145/3298689.3347058. 5, 25, 92, 129, 193, 220, 230

John O'Donovan and Barry Smyth. Trust in recommender systems. In Robert St. Amant, John Riedl, and Anthony Jameson, editors, *Proceedings of the 10th International Conference on Intelligent User Interfaces, IUI 2005, San Diego, California, USA, January 10-13, 2005*, pages 167–174. ACM, 2005. doi: 10.1145/1040830.1040870. URL https://doi.org/10.1145/1040830.1040870. 13

Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005. doi: 10.1109/TKDE.2005.99. URL https://doi.org/10.1109/TKDE.2005.99. 13, 17, 20

Robin D. Burke. Hybrid web recommender systems. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web, Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*, pages 377–408. Springer, 2007. doi: 10.1007/978-3-540-72079-9_12. URL https://doi.org/10.1007/978-3-540-72079-9_12. 14, 21, 47

Marco de Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. Semantics-aware content-based recommender systems. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 119–159. Springer, 2015. doi: 10.1007/978-1-4899-7637-6_4. URL https://doi.org/10.1007/978-1-4899-7637-6_4. 15, 17, 145

Charu C. Aggarwal. *Recommender Systems - The Textbook*. Springer, 2016. ISBN 978-3-319-29657-9. doi: 10.1007/978-3-319-29659-3. URL https://doi.org/10.1007/978-3-319-29659-3. 15

Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval - the concepts and technology behind search, Second edition*. Pearson Education Ltd., Harlow, England, 2011. ISBN 978-0-321-41691-9. URL http://www.mir2ed.org/. 15, 17

Iván Cantador, Alejandro Bellogín, and David Vallet. Content-based recommendation in social tagging systems. In Xavier Amatriain, Marc Torrens, Paul Resnick, and Markus Zanker, editors, *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*, pages 237–240. ACM, 2010. doi: 10.1145/1864708.1864756. URL http://doi.acm.org/10.1145/1864708.1864756. 15

Pasquale Lops, Dietmar Jannach, Cataldo Musto, Toine Bogers, and Marijn Koolen. Trends in content-based recommendation - preface to the special issue on recommender systems based on rich item descriptions. *User Model. User Adapt. Interact.*, 29(2):239–249, 2019. doi: 10.1007/s11257-019-09231-w. URL https://doi.org/10.1007/s11257-019-09231-w. 16

Xavier Amatriain and Josep M. Pujol. Data mining methods for recommender systems. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 227–262. Springer, 2015. doi: 10.1007/978-1-4899-7637-6_7. URL https://doi.org/10.1007/978-1-4899-7637-6_7. 16

Xia Ning, Christian Desrosiers, and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 37–76. Springer, 2015. doi: 10.1007/978-1-4899-7637-6_2. URL https://doi.org/10.1007/978-1-4899-7637-6_2. 17, 18, 20, 36, 110

Yehuda Koren and Robert M. Bell. Advances in collaborative filtering. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 77–118. Springer, 2015. doi: 10.1007/978-1-4899-7637-6_3. URL https://doi.org/10.1007/978-1-4899-7637-6_3. 17, 18, 19, 22, 196, 233

Fabio Aiolli. Efficient top-n recommendation for very large scale binary rated datasets. In Qiang Yang, Irwin King, Qing Li, Pearl Pu, and George Karypis, editors, *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013*, pages 273–280. ACM, 2013. doi: 10.1145/2507157.2507189. URL http://doi.acm.org/10.1145/2507157.2507189. 18, 170

Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In Xavier Amatriain, Marc Torrens, Paul Resnick, and Markus Zanker, editors, *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*, pages 39–46. ACM, 2010. doi: 10.1145/1864708.1864721. URL https://doi.org/10.1145/1864708.1864721. 18

John S. Breese, David Heckerman, and Carl Myers Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In Gregory F. Cooper and Serafín Moral, editors, *UAI '98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, University of Wisconsin Business School, Madison, Wisconsin, USA, July 24-26, 1998*, pages 43–52. Morgan Kaufmann, 1998. URL https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=231&proceeding_id=14. 18

Antonio Hernando, Jesús Bobadilla, and Fernando Ortega. A non negative matrix factorization for collaborative filtering recommender systems based on a bayesian probabilistic model. *Knowl. Based Syst.*, 97:188–202, 2016. doi: 10.1016/j.knosys.2015.12.018. URL https://doi.org/10.1016/j.knosys.2015.12.018. 19

Qing Li, Sung-Hyon Myaeng, and Byeong Man Kim. A probabilistic music recommender considering user opinions and audio features. *Inf. Process. Manag.*, 43(2):473–487, 2007. doi: 10.1016/j.ipm.2006.07.005. URL https://doi.org/10.1016/j.ipm.2006.07.005. 19

Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In Alfredo Cuzzocrea, James Allan, Norman W. Paton, Divesh Srivastava, Rakesh Agrawal, Andrei Z. Broder, Mohammed J. Zaki, K. Selçuk Candan, Alexandros Labrinidis, Assaf Schuster, and Haixun Wang, editors, *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pages 843–852. ACM, 2018. doi: 10.1145/3269206.3271761. URL https://doi.org/10.1145/3269206.3271761. 19

Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 263–272. IEEE Computer Society, 2008. doi: 10.1109/ICDM.2008.22. URL https://doi.org/10.1109/ICDM.2008.22. 19, 28, 82, 117, 131

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In Jeff A. Bilmes and Andrew Y. Ng, editors, *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, pages 452–461. AUAI Press, 2009. URL https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=1630&proceeding_id=25. 19, 28, 44, 83, 117, 206

Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti, editors, *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 811–820. ACM, 2010. doi: 10.1145/1772690.1772773. URL http://doi.acm.org/10.1145/1772690.1772773. 20, 24, 28, 83, 129, 146

Daizong Ding, Mi Zhang, Shao-Yuan Li, Jie Tang, Xiaotie Chen, and Zhi-Hua Zhou. Baydnn: Friend recommendation with bayesian personalized ranking deep neural network. In Ee-Peng Lim, Marianne Winslett, Mark Sanderson, Ada Wai-Chee Fu, Jimeng Sun, J. Shane Culpepper, Eric Lo, Joyce C. Ho, Debora Donato, Rakesh Agrawal, Yu Zheng, Carlos Castillo, Aixin Sun, Vincent S. Tseng, and Chenliang Li, editors, *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 1479–1488. ACM, 2017. doi: 10.1145/3132847.3132941. URL https://doi.org/10.1145/3132847.3132941. 20

Xutao Li, Gao Cong, Xiaoli Li, Tuan-Anh Nguyen Pham, and Shonali Krishnaswamy. Rank-geofm: A ranking based geographical factorization method for point of interest recommendation. In Ricardo A. Baeza-Yates, Mounia Lalmas, Alistair Moffat, and Berthier A. Ribeiro-Neto, editors, *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, pages 433–442. ACM, 2015a. doi: 10.1145/2766462.2767722. URL http://doi.acm.org/10.1145/2766462.2767722. 20, 36, 43, 52, 53, 59, 60, 146, 165, 172, 202

Fajie Yuan, Joemon M. Jose, Guibing Guo, Long Chen, Haitao Yu, and Rami Suleiman Alkhawaldeh. Joint geo-spatial preference and pairwise ranking for point-of-interest recommendation. In *28th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2016, San Jose, CA, USA, November 6-8, 2016*, pages 46–53. IEEE Computer Society, 2016. doi: 10.1109/ICTAI.2016.0018. URL https://doi.org/10.1109/ICTAI.2016.0018. 20, 35, 167, 169, 203

Alexandros Karatzoglou, Linas Baltrunas, and Yue Shi. Learning to rank for recommender systems. In Qiang Yang, Irwin King, Qing Li, Pearl Pu, and George Karypis, editors, *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013*, pages 493–494. ACM, 2013. doi: 10.1145/2507157.2508063. URL https://doi.org/10.1145/2507157.2508063. 20

Marko Balabanovic and Yoav Shoham. Content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, 1997. doi: 10.1145/245108.245124. URL http://doi.acm.org/10.1145/245108.245124. 21, 82

Luis M. de Campos, Juan M. Fernández-Luna, Juan F. Huete, and Miguel A. Rueda-Morales. Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks. *Int. J. Approx. Reason.*, 51(7):785–799, 2010. doi: 10.1016/j.ijar.2010.04.001. URL https://doi.org/10.1016/j.ijar.2010.04.001. 21

Prem Melville, Raymond J. Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. In Rina Dechter, Michael J. Kearns, and Richard S. Sutton, editors, *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence, July 28 - August 1, 2002, Edmonton, Alberta, Canada*, pages 187–192. AAAI Press / The MIT Press, 2002. URL http://www.aaai.org/Library/AAAI/2002/aaai02-029.php. 22

Christoph Trattner, Alexander Oberegger, Lukas Eberhard, Denis Parra, and Leandro Balby Marinho. Understanding the impact of weather for POI recommendations. In Daniel R. Fesenmaier, Tsvi Kuflik, and Julia Neidhardt, editors, *Proceedings of the Workshop on Recommenders in Tourism co-located with 10th ACM Conference on Recommender Systems (RecSys 2016), Boston, MA, USA, September 15, 2016*, volume 1685 of *CEUR Workshop Proceedings*, pages 16–23. CEUR-WS.org, 2016. URL http://ceur-ws.org/Vol-1685/paper3.pdf. 22, 203

Hakan Bagci and Pinar Karagoz. Random walk based context-aware activity recommendation for location based social networks. In *2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015, Campus des Cordeliers, Paris, France, October 19-21, 2015*, pages 1–9. IEEE, 2015. doi: 10.1109/DSAA.2015.7344852. URL https://doi.org/10.1109/DSAA.2015.7344852. 22, 202

Jia-Dong Zhang and Chi-Yin Chow. Core: Exploiting the personalized influence of two-dimensional geographic coordinates for location recommendations. *Inf. Sci.*, 293:163–181, 2015a. doi: 10.1016/j.ins.2014.09.014. URL https://doi.org/10.1016/j.ins.2014.09.014. 22, 59, 202

Xingyi Ren, Meina Song, Haihong E, and Junde Song. Context-aware probabilistic matrix factorization modeling for point-of-interest recommendation. *Neurocomputing*, 241:38–55, 2017. doi: 10.1016/j.neucom.2017.02.005. URL https://doi.org/10.1016/j.neucom.2017.02.005. 22, 38, 44, 45, 52, 53, 59, 204

Yi Ding and Xue Li. Time weight collaborative filtering. In Otthein Herzog, Hans-Jörg Schek, Norbert Fuhr, Abdur Chowdhury, and Wilfried Teiken, editors, *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany, October 31 - November 5, 2005*, pages 485–492. ACM, 2005. doi: 10.1145/1099554.1099689. URL http://doi.acm.org/10.1145/1099554.1099689. 22, 29, 106

Pedro G. Campos, Fernando Díez, and Iván Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Model. User-Adapt. Interact.*, 24(1-2):67–119, 2014. doi: 10.1007/s11257-012-9136-x. URL https://doi.org/10.1007/s11257-012-9136-x. 23, 28, 30, 48, 49, 106, 130

Ruining He and Julian McAuley. Fusing similarity models with markov chains for sparse sequential recommendation. In Francesco Bonchi, Josep Domingo-Ferrer, Ricardo A. Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu, editors, *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, pages 191–200. IEEE, 2016. doi: 10.1109/ICDM.2016.0030. URL https://doi.org/10.1109/ICDM.2016.0030. 23, 24, 83, 88, 114, 117, 119, 121, 146

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Trans. Assoc. Comput. Linguistics*, 5:339–351, 2017. URL https://transacl.org/ojs/index.php/tacl/article/view/1081. 25

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2015. URL http://download.tensorflow.org/paper/whitepaper2015.pdf. 25

Edo Liberty, Zohar S. Karnin, Bing Xiang, Laurence Rouesnel, Baris Coskun, Ramesh Nallapati, Julio Delgado, Amir Sadoughi, Yury Astashonok, Piali Das, Can Balioglu, Saswata Chakravarty, Madhav Jha, Philip Gautier, David Arpin, Tim Januschowski, Valentin Flunkert, Yuyang Wang, Jan Gasthaus, Lorenzo Stella, Syama Sundar Rangapuram, David Salinas, Sebastian Schelter, and

## REFERENCES

Alex Smola. Elastic machine learning algorithms in amazon sagemaker. In David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo, editors, *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, pages 731–737. ACM, 2020. doi: 10.1145/3318464.3386126. URL https://doi.org/10.1145/3318464.3386126. 25

Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv.*, 52(1):5:1–5:38, 2019a. doi: 10.1145/3285029. URL https://doi.org/10.1145/3285029. 25, 45

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.06939. 25, 29, 196, 233

Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. Sequential user-based recurrent neural network recommendations. In Paolo Cremonesi, Francesco Ricci, Shlomo Berkovsky, and Alexander Tuzhilin, editors, *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017*, pages 152–160. ACM, 2017. doi: 10.1145/3109859.3109877. URL https://doi.org/10.1145/3109859.3109877. 25, 196, 233

Zhi Li, Hongke Zhao, Qi Liu, Zhenya Huang, Tao Mei, and Enhong Chen. Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors. In Yike Guo and Faisal Farooq, editors, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 1734–1743. ACM, 2018a. doi: 10.1145/3219819.3220014. URL https://doi.org/10.1145/3219819.3220014. 25

Rong Hu and Pearl Pu. A comparative user study on rating vs. personality quiz based preference elicitation methods. In Cristina Conati, Mathias Bauer, Nuria Oliver, and Daniel S. Weld, editors, *Proceedings of the 14th International Conference on Intelligent User Interfaces, IUI 2009, Sanibel Island, Florida, USA, February 8-11, 2009*, pages 367–372. ACM, 2009. doi: 10.1145/1502650.1502702. URL https://doi.org/10.1145/1502650.1502702. 27

Margaret L Russell, Donna G Moralejo, and Ellen D Burgess. Paying research subjects: participants' perspectives. *Journal of Medical Ethics*, 26(2):126–130, 2000. ISSN 0306-6800. doi: 10.1136/jme.26.2.126. URL https://jme.bmj.com/content/26/2/126. 27

Ron Kohavi, Roger Longbotham, Dan Sommerfield, and Randal M. Henne. Controlled experiments on the web: survey and practical guide. *Data Min. Knowl. Discov.*, 18 (1):140–181, 2009. doi: 10.1007/s10618-008-0114-1. URL https://doi.org/10.1007/s10618-008-0114-1. 27

Bart P. Knijnenburg and Martijn C. Willemsen. Evaluating recommender systems with user experiments. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 309–352. Springer, 2015. doi: 10.1007/978-1-4899-7637-6_9. URL https://doi.org/10.1007/978-1-4899-7637-6_9. 27

Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. Product-based neural networks for user response prediction. In Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu, editors, *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, pages 1149–1154. IEEE Computer Society, 2016. doi: 10.1109/ICDM.2016.0151. URL https://doi.org/10.1109/ICDM.2016.0151. 29

Huayu Li, Yong Ge, Richang Hong, and Hengshu Zhu. Point-of-interest recommendations: Learning potential check-ins from friends. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 975–984. ACM, 2016a. doi: 10.1145/2939672.2939767. URL https://doi.org/10.1145/2939672.2939767. 29, 52, 53, 59, 203

Carl Yang, Lanxiao Bai, Chao Zhang, Quan Yuan, and Jiawei Han. Bridging collaborative filtering and semi-supervised learning: A neural approach for POI recommendation. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 1245–1254. ACM, 2017a. doi: 10.1145/3097983.3098094. URL https://doi.org/10.1145/3097983.3098094. 29, 46, 52, 53, 58, 59, 60, 203

Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. A simple convolutional generative network for next item recommendation. In J. Shane Culpepper, Alistair Moffat, Paul N. Bennett, and Kristina Lerman, editors, *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, pages 582–590. ACM, 2019. doi: 10.1145/3289600.3290975. URL https://doi.org/10.1145/3289600.3290975. 29

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002. doi: 10.1145/582415.582418. URL http://doi.acm.org/10.1145/582415.582418. 30

Sean M. McNee, John Riedl, and Joseph A. Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In Gary M. Olson and Robin Jeffries, editors, *Extended Abstracts Proceedings of the 2006 Conference on Human Factors in Computing Systems, CHI 2006, Montréal, Québec, Canada, April 22-27, 2006*, pages 1097–1101. ACM, 2006. doi: 10.1145/1125451.1125659. URL http://doi.acm.org/10.1145/1125451.1125659. 30, 60

Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In Xavier Amatriain, Marc Torrens, Paul Resnick, and Markus Zanker, editors, *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*, pages 257–260. ACM, 2010. doi: 10.1145/1864708.1864761. URL https://doi.org/10.1145/1864708.1864761. 30

Saul Vargas and Pablo Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In Bamshad Mobasher, Robin D. Burke, Dietmar Jannach, and Gediminas Adomavicius, editors, *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011*, pages 109–116. ACM, 2011. doi: 10.1145/2043932.2043955. URL http://doi.acm.org/10.1145/2043932.2043955. 31, 32, 67, 68, 151

Saul Vargas and Pablo Castells. Improving sales diversity by recommending users to items. In Alfred Kobsa, Michelle X. Zhou, Martin Ester, and Yehuda Koren, editors, *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014*, pages 145–152. ACM, 2014. doi: 10.1145/2645710.2645744. URL https://doi.org/10.1145/2645710.2645744. 31

Barry Smyth and Paul McClave. Similarity vs. diversity. In David W. Aha and Ian D. Watson, editors, *Case-Based Reasoning Research and Development, 4th International Conference on Case-Based Reasoning, ICCBR 2001, Vancouver, BC, Canada, July 30 - August 2, 2001, Proceedings*, volume 2080 of *Lecture Notes in Computer Science*, pages 347–361. Springer, 2001. doi: 10.1007/3-540-44593-5_25. URL https://doi.org/10.1007/3-540-44593-5_25. 31

Alan Said and Alejandro Bellogín. Comparative recommender system evaluation: benchmarking recommendation frameworks. In Alfred Kobsa, Michelle X. Zhou, Martin Ester, and Yehuda Koren, editors, *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014*, pages 129–136. ACM, 2014. doi: 10.1145/2645710.2645746. URL http://doi.acm.org/10.1145/2645710.2645746. 32, 60, 63, 142, 170, 197, 234

Walid Krichene and Steffen Rendle. On sampled metrics for item recommendation. In Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash, editors, *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 1748–1757. ACM, 2020. URL https://dl.acm.org/doi/10.1145/3394486.3403226. 32

Jia-Dong Zhang and Chi-Yin Chow. igslr: personalized geo-social location recommendation: a kernel density estimation approach. In Craig A. Knoblock, Markus Schneider, Peer Kröger, John Krumm, and Peter Widmayer, editors, *21st SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2013, Orlando, FL, USA, November 5-8, 2013*, pages 324–333. ACM, 2013. doi: 10.1145/2525314.2525339. URL http://doi.acm.org/10.1145/2525314.2525339. 35, 59, 175, 202

Yuehua Wang, Zhinong Zhong, Anran Yang, and Ning Jing. A deep point-of-interest recommendation system in location-based social networks. In Ying Tan, Yuhui Shi, and Qirong Tang, editors, *Data Mining and Big Data - Third International Conference, DMBD 2018, Shanghai, China, June 17-22, 2018, Proceedings*, volume 10943 of *Lecture Notes in Computer Science*, pages 547–554. Springer, 2018a. doi: 10.1007/978-3-319-93803-5_51. URL https://doi.org/10.1007/978-3-319-93803-5_51. 35, 204

Yiding Liu, Tuan-Anh Pham, Gao Cong, and Quan Yuan. An experimental evaluation of point-of-interest recommendation in location-based social networks. *PVLDB*, 10(10): 1010–1021, 2017. URL http://www.vldb.org/pvldb/vol10/p1010-liu.pdf. 36, 42, 77, 128, 152, 170, 172, 197, 234

Dingqi Yang, Daqing Zhang, and Bingqing Qu. Participatory cultural mapping based on collective behavior data in location-based social networks. *ACM TIST*, 7(3):30:1–30:23, 2016. doi: 10.1145/2814575. URL http://doi.acm.org/10.1145/2814575. 36, 39, 60, 82, 138, 164, 165, 167, 188

Eunjoon Cho, Seth A. Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In Chid Apté, Joydeep Ghosh, and Padhraic Smyth, editors, *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 1082–1090. ACM, 2011. doi: 10.1145/2020408.2020579. URL https://doi.org/10.1145/2020408.2020579. 36

Pablo Sánchez and Alejandro Bellogín. Challenges on evaluating venue recommendation approaches: Position paper. In Julia Neidhardt, Wolfgang Wörndl, Tsvi Kuflik, and Markus Zanker, editors, *Proceedings of the Workshop on Recommenders in Tourism, RecTour 2018, co-located with the 12th ACM Conference on Recommender Systems (RecSys 2018), Vancouver, Canada, October 7, 2018.*, volume 2222 of *CEUR Workshop Proceedings*, pages 37–40. CEUR-WS.org, 2018a. URL http://ceur-ws.org/Vol-2222/paper8.pdf. 36, 50

Harvey J. Miller. Tobler's first law and spatial analysis. *Annals of the Association of American Geographers*, 94(2): 284–289, 2004. doi: 10.1111/j.1467-8306.2004.09402005.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8306.2004.09402005.x. 36, 127, 142

Yong Liu, Wei Wei, Aixin Sun, and Chunyan Miao. Exploiting geographical neighborhood characteristics for location recommendation. In Jianzhong Li, Xiaoyang Sean Wang, Minos N. Garofalakis, Ian Soboroff, Torsten Suel, and Min Wang, editors, *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pages 739–748. ACM, 2014. doi: 10.1145/2661829.2662002. URL http://doi.acm.org/10.1145/2661829.2662002. 36, 37, 43, 52, 53, 138, 146, 167, 170, 172, 202

Defu Lian, Cong Zhao, Xing Xie, Guangzhong Sun, Enhong Chen, and Yong Rui. Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation. In Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani, editors, *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 831–840. ACM, 2014. doi: 10.1145/2623330.2623638. URL http://doi.acm.org/10.1145/2623330.2623638. 36, 43, 52, 53, 58, 59, 202

Bo Hu and Martin Ester. Social topic modeling for point-of-interest recommendation in location-based social networks. In Ravi Kumar, Hannu Toivonen, Jian Pei, Joshua Zhexue Huang, and Xindong Wu, editors, *2014 IEEE International Conference on Data Mining, ICDM 2014, Shenzhen, China, December 14-17, 2014*, pages 845–850. IEEE Computer Society, 2014. doi: 10.1109/ICDM.2014.124. URL https://doi.org/10.1109/ICDM.2014.124. 37, 60, 202

Jarana Manotumruksa, Craig MacDonald, and Iadh Ounis. Regularising factorised models for venue recommendation using friends and their comments. In Snehasis Mukhopadhyay, ChengXiang Zhai, Elisa Bertino, Fabio Crestani, Javed Mostafa, Jie Tang, Luo Si, Xiaofang Zhou, Yi Chang, Yunyao Li, and Parikshit Sondhi, editors, *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 1981–1984. ACM, 2016. doi: 10.1145/2983323.2983889. URL https://doi.org/10.1145/2983323.2983889. 37, 203

Weizhi Nie, Anan Liu, Xiaorong Zhu, and Yuting Su. Quality models for venue recommendation in location-based social network. *Multimedia Tools Appl.*, 75(20):12521–12534, 2016. doi: 10.1007/s11042-014-2339-x. URL https://doi.org/10.1007/s11042-014-2339-x. 37, 203

Suhang Wang, Yilin Wang, Jiliang Tang, Kai Shu, Suhas Ranganath, and Huan Liu. What your images reveal: Exploiting visual contents for point-of-interest recommendation. In Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich, editors, *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 391–400. ACM, 2017a. doi: 10.1145/3038912.3052638. URL https://doi.org/10.1145/3038912.3052638. 37, 46, 52, 53, 204

Xin Zheng, Jialong Han, and Aixin Sun. A survey of location prediction on twitter. *IEEE Trans. Knowl. Data Eng.*, 30 (9):1652–1671, 2018. doi: 10.1109/TKDE.2018.2807840. URL https://doi.org/10.1109/TKDE.2018.2807840. 37, 42

Guo Zhong and Changyi Ma. Personalized poi recommendation model in lbsns. In Fatos Xhafa, Srikanta Patnaik, and Albert Y. Zomaya, editors, *Advances in Intelligent Systems and Interactive Applications*, pages 608–613. Springer, 2018. ISBN 978-3-319-69096-4. 37, 204

Yali Si, Fuzhi Zhang, and Wenyuan Liu. An adaptive point-of-interest recommendation method for location-based social networks based on user activity and spatial features. *Knowl.-Based Syst.*, 163:267–282, 2019. doi: 10.1016/j.knosys.2018.08.031. URL https://doi.org/10.1016/j.knosys.2018.08.031. 37, 48, 52, 53, 205

Josh Jia-Ching Ying, Eric Hsueh-Chan Lu, Wen-Ning Kuo, and Vincent S. Tseng. Urban point-of-interest recommendation by mining user check-in behaviors. In Ouri E. Wolfson and Yu Zheng, editors, *Proceedings of the ACM SIGKDD International Workshop on Urban Computing, UrbComp@KDD 2012, Beijing, China, August 12, 2012*, pages 63–70. ACM, 2012. doi: 10.1145/2346496.2346507. URL https://doi.org/10.1145/2346496.2346507. 37, 47, 52, 53, 58, 202

# REFERENCES

Jia-Dong Zhang and Chi-Yin Chow. Geosoca: Exploiting geographical, social and categorical correlations for point-of-interest recommendations. In Ricardo A. Baeza-Yates, Mounia Lalmas, Alistair Moffat, and Berthier A. Ribeiro-Neto, editors, *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, pages 443–452. ACM, 2015b. doi: 10.1145/2766462.2767711. URL http://doi.acm.org/10.1145/2766462.2767711. 37, 48, 52, 53, 59, 165, 169, 202

Nai-Hung Cheng and Chia-Hui Chang. Evaluation of social, geography, location effects for point-of-interest recommendation. In Wei Ding, Takashi Washio, Hui Xiong, George Karypis, Bhavani M. Thuraisingham, Diane J. Cook, and Xindong Wu, editors, *13th IEEE International Conference on Data Mining Workshops, ICDM Workshops, TX, USA, December 7-10, 2013*, pages 766–772. IEEE Computer Society, 2013. doi: 10.1109/ICDMW.2013.77. URL https://doi.org/10.1109/ICDMW.2013.77. 38, 54, 202

Mohamed Sarwat, Justin J. Levandoski, Ahmed Eldawy, and Mohamed F. Mokbel. Lars*: An efficient and scalable location-aware recommender system. *IEEE Trans. Knowl. Data Eng.*, 26(6):1384–1399, 2014. doi: 10.1109/TKDE.2013.29. URL https://doi.org/10.1109/TKDE.2013.29. 40

Wei Zhang and Jianyong Wang. Location and time aware social collaborative retrieval for new successive point-of-interest recommendation. In James Bailey, Alistair Moffat, Charu C. Aggarwal, Maarten de Rijke, Ravi Kumar, Vanessa Murdock, Timos K. Sellis, and Jeffrey Xu Yu, editors, *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, pages 1221–1230. ACM, 2015. doi: 10.1145/2806416.2806564. URL http://doi.acm.org/10.1145/2806416.2806564. 40, 202

Takeshi Kurashima, Tomoharu Iwata, Go Irie, and Ko Fujimura. Travel route recommendation using geotags in photo sharing sites. In Jimmy Huang, Nick Koudas, Gareth J. F. Jones, Xindong Wu, Kevyn Collins-Thompson, and Aijun An, editors, *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*, pages 579–588. ACM, 2010. doi: 10.1145/1871437.1871513. URL http://doi.acm.org/10.1145/1871437.1871513. 40

Xuefeng Chen, Yifeng Zeng, Gao Cong, Shengchao Qin, Yanping Xiang, and Yuanshun Dai. On information coverage for location category based point-of-interest recommendation. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 37–43. AAAI Press, 2015. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9703. 40

Panagiotis Symeonidis, Alexis Papadimitriou, Yannis Manolopoulos, Pinar Senkul, and Ismail Hakki Toroslu. Geo-social recommendations based on incremental tensor reduction and local path traversal. In Christian S. Jensen, Wang-Chien Lee, Yu Zheng, and Mohamed F. Mokbel, editors, *Proceedings of the 2011 International Workshop on Location Based Social Networks, LBSN 2011, November 1, 2011, Chicago, IL, USA, Proceedings*, pages 89–96. ACM, 2011. 40, 202

Panagiotis Symeonidis, Dimitrios Ntempos, and Yannis Manolopoulos. *Recommender Systems for Location-based Social Networks*. Springer Briefs in Electrical and Computer Engineering. Springer, 2014. ISBN 978-1-4939-0285-9. doi: 10.1007/978-1-4939-0286-6. URL https://doi.org/10.1007/978-1-4939-0286-6. 41

Yonghong Yu and Xingguo Chen. A survey of point-of-interest recommendation in location-based social networks. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. 41

Jie Bao, Yu Zheng, David Wilkie, and Mohamed F. Mokbel. Recommendations in location-based social networks: a survey. *GeoInformatica*, 19(3):525–565, 2015. doi: 10.1007/s10707-014-0220-8. URL https://doi.org/10.1007/s10707-014-0220-8. 41

Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati E. Pantziou. A survey on algorithmic approaches for solving tourist trip design problems. *J. Heuristics*, 20(3):291–328, 2014. doi: 10.1007/s10732-014-9242-5. URL https://doi.org/10.1007/s10732-014-9242-5. 41

Giannis Christoforidis, Pavlos Kefalas, Apostolos N. Papadopoulos, and Yannis Manolopoulos. Recommending points of interest in lbsns using deep learning techniques. In Petia D. Koprinkova-Hristova, Tuly Yildirim, Vincenzo Piuri, Lazaros S. Iliadis, and David Camacho, editors, *IEEE International Symposium on INnovations in Intelligent SysTems and Applications, INISTA 2019, Sofia, Bulgaria, July 3-5, 2019*, pages 1–6. IEEE, 2019. doi: 10.1109/INISTA.2019.8778310. URL https://doi.org/10.1109/INISTA.2019.8778310. 42

Zhijun Ding, Xiaolun Li, Changjun Jiang, and Mengchu Zhou. Objectives and state-of-the-art of location-based social network recommender systems. *ACM Comput. Surv.*, 51 (1):18:1–18:28, 2018a. doi: 10.1145/3154526. URL https://doi.org/10.1145/3154526. 42

Shenglin Zhao, Michael R. Lyu, and Irwin King. *Point-of-Interest Recommendation in Location-Based Social Networks*. Springer Briefs in Computer Science. Springer, 2018a. ISBN 978-981-13-1348-6. doi: 10.1007/978-981-13-1349-3. URL https://doi.org/10.1007/978-981-13-1349-3. 42

Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat-Thalmann. Time-aware point-of-interest recommendation. In Gareth J. F. Jones, Paraic Sheridan, Diane Kelly, Maarten de Rijke, and Tetsuya Sakai, editors, *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013*, pages 363–372. ACM, 2013. doi: 10.1145/2484028.2484030. URL https://doi.org/10.1145/2484028.2484030. 42, 52, 53, 55, 59, 60, 202

Dingqi Yang, Daqing Zhang, Zhiyong Yu, and Zhu Wang. A sentiment-enhanced personalized location recommendation system. In Gerd Stumme and Andreas Hotho, editors, *24th ACM Conference on Hypertext and Social Media (part of ECRC), HT '13, Paris, France - May 02 - 04, 2013*, pages 119–128. ACM, 2013. doi: 10.1145/2481492.2481505. URL https://doi.org/10.1145/2481492.2481505. 42, 52, 53, 56, 202

Justin J. Levandoski, Mohamed Sarwat, Ahmed Eldawy, and Mohamed F. Mokbel. LARS: A location-aware recommender system. In Anastasios Kementsietsidis and Marcos Antonio Vaz Salles, editors, *IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012*, pages 450–461. IEEE Computer Society, 2012. doi: 10.1109/ICDE.2012.54. URL https://doi.org/10.1109/ICDE.2012.54. 43, 52, 53, 162, 202

Bin Liu, Yanjie Fu, Zijun Yao, and Hui Xiong. Learning geographical preferences for point-of-interest recommendation. In Inderjit S. Dhillon, Yehuda Koren, Rayid Ghani, Ted E. Senator, Paul Bradley, Rajesh Parekh, Jingrui He, Robert L. Grossman, and Ramasamy Uthurusamy, editors, *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, pages 1043–1051. ACM, 2013a. doi: 10.1145/2487575.2487673. URL https://doi.org/10.1145/2487575.2487673. 43, 52, 53, 59, 202

Hao Wang, Huawei Shen, Wentao Ouyang, and Xueqi Cheng. Exploiting poi-specific geographical influence for point-of-interest recommendation. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018,* *Stockholm, Sweden*, pages 3877–3883. ijcai.org, 2018b. doi: 10.24963/ijcai.2018/539. URL https://doi.org/10.24963/ijcai.2018/539. 43, 44, 52, 53, 59, 204

Jing He, Xin Li, Lejian Liao, Dandan Song, and William K. Cheung. Inferring a personalized next point-of-interest recommendation model with latent behavior patterns. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 137–143. AAAI Press, 2016. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12361. 43, 52, 53, 59, 203

Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. Exploring temporal effects for location recommendation on location-based social networks. In Qiang Yang, Irwin King, Qing Li, Pearl Pu, and George Karypis, editors, *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013*, pages 93–100. ACM, 2013. doi: 10.1145/2507157.2507182. URL http://doi.acm.org/10.1145/2507157.2507182. 43, 52, 53, 55, 58, 59, 165, 202

Shenglin Zhao, Tong Zhao, Haiqin Yang, Michael R. Lyu, and Irwin King. STELLAR: spatial-temporal latent ranking for successive point-of-interest recommendation. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 315–322. AAAI Press, 2016a. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12249. 44, 52, 53, 55, 59, 61, 203

Shenglin Zhao, Tong Zhao, Irwin King, and Michael R. Lyu. Geo-teaser: Geo-temporal sequential embedding rank for point-of-interest recommendation. In Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich, editors, *Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia, April 3-7, 2017*, pages 153–162. ACM, 2017a. doi: 10.1145/3041021.3054138. URL https://doi.org/10.1145/3041021.3054138. 44, 52, 53, 59, 60, 61, 203

Lina Yao, Quan Z. Sheng, Xianzhi Wang, Wei Emma Zhang, and Yongrui Qin. Collaborative location recommendation by integrating multi-dimensional contextual information. *ACM Trans. Internet Techn.*, 18(3):32:1–32:24, 2018. doi: 10.1145/3134438. URL https://doi.org/10.1145/3134438. 44, 52, 53, 204

Rong Gao, Jing Li, Xuefei Li, Chengfang Song, and Yifei Zhou. A personalized point-of-interest recommendation model via fusion of geo-social information. *Neurocomputing*, 273:159–170, 2018a. doi: 10.1016/j.neucom.2017.08.020. URL https://doi.org/10.1016/j.neucom.2017.08.020. 44, 52, 53, 54, 204

Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. Content-aware point of interest recommendation on location-based social networks. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 1721–1727. AAAI Press, 2015a. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9560. 44, 52, 53, 202

Jia-Dong Zhang, Chi-Yin Chow, and Yanhua Li. LORE: exploiting sequential influence for location recommendations. In Yan Huang, Markus Schneider, Michael Gertz, John Krumm, and Jagan Sankaranarayanan, editors, *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Dallas/Fort Worth, TX, USA, November 4-7, 2014*, pages 103–112. ACM, 2014. doi: 10.1145/2666310.2666400. URL http://doi.acm.org/10.1145/2666310.2666400. 44, 48, 52, 53, 58, 59, 146, 202

Xin Li, Guandong Xu, Enhong Chen, and Yu Zong. Learning recency based comparative choice towards point-of-interest recommendation. *Expert Syst. Appl.*, 42(9):4274–4283, 2015b. doi: 10.1016/j.eswa.2015.01.054. URL https://doi.org/10.1016/j.eswa.2015.01.054. 44, 58, 60, 202

Yanchi Liu, Chuanren Liu, Bin Liu, Meng Qu, and Hui Xiong. Unified point-of-interest recommendation with temporal interval assessment. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1015–1024. ACM, 2016a. doi: 10.1145/2939672.2939773. URL https://doi.org/10.1145/2939672.2939773. 45, 52, 53, 203

Qing Guo, Yi Huang, and Yin-Leng Theng. Topic-sensitive location recommendation with spatial awareness. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, WI-IAT 2015, Singapore, December 6-9, 2015 - Volume I*, pages 237–243. IEEE Computer Society, 2015a. doi: 10.1109/WI-IAT.2015.203. URL https://doi.org/10.1109/WI-IAT.2015.203. 45, 58, 202

Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Chengqi Zhang. Modeling location-based user rating profiles for personalized recommendation. *TKDD*, 9(3):19:1–19:41, 2015. doi: 10.1145/2663356. URL https://doi.org/10.1145/2663356. 45, 52, 53, 202

Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. Where you like to go next: Successive point-of-interest recommendation. In Francesca Rossi, editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 2605–2611. IJCAI/AAAI, 2013. URL http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6633. 45, 52, 53, 202

Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. Personalized ranking metric embedding for next new POI recommendation. In Qiang Yang and Michael J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2069–2075. AAAI Press, 2015. URL http://ijcai.org/Abstract/15/293. 45, 52, 53, 202

Lei Tang, Dandan Cai, Zongtao Duan, Junchi Ma, Meng Han, and Hanbo Wang. Discovering travel community for POI recommendation on location-based social networks. *Complexity*, 2019:8503962:1–8503962:8, 2019. doi: 10.1155/2019/8503962. URL https://doi.org/10.1155/2019/8503962. 45, 52, 53, 205

Haochao Ying, Jian Wu, Guandong Xu, Yanchi Liu, Tingting Liang, Xiao Zhang, and Hui Xiong. Time-aware metric embedding with asymmetric projection for successive POI recommendation. *World Wide Web*, 22(5): 2209–2224, 2019. doi: 10.1007/s11280-018-0596-8. URL https://doi.org/10.1007/s11280-018-0596-8. 45, 52, 53, 205

Jarana Manotumruksa, Craig Macdonald, and Iadh Ounis. A contextual attention recurrent architecture for context-aware venue recommendation. In Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz, editors, *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 555–564. ACM, 2018. doi: 10.1145/3209978.3210042. URL https://doi.org/10.1145/3209978.3210042. 46, 52, 53, 59, 60, 204

Marwa Naili, Anja Habacha Chaïbi, and Henda Hajjami Ben Ghezala. Comparative study of word embedding methods in topic segmentation. In Cecilia Zanni-Merk, Claudia S. Frydman, Carlos Toro, Yulia Hicks, Robert J. Howlett, and Lakhmi C. Jain, editors, *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 21st International Conference KES-2017, Marseille, France, 6-8 September 2017*, volume 112 of *Procedia Computer Science*, pages 340–349. Elsevier, 2017. doi: 10.1016/j.procs.2017.08.009. URL https://doi.org/10.1016/j.procs.2017.08.009. 46

# REFERENCES

HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Trans. Knowl. Data Eng.*, 30(9):1616–1637, 2018. doi: 10.1109/TKDE.2018.2807452. URL https://doi.org/10.1109/TKDE.2018.2807452. 46

Tieyun Qian, Bei Liu, Quoc Viet Hung Nguyen, and Hongzhi Yin. Spatiotemporal representation learning for translation-based POI recommendation. *ACM Trans. Inf. Syst.*, 37(2):18:1–18:24, 2019. doi: 10.1145/3295499. URL https://doi.org/10.1145/3295499. 46, 52, 53, 59, 205

Anastasios Noulas, Salvatore Scellato, Neal Lathia, and Cecilia Mascolo. A random walk around the city: New venue recommendation in location-based social networks. In *2012 International Conference on Privacy, Security, Risk and Trust, PASSAT 2012, and 2012 International Confernece on Social Computing, SocialCom 2012, Amsterdam, Netherlands, September 3-5, 2012*, pages 144–153. IEEE Computer Society, 2012. doi: 10.1109/SocialCom-PASSAT.2012.70. URL https://doi.org/10.1109/SocialCom-PASSAT.2012.70. 47, 52, 53, 58, 202

Quan Yuan, Gao Cong, and Aixin Sun. Graph-based point-of-interest recommendation with geographical and temporal influences. In Jianzhong Li, Xiaoyang Sean Wang, Minos N. Garofalakis, Ian Soboroff, Torsten Suel, and Min Wang, editors, *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pages 659–668. ACM, 2014. doi: 10.1145/2661829.2661983. URL https://doi.org/10.1145/2661829.2661983. 47, 52, 53, 55, 59, 202

Jie Bao, Yu Zheng, and Mohamed F. Mokbel. Location-based and preference-aware recommendation using sparse geo-social networking data. In Isabel F. Cruz, Craig A. Knoblock, Peer Kröger, Egemen Tanin, and Peter Widmayer, editors, *SIGSPATIAL 2012 International Conference on Advances in Geographic Information Systems (formerly known as GIS), SIGSPATIAL'12, Redondo Beach, CA, USA, November 7-9, 2012*, pages 199–208. ACM, 2012. doi: 10.1145/2424321.2424348. URL https://doi.org/10.1145/2424321.2424348. 47, 52, 53, 59, 202

Josh Jia-Ching Ying, Wen-Ning Kuo, Vincent S. Tseng, and Eric Hsueh-Chan Lu. Mining user check-in behavior with a random walk for urban point-of-interest recommendations. *ACM TIST*, 5(3):40:1–40:26, 2014. doi: 10.1145/2523068. URL https://doi.org/10.1145/2523068. 47, 52, 53, 202

Bingrui Geng, Licheng Jiao, Maoguo Gong, Lingling Li, and Yue Wu. A two-step personalized location recommendation based on multi-objective immune algorithm. *Inf. Sci.*, 475:161–181, 2019. doi: 10.1016/j.ins.2018.09.068. URL https://doi.org/10.1016/j.ins.2018.09.068. 48, 52, 53, 205

Jing He, Xin Li, and Lejian Liao. Category-aware next point-of-interest recommendation via listwise bayesian personalized ranking. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 1837–1843. ijcai.org, 2017a. doi: 10.24963/ijcai.2017/255. URL https://doi.org/10.24963/ijcai.2017/255. 48, 52, 53, 59, 74, 75, 76, 138, 143, 204

Min Xie, Hongzhi Yin, Hao Wang, Fanjiang Xu, Weitong Chen, and Sen Wang. Learning graph-based POI embedding for location-based recommendation. In Snehasis Mukhopadhyay, ChengXiang Zhai, Elisa Bertino, Fabio Crestani, Javed Mostafa, Jie Tang, Luo Si, Xiaofang Zhou, Yi Chang, Yunyao Li, and Parikshit Sondhi, editors, *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 15–24. ACM, 2016a. doi: 10.1145/2983323.2983711. URL https://doi.org/10.1145/2983323.2983711. 48, 52, 53, 165, 203

Dawei Chen, Cheng Soon Ong, and Lexing Xie. Learning points and routes to recommend trajectories. In Snehasis Mukhopadhyay, ChengXiang Zhai, Elisa Bertino, Fabio Crestani, Javed Mostafa, Jie Tang, Luo Si, Xiaofang Zhou, Yi Chang, Yunyao Li, and Parikshit Sondhi, editors, *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 2227–2232. ACM, 2016a. doi: 10.1145/2983323.2983672. URL http://doi.acm.org/10.1145/2983323.2983672. 50, 77, 134

Pablo Sánchez and Alejandro Bellogín. Applying reranking strategies to route recommendation using sequence-aware evaluation. *User Model. User Adapt. Interact.*, 30(4):659–725, 2020. doi: 10.1007/s11257-020-09258-4. URL https://doi.org/10.1007/s11257-020-09258-4. 50

Hakan Bagci and Pinar Karagoz. Context-aware friend recommendation for location based social networks using random walk. In Jacqueline Bourdeau, Jim Hendler, Roger Nkambou, Ian Horrocks, and Ben Y. Zhao, editors, *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11-15, 2016, Companion Volume*, pages 531–536. ACM, 2016. doi: 10.1145/2872518.2890466. URL https://doi.org/10.1145/2872518.2890466. 51

Cheng-Hao Chu, Wan-Chuen Wu, Cheng-Chi Wang, Tzung-Shi Chen, and Jen-Jee Chen. Friend recommendation for location-based mobile social networks. In Leonard Barolli, Ilsun You, Fatos Xhafa, Fang-Yie Leu, and Hsing-Chung Chen, editors, *Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2013, Taichung, Taiwan, July 3-5, 2013*, pages 365–370. IEEE Computer Society, 2013. doi: 10.1109/IMIS.2013.68. URL https://doi.org/10.1109/IMIS.2013.68. 51

Frederick Ayala-Gómez, Bálint Daróczy, Michael Mathioudakis, András A. Benczúr, and Aristides Gionis. Where could we go?: Recommendations for groups in location-based social networks. In Peter Fox, Deborah L. McGuinness, Lindsay Poirier, Paolo Boldi, and Katharina Kinder-Kurlanda, editors, *Proceedings of the 2017 ACM on Web Science Conference, WebSci 2017, Troy, NY, USA, June 25 - 28, 2017*, pages 93–102. ACM, 2017. doi: 10.1145/3091478.3091485. URL https://doi.org/10.1145/3091478.3091485. 51

Sanjay Purushotham, C.-C. Jay Kuo, Junaith Shahabdeen, and Lama Nachman. Collaborative group-activity recommendation in location-based social networks. In Rolf A. de By and Carola Wenk, editors, *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Crowdsourced and Volunteered Geographic Information, GeoCrowd '14, Dallas, Texas, USA, November 4, 2014*, pages 8–15. ACM, 2014. doi: 10.1145/2676440.2676442. URL https://doi.org/10.1145/2676440.2676442. 51

Chen Ma, Yingxue Zhang, Qinglong Wang, and Xue Liu. Point-of-interest recommendation: Exploiting self-attentive autoencoders with neighbor-aware influence. In Alfredo Cuzzocrea, James Allan, Norman W. Paton, Divesh Srivastava, Rakesh Agrawal, Andrei Z. Broder, Mohammed J. Zaki, K. Selçuk Candan, Alexandros Labrinidis, Assaf Schuster, and Haixun Wang, editors, *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pages 697–706. ACM, 2018. doi: 10.1145/3269206.3271733. URL https://doi.org/10.1145/3269206.3271733. 52, 53, 60, 204

Huiji Gao, Jiliang Tang, and Huan Liu. gscorr: modeling geo-social correlations for new check-ins on location-based social networks. In Xue-wen Chen, Guy Lebanon, Haixun Wang, and Mohammed J. Zaki, editors, *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 1582–1586. ACM, 2012. doi: 10.1145/2396761.2398477. URL http://doi.acm.org/10.1145/2396761.2398477. 54, 58

Chen Cheng, Haiqin Yang, Irwin King, and Michael R. Lyu. A unified point-of-interest recommendation framework in location-based social networks. *ACM TIST*, 8(1):10:1–10:21, 2016. doi: 10.1145/2901299. URL https://doi.org/10.1145/2901299. 54, 203

Alejandro Bellogín, Pablo Castells, and Iván Cantador. Statistical biases in information retrieval metrics for recommender systems. *Inf. Retr. Journal*, 20(6):606–634, 2017. doi: 10.1007/s10791-017-9312-z. URL https://doi.org/10.1007/s10791-017-9312-z. 57, 74, 213

Jia-Dong Zhang, Chi-Yin Chow, and Yanhua Li. igeorec: A personalized and efficient geographical location recommendation framework. *IEEE Trans. Services Computing*, 8 (5):701–714, 2015a. doi: 10.1109/TSC.2014.2328341. URL https://doi.org/10.1109/TSC.2014.2328341. 58, 59, 202

Torin Stepan, Jason M. Morawski, Scott Dick, and James Miller. Incorporating spatial, temporal, and social context in recommendations for location-based social networks. *IEEE Trans. Comput. Social Systems*, 3(4):164–175, 2016. doi: 10.1109/TCSS.2016.2631473. URL https://doi.org/10.1109/TCSS.2016.2631473. 58, 59, 203

Saeid Hosseini and Lei Thor Li. Point-of-interest recommendation using temporal orientations of users and locations. In Shamkant B. Navathe, Weili Wu, Shashi Shekhar, Xiaoyong Du, X. Sean Wang, and Hui Xiong, editors, *Database Systems for Advanced Applications - 21st International Conference, DASFAA 2016, Dallas, TX, USA, April 16-19, 2016, Proceedings, Part I*, volume 9642 of *Lecture Notes in Computer Science*, pages 330–347. Springer, 2016. doi: 10.1007/978-3-319-32025-0_21. URL https://doi.org/10.1007/978-3-319-32025-0_21. 58, 59, 203

Yan Chen, Xin Li, Lin Li, Guiquan Liu, and Guangdong Xu. Modeling user mobility via user psychological and geographical behaviors towards point of-interest recommendation. In Shamkant B. Navathe, Weili Wu, Shashi Shekhar, Xiaoyong Du, X. Sean Wang, and Hui Xiong, editors, *Database Systems for Advanced Applications - 21st International Conference, DASFAA 2016, Dallas, TX, USA, April 16-19, 2016, Proceedings, Part I*, volume 9642 of *Lecture Notes in Computer Science*, pages 364–380. Springer, 2016b. doi: 10.1007/978-3-319-32025-0_23. URL https://doi.org/10.1007/978-3-319-32025-0_23. 58, 203

Dequan Zhou and Xin Wang. Probabilistic category-based location recommendation utilizing temporal influence and geographical influence. In *International Conference on Data Science and Advanced Analytics, DSAA 2014, Shanghai, China, October 30 - November 1, 2014*, pages 115–121. IEEE, 2014. doi: 10.1109/DSAA.2014.7058061. URL https://doi.org/10.1109/DSAA.2014.7058061. 58, 202

Huayu Li, Yong Ge, Defu Lian, and Hao Liu. Learning user's intrinsic and extrinsic interests for point-of-interest recommendation: A unified approach. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 2117–2123. ijcai.org, 2017a. doi: 10.24963/ijcai.2017/294. URL https://doi.org/10.24963/ijcai.2017/294. 58, 59, 204

Lina Yao, Quan Z. Sheng, Yongrui Qin, Xianzhi Wang, Ali Shemshadi, and Qi He. Context-aware point-of-interest recommendation using tensor factorization with social regularization. In Ricardo Baeza-Yates, Mounia Lalmas, Alistair Moffat, and Berthier A. Ribeiro-Neto, editors, *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, pages 1007–1010. ACM, 2015. doi: 10.1145/2766462.2767794. URL https://doi.org/10.1145/2766462.2767794. 58, 202

Saurabh Gupta, Sayan Pathak, and Bivas Mitra. Complementary usage of tips and reviews for location recommendation in yelp. In Tru Cao, Ee-Peng Lim, Zhi-Hua Zhou, Tu Bao Ho, David Wai-Lok Cheung, and Hiroshi Motoda, editors, *Advances in Knowledge Discovery and Data Mining - 19th Pacific-Asia Conference, PAKDD 2015, Ho Chi Minh City, Vietnam, May 19-22, 2015, Proceedings, Part II*, volume 9078 of *Lecture Notes in Computer Science*, pages 720–731. Springer, 2015. doi: 10.1007/978-3-319-18032-8_56. URL https://doi.org/10.1007/978-3-319-18032-8_56. 58, 202

Ramesh Baral, Xiaolong Zhu, S. S. Iyengar, and Tao Li. Reel: R eview aware explanation of location recommendation. In Tanja Mitrovic, Jie Zhang, Li Chen, and David Chin, editors, *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization, UMAP 2018, Singapore, July 08-11, 2018*, pages 23–32. ACM, 2018. doi: 10.1145/3209219.3209237. URL https://doi.org/10.1145/3209219.3209237. 58, 204

Defu Lian, Yong Ge, Fuzheng Zhang, Nicholas Jing Yuan, Xing Xie, Tao Zhou, and Yong Rui. Content-aware collaborative filtering for location recommendation based on human mobility data. In Charu C. Aggarwal, Zhi-Hua Zhou, Alexander Tuzhilin, Hui Xiong, and Xindong Wu, editors, *2015 IEEE International Conference on Data Mining, ICDM 2015, Atlantic City, NJ, USA, November 14-17, 2015*, pages 261–270. IEEE Computer Society, 2015. doi: 10.1109/ICDM.2015.69. URL https://doi.org/10.1109/ICDM.2015.69. 58, 203

Defu Lian, Zhenyu Zhang, Yong Ge, Fuzheng Zhang, Nicholas Jing Yuan, and Xing Xie. Regularized content-aware tensor factorization meets temporal-aware location recommendation. In Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu, editors, *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, pages 1029–1034. IEEE Computer Society, 2016. doi: 10.1109/ICDM.2016.0131. URL https://doi.org/10.1109/ICDM.2016.0131. 58, 203

Logesh Ravi and V. Subramaniyaswamy. A reliable point of interest recommendation based on trust relevancy between users. *Wireless Personal Communications*, 97(2): 2751–2780, 2017a. doi: 10.1007/s11277-017-4633-1. URL https://doi.org/10.1007/s11277-017-4633-1. 58, 203

Ramesh Baral and Tao Li. MAPS: A multi aspect personalized POI recommender system. In Shilad Sen, Werner Geyer, Jill Freyne, and Pablo Castells, editors, *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 281–284. ACM, 2016. doi: 10.1145/2959100.2959187. URL https://doi.org/10.1145/2959100.2959187. 58, 203

Ramesh Baral, Dingding Wang, Tao Li, and Shu-Ching Chen. Geotecs: Exploiting geographical, temporal, categorical and social aspects for personalized POI recommendation (invited paper). In *17th IEEE International Conference on Information Reuse and Integration, IRI 2016, Pittsburgh, PA, USA, July 28-30, 2016*, pages 94–101. IEEE Computer Society, 2016. doi: 10.1109/IRI.2016.20. URL https://doi.org/10.1109/IRI.2016.20. 58, 203

Hamidu Abdel-Fatao, Jiuyong Li, and Jixue Liu. Unifying spatial, temporal and semantic features for an effective GPS trajectory-based location recommendation. In Mohamed A. Sharaf, Muhammad Aamir Cheema, and Jianzhong Qi, editors, *Databases Theory and Applications - 26th Australasian Database Conference, ADC 2015, Melbourne, VIC, Australia, June 4-7, 2015. Proceedings*, volume 9093 of *Lecture Notes in Computer Science*, pages 41–53. Springer, 2015. doi: 10.1007/978-3-319-19548-3_4. URL https://doi.org/10.1007/978-3-319-19548-3_4. 58, 202

Liang Zhu, Changqiao Xu, Jianfeng Guan, and Hongke Zhang. SEM-PPA: A semantical pattern and preference-aware service mining method for personalized point of interest recommendation. *J. Network and Computer Applications*, 82: 35–46, 2017. doi: 10.1016/j.jnca.2016.12.033. URL https://doi.org/10.1016/j.jnca.2016.12.033. 58, 204

Liwei Huang, Yutao Ma, and Yanbo Liu. Point-of-interest recommendation in location-based social networks with personalized geo-social influence. *China Communications*, 12(12):21–31, December 2015. ISSN 1673-5447. doi: 10.1109/CC.2015.7385525. 59, 202

## REFERENCES

Tieyun Qian, Bei Liu, Liang Hong, and Zhenni You. Time and location aware points of interest recommendation in location-based social networks. *J. Comput. Sci. Technol.*, 33(6):1219–1230, 2018. doi: 10.1007/s11390-018-1883-7. URL https://doi.org/10.1007/s11390-018-1883-7. 59, 204

Giannis Christoforidis, Pavlos Kefalas, Apostolos Papadopoulos, and Yannis Manolopoulos. Recommendation of points-of-interest using graph embeddings. In Francesco Bonchi, Foster J. Provost, Tina Eliassi-Rad, Wei Wang, Ciro Cattuto, and Rayid Ghani, editors, *5th IEEE International Conference on Data Science and Advanced Analytics, DSAA 2018, Turin, Italy, October 1-3, 2018*, pages 31–40. IEEE, 2018. doi: 10.1109/DSAA.2018.00013. URL https://doi.org/10.1109/DSAA.2018.00013. 59, 60, 205

Jia-Dong Zhang and Chi-Yin Chow. Ticrec: A probabilistic framework to utilize temporal influence correlations for time-aware location recommendations. *IEEE Trans. Services Computing*, 9(4):633–646, 2016. doi: 10.1109/TSC.2015.2413783. URL https://doi.org/10.1109/TSC.2015.2413783. 59, 203

Rong Gao, Jing Li, Xuefei Li, Chengfang Song, Jun Chang, Donghua Liu, and Chunzhi Wang. STSCR: exploring spatial-temporal sequential influence and social information for location recommendation. *Neurocomputing*, 319: 118–133, 2018b. doi: 10.1016/j.neucom.2018.07.041. URL https://doi.org/10.1016/j.neucom.2018.07.041. 59, 204

Rong Gao, Jing Li, Bo Du, Xuefei Li, Jun Chang, Chengfang Song, and Donghua Liu. Exploiting geo-social correlations to improve pairwise ranking for point-of-interest recommendation. *China Communications*, 15(7):180–201, July 2018. ISSN 1673-5447. doi: 10.1109/CC.2018.8424613. 59, 204

Jarana Manotumruksa, Craig Macdonald, and Iadh Ounis. A deep recurrent collaborative filtering framework for venue recommendation. In Ee-Peng Lim, Marianne Winslett, Mark Sanderson, Ada Wai-Chee Fu, Jimeng Sun, J. Shane Culpepper, Eric Lo, Joyce C. Ho, Debora Donato, Rakesh Agrawal, Yu Zheng, Carlos Castillo, Aixin Sun, Vincent S. Tseng, and Chenliang Li, editors, *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 1429–1438. ACM, 2017a. doi: 10.1145/3132847.3133036. URL http://doi.acm.org/10.1145/3132847.3133036. 59, 203

Jarana Manotumruksa, Craig Macdonald, and Iadh Ounis. A contextual recurrent collaborative filtering framework for modelling sequences of venue checkins. *Information Processing and Management*, page 102092, 2019a. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2019.102092. URL http://www.sciencedirect.com/science/article/pii/S0306457319301876. 59, 175, 205

Jarana Manotumruksa, Dimitrios Rafailidis, Craig Macdonald, and Iadh Ounis. On cross-domain transfer in venue recommendation. In Leif Azzopardi, Benno Stein, Norbert Fuhr, Philipp Mayr, Claudia Hauff, and Djoerd Hiemstra, editors, *Advances in Information Retrieval - 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14-18, 2019, Proceedings, Part I*, volume 11437 of *Lecture Notes in Computer Science*, pages 443–456. Springer, 2019b. doi: 10.1007/978-3-030-15712-8_29. URL https://doi.org/10.1007/978-3-030-15712-8_29. 59, 205

Bin Liu, Hui Xiong, Spiros Papadimitriou, Yanjie Fu, and Zijun Yao. A general geographical probabilistic factor model for point of interest recommendation. *IEEE Trans. Knowl. Data Eng.*, 27(5):1167–1179, 2015. doi: 10.1109/TKDE.2014.2362525. URL https://doi.org/10.1109/TKDE.2014.2362525. 59, 202

Min Xie, Hongzhi Yin, Fanjiang Xu, Hao Wang, and Xiaofang Zhou. Graph-based metric embedding for next POI recommendation. In Wojciech Cellary, Mohamed F. Mokbel, Jianmin Wang, Hua Wang, Rui Zhou, and Yanchun Zhang, editors, *Web Information Systems Engineering - WISE 2016 - 17th International Conference, Shanghai, China, November 8-10, 2016, Proceedings, Part II*, volume 10042 of *Lecture Notes in Computer Science*, pages 207–222, 2016b. doi: 10.1007/978-3-319-48743-4_17. URL https://doi.org/10.1007/978-3-319-48743-4_17. 59, 203

Saeid Hosseini, Hongzhi Yin, Meihui Zhang, Xiaofang Zhou, and Shazia Wasim Sadiq. Jointly modeling heterogeneous temporal properties in location recommendation. In K. Selçuk Candan, Lei Chen, Torben Bach Pedersen, Lijun Chang, and Wen Hua, editors, *Database Systems for Advanced Applications - 22nd International Conference, DASFAA 2017, Suzhou, China, March 27-30, 2017, Proceedings, Part I*, volume 10177 of *Lecture Notes in Computer Science*, pages 490–506. Springer, 2017. doi: 10.1007/978-3-319-55753-3_31. URL https://doi.org/10.1007/978-3-319-55753-3_31. 59, 204

Saeid Hosseini, Hongzhi Yin, Xiaofang Zhou, Shazia W. Sadiq, Mohammad Reza Kangavari, and Ngai-Man Cheung. Leveraging multi-aspect time-related influence in location recommendation. *World Wide Web*, 22(3):1001–1028, 2019. doi: 10.1007/s11280-018-0573-2. URL https://doi.org/10.1007/s11280-018-0573-2. 59, 205

Bin Xia, Yun Li, Qianmu Li, and Tao Li. Attention-based recurrent neural network for location recommendation. In Tianrui Li, Luis Martínez-López, and Yun Li, editors, *12th International Conference on Intelligent Systems and Knowledge Engineering, ISKE 2017, Nanjing, China, November 24-26, 2017*, pages 1–6. IEEE, 2017a. doi: 10.1109/ISKE.2017.8258747. URL https://doi.org/10.1109/ISKE.2017.8258747. 59, 204

Bin Xia, Tao Li, Qianmu Li, and Hong Zhang. Noise-tolerance matrix completion for location recommendation. *Data Min. Knowl. Discov.*, 32(1):1–24, 2018. doi: 10.1007/s10618-017-0516-z. URL https://doi.org/10.1007/s10618-017-0516-z. 59, 205

Naoki Kojima and Tomohiro Takagi. Location recommendation incorporating temporal and spatial effects. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, WI-IAT 2015, Singapore, December 6-9, 2015 - Volume I*, pages 280–283. IEEE Computer Society, 2015. doi: 10.1109/WI-IAT.2015.131. URL https://doi.org/10.1109/WI-IAT.2015.131. 59, 202

Pengpeng Zhao, Xiefeng Xu, Yanchi Liu, Ziting Zhou, Kai Zheng, Victor S. Sheng, and Hui Xiong. Exploiting hierarchical structures for POI recommendation. In Vijay Raghavan, Srinivas Aluru, George Karypis, Lucio Miele, and Xindong Wu, editors, *2017 IEEE International Conference on Data Mining, ICDM 2017, New Orleans, LA, USA, November 18-21, 2017*, pages 655–664. IEEE Computer Society, 2017b. doi: 10.1109/ICDM.2017.75. URL https://doi.org/10.1109/ICDM.2017.75. 59, 203

Yali Si, Fuzhi Zhang, and Wenyuan Liu. CTF-ARA: an adaptive method for POI recommendation based on check-in and temporal features. *Knowl.-Based Syst.*, 128:59–70, 2017. doi: 10.1016/j.knosys.2017.04.013. URL https://doi.org/10.1016/j.knosys.2017.04.013. 59, 204

Yonghong Yu, Hao Wang, Shuanzhu Sun, and Yang Gao. Exploiting location significance and user authority for point-of-interest recommendation. In Jinho Kim, Kyuseok Shim, Longbing Cao, Jae-Gil Lee, Xuemin Lin, and Yang-Sae Moon, editors, *Advances in Knowledge Discovery and Data Mining - 21st Pacific-Asia Conference, PAKDD 2017, Jeju, South Korea, May 23-26, 2017, Proceedings, Part II*, volume 10235 of *Lecture Notes in Computer Science*, pages 119–130, 2017. doi: 10.1007/978-3-319-57529-2_10. URL https://doi.org/10.1007/978-3-319-57529-2_10. 59, 204

Ying Xu, Ying Li, Wei Yang, and Jin Zhang. A multi-factor influencing POI recommendation model based on matrix factorization. In *Tenth International Conference on Advanced Computational Intelligence, ICACI 2018, Xiamen, China, March 29-31, 2018*, pages 514–519. IEEE, 2018a. doi: 10.1109/ICACI.2018.8377512. URL https://doi.org/10.1109/ICACI.2018.8377512. 59, 204

Jinpeng Chen, Wen Zhang, Pei Zhang, Pinguang Ying, Kun Niu, and Ming Zou. Exploiting spatial and temporal for point of interest recommendation. *Complexity*, 2018:6928605:1–6928605:16, 2018a. doi: 10.1155/2018/6928605. URL https://doi.org/10.1155/2018/6928605. 59, 204

K. U Kala and M Nandhini. Context-category specific sequence aware point-of-interest recommender system with multi-gated recurrent unit. *Journal of Ambient Intelligence and Humanized Computing*, 2019. doi: https://doi.org/10.1007/s12652-019-01583-w. 59, 205

Lu Gao, Yuhua Li, Ruixuan Li, Zhenlong Zhu, Xiwu Gu, and Olivier Habimana. St-rnet: A time-aware point-of-interest recommendation method based on neural network. In *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*, pages 1–8. IEEE, 2019. doi: 10.1109/IJCNN.2019.8852377. URL https://doi.org/10.1109/IJCNN.2019.8852377. 59, 205

Jun Zeng, Haoran Tang, Yinghua Li, and Xin He. A deep learning model based on sparse matrix for point-of-interest recommendation. In Angelo Perkusich, editor, *The 31st International Conference on Software Engineering and Knowledge Engineering, SEKE 2019, Hotel Tivoli, Lisbon, Portugal, July 10-12, 2019*, pages 379–492. KSI Research Inc. and Knowledge Systems Institute Graduate School, 2019. doi: 10.18293/SEKE2019-156. URL https://doi.org/10.18293/SEKE2019-156. 59, 205

Xin Li, Dongcheng Han, Jing He, Lejian Liao, and Mingzhong Wang. Next and next new POI recommendation via latent behavior pattern inference. *ACM Trans. Inf. Syst.*, 37(4):46:1–46:28, 2019a. doi: 10.1145/3354187. URL https://doi.org/10.1145/3354187. 59, 205

Yijun Su, Xiang Li, Wei Tang, Daren Zha, Ji Xiang, and Neng Gao. Personalized point-of-interest recommendation on ranking with poisson factorization. In *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*, pages 1–8. IEEE, 2019a. doi: 10.1109/IJCNN.2019.8852462. URL https://doi.org/10.1109/IJCNN.2019.8852462. 59, 205

Makbule Gulcin Ozsoy, Faruk Polat, and Reda Alhajj. Multi-objective optimization based location and social network aware recommendation. In Elisa Bertino, Shu-Ching Chen, Karl Aberer, Prashant Krishnamurthy, and Murat Kantarcioglu, editors, *10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2014, Miami, Florida, USA, October 22-25, 2014*, pages 233–242. ICST / IEEE, 2014. doi: 10.4108/icst.collaboratecom.2014.257382. URL https://doi.org/10.4108/icst.collaboratecom.2014.257382. 59, 202

Makbule Gulcin Ozsoy, Faruk Polat, and Reda Alhajj. Time preference aware dynamic recommendation enhanced with location, social network and temporal information. In Ravi Kumar, James Caverlee, and Hanghang Tong, editors, *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2016, San Francisco, CA, USA, August 18-21, 2016*, pages 909–916. IEEE Computer Society, 2016. doi: 10.1109/ASONAM.2016.7752347. URL https://doi.org/10.1109/ASONAM.2016.7752347. 59, 203

Jinghua Zhu, Qian Ming, and Yong Liu. Trust-distrust-aware point-of-interest recommendation in location-based social network. In Sriram Chellappan, Wei Cheng, and Wei Li, editors, *Wireless Algorithms, Systems, and Applications - 13th International Conference, WASA 2018, Tianjin, China, June 20-22, 2018, Proceedings*, volume 10874 of *Lecture Notes in Computer Science*, pages 709–719. Springer, 2018a. doi: 10.1007/978-3-319-94268-1_58. URL https://doi.org/10.1007/978-3-319-94268-1_58. 59, 61, 204

Jinghua Zhu, Chao Wang, Xu Guo, Qian Ming, Jinbao Li, and Yong Liu. Friend and POI recommendation based on social trust cluster in location-based social networks. *EURASIP J. Wireless Comm. and Networking*, 2019:89, 2019. doi: 10.1186/s13638-019-1388-2. URL https://doi.org/10.1186/s13638-019-1388-2. 59, 61

Ranzhen Li, Yanyan Shen, and Yanmin Zhu. Next point-of-interest recommendation with temporal and multi-level context attention. In *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*, pages 1110–1115. IEEE Computer Society, 2018b. doi: 10.1109/ICDM.2018.00144. URL https://doi.org/10.1109/ICDM.2018.00144. 60, 205

Dongjin Yu, Kaihui Xu, and Dongjing Wang. Modeling user contextual behavior semantics with geographical influence for point-of-interest recommendation. In Angelo Perkusich, editor, *The 31st International Conference on Software Engineering and Knowledge Engineering, SEKE 2019, Hotel Tivoli, Lisbon, Portugal, July 10-12, 2019*, pages 373–484. KSI Research Inc. and Knowledge Systems Institute Graduate School, 2019. doi: 10.18293/SEKE2019-178. URL https://doi.org/10.18293/SEKE2019-178. 60, 205

Timothy G. Armstrong, Alistair Moffat, William Webber, and Justin Zobel. Improvements that don't add up: ad-hoc retrieval results since 1998. In David Wai-Lok Cheung, Il-Yeol Song, Wesley W. Chu, Xiaohua Hu, and Jimmy J. Lin, editors, *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 601–610. ACM, 2009. doi: 10.1145/1645953.1646031. URL https://doi.org/10.1145/1645953.1646031. 63

Szu-Yu Chou, Yi-Hsuan Yang, and Yu-Ching Lin. Evaluating music recommendation in a real-world setting: On data splitting and evaluation metrics. In *2015 IEEE International Conference on Multimedia and Expo, ICME 2015, Turin, Italy, June 29 - July 3, 2015*, pages 1–6. IEEE Computer Society, 2015. doi: 10.1109/ICME.2015.7177456. URL https://doi.org/10.1109/ICME.2015.7177456. 69

Rosie Jones and Fernando Diaz. Temporal profiles of queries. *ACM Trans. Inf. Syst.*, 25(3):14, 2007. doi: 10.1145/1247715.1247720. URL https://doi.org/10.1145/1247715.1247720. 69

Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22 (1):5–53, 2004. doi: 10.1145/963770.963772. URL http://doi.acm.org/10.1145/963770.963772. 70, 75

Elisa Mena-Maldonado, Rocío Cañamares, Pablo Castells, Yongli Ren, and Mark Sanderson. Agreement and disagreement between true and false-positive metrics in recommender systems evaluation. In Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu, editors, *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 841–850. ACM, 2020. doi: 10.1145/3397271.3401096. URL https://doi.org/10.1145/3397271.3401096. 70

Michael D. Ekstrand and John Riedl. When recommenders fail: predicting recommender failure for algorithm selection and combination. In Padraig Cunningham, Neil J. Hurley, Ido Guy, and Sarabjot Singh Anand, editors, *Sixth ACM Conference on Recommender Systems, RecSys '12, Dublin, Ireland, September 9-13, 2012*, pages 233–236. ACM, 2012. doi: 10.1145/2365952.2366002. URL http://doi.acm.org/10.1145/2365952.2366002. 71

Huan Gui, Haishan Liu, Xiangrui Meng, Anmol Bhasin, and Jiawei Han. Downside management in recommender systems. In Ravi Kumar, James Caverlee, and Hanghang Tong, editors, *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2016, San Francisco, CA, USA, August 18-21, 2016*, pages 394–401. IEEE Computer Society, 2016. doi: 10.1109/ASONAM.2016.7752264. URL https://doi.org/10.1109/ASONAM.2016.7752264. 71

## REFERENCES

Ellen M. Voorhees. Measuring ineffectiveness. In Mark Sanderson, Kalervo Järvelin, James Allan, and Peter Bruza, editors, *SIGIR 2004: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Sheffield, UK, July 25-29, 2004*, pages 562–563. ACM, 2004. doi: 10.1145/1008992.1009121. URL http://doi.acm.org/10.1145/1008992.1009121. 71

Sung-Hyon Myaeng and Robert R. Korfhage. Integration of user profiles: models and experiments in information retrieval. *Inf. Process. Manage.*, 26(6):719–738, 1990. doi: 10.1016/0306-4573(90)90048-7. URL https://doi.org/10.1016/0306-4573(90)90048-7. 71

Nicholas J Belkin, Michael Cole, and Ralf Bierig. Is relevance the right criterion for evaluating interactive information retrieval. In *In Proceedings of the SIGIR 2008 Workshop on Beyond Binary Relevance: Preferences, Diversity, and SetLevel judgments*, 2008. 71

Rocío Cañamares and Pablo Castells. From the PRP to the low prior discovery recall principle for recommender systems. In Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz, editors, *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 1081–1084. ACM, 2018. doi: 10.1145/3209978.3210076. URL http://doi.acm.org/10.1145/3209978.3210076. 71

Charles L. A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. Novelty and diversity in information retrieval evaluation. In Sung-Hyon Myaeng, Douglas W. Oard, Fabrizio Sebastiani, Tat-Seng Chua, and Mun-Kew Leong, editors, *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008*, pages 659–666. ACM, 2008. doi: 10.1145/1390334.1390446. URL http://doi.acm.org/10.1145/1390334.1390446. 71

S. E. Robertson. Readings in information retrieval. In Karen Sparck Jones and Peter Willett, editors, *Readings in Information Retrieval*, chapter The Probability Ranking Principle in IR, pages 281–286. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997. ISBN 1-55860-454-5. URL http://dl.acm.org/citation.cfm?id=275537.275701. 72

Igo Ramalho Brilhante, José Antônio Fernandes de Macêdo, Franco Maria Nardini, Raffaele Perego, and Chiara Renso. Where shall we go today?: planning touristic tours with tripbuilder. In Qi He, Arun Iyengar, Wolfgang Nejdl, Jian Pei, and Rajeev Rastogi, editors, *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, pages 757–762. ACM, 2013. doi: 10.1145/2505515.2505643. URL https://doi.org/10.1145/2505515.2505643. 74, 75, 76, 138, 143

Enrico Palumbo, Giuseppe Rizzo, Raphaël Troncy, and Elena Baralis. Predicting your next stop-over from location-based social network data with recurrent neural networks. In Julia Neidhardt, Daniel R. Fesenmaier, Tsvi Kuflik, and Wolfgang Wörndl, editors, *Proceedings of the 2nd Workshop on Recommenders in Tourism co-located with 11th ACM Conference on Recommender Systems (RecSys 2017), Como, Italy, August 27, 2017.*, volume 1906 of *CEUR Workshop Proceedings*, pages 1–8. CEUR-WS.org, 2017. URL http://ceur-ws.org/Vol-1906/paper1.pdf. 74, 129, 143, 179

Kwan Hui Lim, Jeffrey Chan, Christopher Leckie, and Shanika Karunasekera. Personalized tour recommendation based on user interests and points of interest visit durations. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1778–1784. AAAI Press, 2015. URL http://ijcai.org/Abstract/15/253. 74, 128, 129, 143

Dietmar Jannach and Gediminas Adomavicius. Recommendations with a purpose. In Shilad Sen, Werner Geyer, Jill Freyne, and Pablo Castells, editors, *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 7–10. ACM, 2016. doi: 10.1145/2959100.2959186. URL https://doi.org/10.1145/2959100.2959186. 74

Michael Trusov, Anand V Bodapati, and Randolph E Bucklin. Determining influential users in internet social networks. *Journal of Marketing Research*, 47(4):643–658, 2010. 75

Alan Said and Alejandro Bellogín. Coherence and inconsistencies in rating behavior: estimating the magic barrier of recommender systems. *User Model. User-Adapt. Interact.*, 28(2):97–125, 2018. doi: 10.1007/s11257-018-9202-0. URL https://doi.org/10.1007/s11257-018-9202-0. 75

Yashar Deldjoo, Vito Walter Anelli, Hamed Zamani, Alejandro Bellogín, and Tommaso Di Noia. A flexible framework for evaluating user and item fairness in recommender systems. *User Modeling and User-Adapted Interaction*, Jan 2021. ISSN 1573-1391. doi: 10.1007/s11257-020-09285-1. URL https://doi.org/10.1007/s11257-020-09285-1. 75

Shir Frumerman, Guy Shani, Bracha Shapira, and Oren Sar Shalom. Are all rejected recommendations equally bad?: Towards analysing rejected recommendations. In George Angelos Papadopoulos, George Samaras, Stephan Weibelzahl, Dietmar Jannach, and Olga C. Santos, editors, *Proceedings of the 27th ACM Conference on User Modeling, Adaptation and Personalization, UMAP 2019, Larnaca, Cyprus, June 9-12, 2019.*, pages 157–165. ACM, 2019. doi: 10.1145/3320435.3320448. URL https://doi.org/10.1145/3320435.3320448. 75

Aditya Krishna Menon, Dawei Chen, Lexing Xie, and Cheng Soon Ong. Revisiting revisits in trajectory recommendation. In Jie Yang, Zhu Sun, Alessandro Bozzon, Jie Zhang, and Martha Larson, editors, *Proceedings of International Workshop on Citizens for Recommender Systems, CitRec@RecSys 2017, 31 August 2017, Como, Italy*, pages 2:1–2:6. ACM, 2017. doi: 10.1145/3127325.3127326. URL http://doi.acm.org/10.1145/3127325.3127326. 77

Diego Monti, Enrico Palumbo, Giuseppe Rizzo, and Maurizio Morisio. Sequeval: A framework to assess and benchmark sequence-based recommender systems. *CoRR*, abs/1810.04956, 2018. URL http://arxiv.org/abs/1810.04956. 77, 138, 139, 142

Hamed Zamani, Markus Schedl, Paul Lamere, and Ching-Wei Chen. An analysis of approaches taken in the ACM recsys challenge 2018 for automatic music playlist continuation. *ACM Trans. Intell. Syst. Technol.*, 10(5):57:1–57:21, 2019. doi: 10.1145/3344257. URL https://doi.org/10.1145/3344257. 77

François Maillet, Douglas Eck, Guillaume Desjardins, and Paul Lamere. Steerable playlist generation by learning song similarity from radio station playlists. In Keiji Hirata, George Tzanetakis, and Kazuyoshi Yoshii, editors, *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, Kobe International Conference Center, Kobe, Japan, October 26-30, 2009*, pages 345–350. International Society for Music Information Retrieval, 2009. URL http://ismir2009.ismir.net/proceedings/OS4-2.pdf. 77

Hoyoung Jeung, Man Lung Yiu, and Christian S. Jensen. Trajectory pattern mining. In Yu Zheng and Xiaofang Zhou, editors, *Computing with Spatial Trajectories*, pages 143–177. Springer, 2011. doi: 10.1007/978-1-4614-1629-6_5. URL https://doi.org/10.1007/978-1-4614-1629-6_5. 77

Alberto Apostolico. String editing and longest common subsequences. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages: Volume 2. Linear Modeling: Background and Application*, pages 361–398. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997. ISBN 978-3-662-07675-0. doi: 10.1007/978-3-662-07675-0_8. URL http://dx.doi.org/10.1007/978-3-662-07675-0_8. 78

Dan Gusfield. *Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology*. Cambridge University Press, 1997. ISBN 0-521-58519-8. doi: 10.1017/cbo9780511574931. URL https://doi.org/10.1017/cbo9780511574931. 81

F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *TiiS*, 5(4):19:1–19:19, 2016. doi: 10.1145/2827872. URL http://doi.acm.org/10.1145/2827872. 87, 114

F. T. de la Rosa, María Teresa Gómez López, and Rafael M. Gasca. Analysis and visualization of the DX community with information extracted from the web. In Kim Viborg Andersen, John K. Debenham, and Roland R. Wagner, editors, *Database and Expert Systems Applications, 16th International Conference, DEXA 2005, Copenhagen, Denmark, August 22-26, 2005, Proceedings*, volume 3588 of *Lecture Notes in Computer Science*, pages 726–735. Springer, 2005. doi: 10.1007/11546924_71. URL https://doi.org/10.1007/11546924_71. 109

Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the web. In Vincent Y. Shen, Nobuo Saito, Michael R. Lyu, and Mary Ellen Zurko, editors, *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, pages 613–622. ACM, 2001. doi: 10.1145/371920.372165. URL http://doi.acm.org/10.1145/371920.372165. 110

M. Elena Renda and Umberto Straccia. Web metasearch: Rank vs. score based rank aggregation methods. In Gary B. Lamont, Hisham Haddad, George A. Papadopoulos, and Brajendra Panda, editors, *Proceedings of the 2003 ACM Symposium on Applied Computing (SAC), March 9-12, 2003, Melbourne, FL, USA*, pages 841–846. ACM, 2003. doi: 10.1145/952532.952698. URL http://doi.acm.org/10.1145/952532.952698. 110, 133

Simon Dooms, Alejandro Bellogín, Toon De Pessemier, and Luc Martens. A framework for dataset benchmarking and its application to a new movie rating dataset. *ACM TIST*, 7(3):41:1–41:28, 2016. doi: 10.1145/2751565. URL http://doi.acm.org/10.1145/2751565. 114

Bernard J. Jansen, Amanda Spink, Chris Blakely, and Sherry Koshman. Defining a session on web search engines. *JASIST*, 58(6):862–871, 2007. doi: 10.1002/asi.20564. URL https://doi.org/10.1002/asi.20564. 128

Myra Spiliopoulou, Bamshad Mobasher, Bettina Berendt, and Miki Nakagawa. A framework for the evaluation of session reconstruction heuristics in web-usage analysis. *INFORMS Journal on Computing*, 15(2):171–190, 2003. doi: 10.1287/ijoc.15.2.171.14445. URL https://doi.org/10.1287/ijoc.15.2.171.14445. 128

David Ben-Shimon, Alexander Tsikinovsky, Michael Friedmann, Bracha Shapira, Lior Rokach, and Johannes Hoerle. Recsys challenge 2015 and the YOOCHOOSE dataset. In Hannes Werthner, Markus Zanker, Jennifer Golbeck, and Giovanni Semeraro, editors, *Proceedings of the 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria, September 16-20, 2015*, pages 357–358. ACM, 2015. URL https://dl.acm.org/citation.cfm?id=2798723. 128

Guimei Liu, Tam T. Nguyen, Gang Zhao, Wei Zha, Jianbo Yang, Jianneng Cao, Min Wu, Peilin Zhao, and Wei Chen. Repeat buyer prediction for e-commerce. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 155–164. ACM, 2016b. doi: 10.1145/2939672.2939674. URL https://doi.org/10.1145/2939672.2939674. 128

Munmun De Choudhury, Moran Feldman, Sihem Amer-Yahia, Nadav Golbandi, Ronny Lempel, and Cong Yu. Automatic construction of travel itineraries using social breadcrumbs. In Mark H. Chignell and Elaine G. Toms, editors, *HT'10, Proceedings of the 21st ACM Conference on Hypertext and Hypermedia, Toronto, Ontario, Canada, June 13-16, 2010*, pages 35–44. ACM, 2010. doi: 10.1145/1810617.1810626. URL http://doi.acm.org/10.1145/1810617.1810626. 128, 129, 176

Kwan Hui Lim, Jeffrey Chan, Christopher Leckie, and Shanika Karunasekera. Personalized trip recommendation for tourists based on user interests, points of interest visit durations and visit recency. *Knowl. Inf. Syst.*, 54 (2):375–406, 2018. doi: 10.1007/s10115-017-1056-y. URL https://doi.org/10.1007/s10115-017-1056-y. 128

Robin Burke, Michael P. O'Mahony, and Neil J. Hurley. Robust collaborative recommendation. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 961–995. Springer, 2015. doi: 10.1007/978-1-4899-7637-6_28. URL https://doi.org/10.1007/978-1-4899-7637-6_28. 128

Alan Said and Alejandro Bellogín. Replicable evaluation of recommender systems. In Hannes Werthner, Markus Zanker, Jennifer Golbeck, and Giovanni Semeraro, editors, *Proceedings of the 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria, September 16-20, 2015*, pages 363–364. ACM, 2015. URL https://dl.acm.org/citation.cfm?id=2792841. 129

Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems. In Tanja Mitrovic, Jie Zhang, Li Chen, and David Chin, editors, *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization, UMAP 2018, Singapore, July 08-11, 2018*, pages 373–374. ACM, 2018. doi: 10.1145/3209219.3209270. URL http://doi.acm.org/10.1145/3209219.3209270. 131

Rodrygo L. T. Santos, Craig MacDonald, and Iadh Ounis. Search result diversification. *Foundations and Trends in Information Retrieval*, 9(1):1–90, 2015. doi: 10.1561/1500000040. URL https://doi.org/10.1561/1500000040. 132

Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In Allan Ellis and Tatsuya Hagino, editors, *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005*, pages 22–32. ACM, 2005. doi: 10.1145/1060745.1060754. URL https://doi.org/10.1145/1060745.1060754. 132

Filip Radlinski and Susan T. Dumais. Improving personalized web search using result diversification. In Efthimis N. Efthimiadis, Susan T. Dumais, David Hawking, and Kalervo Järvelin, editors, *SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006*, pages 691–692. ACM, 2006. doi: 10.1145/1148170.1148320. URL https://doi.org/10.1145/1148170.1148320. 132

Rodrygo L. T. Santos, Craig Macdonald, and Iadh Ounis. Exploiting query reformulations for web search result diversification. In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti, editors, *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 881–890. ACM, 2010. doi: 10.1145/1772690.1772780. URL https://doi.org/10.1145/1772690.1772780. 132

Saul Vargas, Pablo Castells, and David Vallet. Intent-oriented diversity in recommender systems. In Wei-Ying Ma, Jian-Yun Nie, Ricardo A. Baeza-Yates, Tat-Seng Chua, and W. Bruce Croft, editors, *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*, pages 1211–1212. ACM, 2011. doi: 10.1145/2009916.2010124. URL https://doi.org/10.1145/2009916.2010124. 132

# REFERENCES

Jacek Wasilewski and Neil Hurley. Intent-aware item-based collaborative filtering for personalised diversification. In Tanja Mitrovic, Jie Zhang, Li Chen, and David Chin, editors, *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization, UMAP 2018, Singapore, July 08-11, 2018*, pages 81–89. ACM, 2018. doi: 10.1145/3209219.3209234. URL https://doi.org/10.1145/3209219.3209234. 132

Marius Kaminskas and Derek Bridge. Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems. *TiiS*, 7(1):2:1–2:42, 2017. doi: 10.1145/2926720. URL https://doi.org/10.1145/2926720. 132, 133

Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Managing popularity bias in recommender systems with personalized re-ranking. In Roman Barták and Keith W. Brawner, editors, *Proceedings of the Thirty-Second International Florida Artificial Intelligence Research Society Conference, Sarasota, Florida, USA, May 19-22 2019.*, pages 413–418. AAAI Press, 2019b. URL https://aaai.org/ocs/index.php/FLAIRS/FLAIRS19/paper/view/18199. 132

Xin Li, Mingming Jiang, Huiting Hong, and Lejian Liao. A time-aware personalized point-of-interest recommendation via high-order tensor factorization. *ACM Trans. Inf. Syst.*, 35(4):31:1–31:23, 2017b. doi: 10.1145/3057283. URL http://doi.acm.org/10.1145/3057283. 138

Gunjan Kumar, Houssem Jerbi, and Michael P. O'Mahony. Towards the recommendation of personalised activity sequences in the tourism domain. In Julia Neidhardt, Daniel R. Fesenmaier, Tsvi Kuflik, and Wolfgang Wörndl, editors, *Proceedings of the 2nd Workshop on Recommenders in Tourism co-located with 11th ACM Conference on Recommender Systems (RecSys 2017), Como, Italy, August 27, 2017.*, volume 1906 of *CEUR Workshop Proceedings*, pages 26–30. CEUR-WS.org, 2017. URL http://ceur-ws.org/Vol-1906/paper4.pdf. 142

Wayne Xin Zhao, Ningnan Zhou, Aixin Sun, Ji-Rong Wen, Jialong Han, and Edward Y. Chang. A time-aware trajectory embedding model for next-location recommendation. *Knowl. Inf. Syst.*, 56(3):559–579, 2018b. doi: 10.1007/s10115-017-1107-4. URL https://doi.org/10.1007/s10115-017-1107-4. 142

Pablo Sánchez and Alejandro Bellogín. Time-aware novelty metrics for recommender systems. In Gabriella Pasi, Benjamin Piwowarski, Leif Azzopardi, and Allan Hanbury, editors, *Advances in Information Retrieval - 40th European Conference on IR Research, ECIR 2018, Grenoble, France, March 26-29, 2018, Proceedings*, volume 10772 of *Lecture Notes in Computer Science*, pages 357–370. Springer, 2018b. doi: 10.1007/978-3-319-76941-7_27. URL https://doi.org/10.1007/978-3-319-76941-7_27. 145

Guibing Guo, Jie Zhang, Zhu Sun, and Neil Yorke-Smith. Librec: A java library for recommender systems. In Alexandra I. Cristea, Judith Masthoff, Alan Said, and Nava Tintarev, editors, *Posters, Demos, Late-breaking Results and Workshop Proceedings of the 23rd Conference on User Modeling, Adaptation, and Personalization (UMAP 2015), Dublin, Ireland, June 29 - July 3, 2015.*, volume 1388 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015b. URL http://ceur-ws.org/Vol-1388/demo_paper1.pdf. 146

Daniel Valcarce, Alejandro Bellogín, Javier Parapar, and Pablo Castells. Assessing ranking metrics in top-n recommendation. *Inf. Retr. J.*, 23(4):411–448, 2020. doi: 10.1007/s10791-020-09377-x. URL https://doi.org/10.1007/s10791-020-09377-x. 148

Dietmar Jannach, Lukas Lerche, Iman Kamehkhosh, and Michael Jugovac. What recommenders recommend: an analysis of recommendation biases and possible countermeasures. *User Model. User-Adapt. Interact.*, 25(5):427–491, 2015. doi: 10.1007/s11257-015-9165-3. URL https://doi.org/10.1007/s11257-015-9165-3. 151, 182, 186

Rocío Cañamares and Pablo Castells. A probabilistic reformulation of memory-based collaborative filtering: Implications on popularity biases. In Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen P. de Vries, and Ryen W. White, editors, *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 215–224. ACM, 2017. doi: 10.1145/3077136.3080836. URL http://doi.acm.org/10.1145/3077136.3080836. 151

Jie Lu, Vahid Behbood, Peng Hao, Hua Zuo, Shan Xue, and Guangquan Zhang. Transfer learning using computational intelligence: A survey. *Knowl.-Based Syst.*, 80:14–23, 2015a. doi: 10.1016/j.knosys.2015.01.010. URL https://doi.org/10.1016/j.knosys.2015.01.010. 162

Xu Yu, Yan Chu, Feng Jiang, Ying Guo, and Dunwei Gong. Svms classification based two-side cross domain collaborative filtering by inferring intrinsic user and item features. *Knowl.-Based Syst.*, 141:80–91, 2018. doi: 10.1016/j.knosys.2017.11.010. URL https://doi.org/10.1016/j.knosys.2017.11.010. 162

Yu Zheng. Methodologies for cross-domain data fusion: An overview. *IEEE Trans. Big Data*, 1(1):16–34, 2015. doi: 10.1109/TBDATA.2015.2465959. URL https://doi.org/10.1109/TBDATA.2015.2465959. 162

Michael D. Ekstrand, Mucun Tian, Ion Madrazo Azpiazu, Jennifer D. Ekstrand, Oghenemaro Anuyah, David McNeill, and Maria Soledad Pera. All the cool kids, how do they fit in?: Popularity and demographic biases in recommender evaluation and effectiveness. In Sorelle A. Friedler and Christo Wilson, editors, *Conference on Fairness, Accountability and Transparency, FAT 2018, 23-24 February 2018, New York, NY, USA*, volume 81 of *Proceedings of Machine Learning Research*, pages 172–186. PMLR, 2018. URL http://proceedings.mlr.press/v81/ekstrand18b.html. 162

Bora Edizel, Francesco Bonchi, Sara Hajian, André Panisson, and Tamir Tassa. Fairecsys: mitigating algorithmic bias in recommender systems. *Int. J. Data Sci. Anal.*, 9(2):197–213, 2020. doi: 10.1007/s41060-019-00181-5. URL https://doi.org/10.1007/s41060-019-00181-5. 162

Linus W. Dietz, Rinita Roy, and Wolfgang Wörndl. Characterisation of traveller types using check-in data from location-based social networks. In Juho Pesonen and Julia Neidhardt, editors, *Information and Communication Technologies in Tourism 2019, ENTER 2019, Proceedings of the International Conference in Nicosia, Cyprus, January 30-February 1, 2019*, pages 15–26. Springer, 2019. doi: 10.1007/978-3-030-05940-8_2. URL https://doi.org/10.1007/978-3-030-05940-8_2. 162

Iván Cantador, Ignacio Fernández-Tobías, Shlomo Berkovsky, and Paolo Cremonesi. Cross-domain recommender systems. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 919–959. Springer, 2015. doi: 10.1007/978-1-4899-7637-6_27. URL https://doi.org/10.1007/978-1-4899-7637-6_27. 163, 164, 166

Dimitrios Rafailidis and Fabio Crestani. A collaborative ranking model for cross-domain recommendations. In Ee-Peng Lim, Marianne Winslett, Mark Sanderson, Ada Wai-Chee Fu, Jimeng Sun, J. Shane Culpepper, Eric Lo, Joyce C. Ho, Debora Donato, Rakesh Agrawal, Yu Zheng, Carlos Castillo, Aixin Sun, Vincent S. Tseng, and Chenliang Li, editors, *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 2263–2266. ACM, 2017. doi: 10.1145/3132847.3133107. URL http://doi.acm.org/10.1145/3132847.3133107. 164

Shaghayegh Sahebi and Peter Brusilovsky. It takes two to tango: An exploration of domain pairs for cross-domain collaborative filtering. In Hannes Werthner, Markus Zanker, Jennifer Golbeck, and Giovanni Semeraro, editors, *Proceedings of the 9th ACM Conference on Recommender*

*Systems, RecSys 2015, Vienna, Austria, September 16-20, 2015*, pages 131–138. ACM, 2015. doi: 10.1145/2792838.2800188. URL http://doi.acm.org/10.1145/2792838.2800188. 164, 187

Jia-Dong Zhang and Chi-Yin Chow. Spatiotemporal sequential influence modeling for location recommendations: A gravity-based approach. *ACM TIST*, 7(1):11:1–11:25, 2015c. doi: 10.1145/2786761. URL https://doi.org/10.1145/2786761. 165

Chen Cheng, Haiqin Yang, Irwin King, and Michael R. Lyu. Fused matrix factorization with geographical and social influence in location-based social networks. In Jörg Hoffmann and Bart Selman, editors, *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*. AAAI Press, 2012. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/4748. 169

Jungkyu Han and Hayato Yamana. Geographic diversification of recommended pois in frequently visited areas. *ACM Trans. Inf. Syst.*, 38(1):1:1–1:39, 2020. doi: 10.1145/3362505. URL https://doi.org/10.1145/3362505. 170

Huiji Gao, Jiliang Tang, and Huan Liu. Addressing the cold-start problem in location recommendation using geo-social correlations. *Data Min. Knowl. Discov.*, 29(2):299–323, 2015b. doi: 10.1007/s10618-014-0343-4. URL https://doi.org/10.1007/s10618-014-0343-4. 175, 202

Xia Ning and George Karypis. SLIM: sparse linear methods for top-n recommender systems. In Diane J. Cook, Jian Pei, Wei Wang, Osmar R. Zaïane, and Xindong Wu, editors, *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*, pages 497–506. IEEE Computer Society, 2011. doi: 10.1109/ICDM.2011.134. URL https://doi.org/10.1109/ICDM.2011.134. 176

Santosh Kabbur, Xia Ning, and George Karypis. FISM: factored item similarity models for top-n recommender systems. In Inderjit S. Dhillon, Yehuda Koren, Rayid Ghani, Ted E. Senator, Paul Bradley, Rajesh Parekh, Jingrui He, Robert L. Grossman, and Ramasamy Uthurusamy, editors, *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, pages 659–667. ACM, 2013. doi: 10.1145/2487575.2487589. URL https://doi.org/10.1145/2487575.2487589. 176

R. Manmatha, Toni M. Rath, and Fangfang Feng. Modeling score distributions for combining the outputs of search engines. In W. Bruce Croft, David J. Harper, Donald H. Kraft, and Justin Zobel, editors, *SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, September 9-13, 2001, New Orleans, Louisiana, USA*, pages 267–275. ACM, 2001. doi: 10.1145/383952.384005. URL http://doi.acm.org/10.1145/383952.384005. 196, 233

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich, editors, *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 173–182. ACM, 2017b. doi: 10.1145/3038912.3052569. URL https://doi.org/10.1145/3038912.3052569. 196, 233

Pablo Sánchez and Alejandro Bellogín. Building user profiles based on sequences for content and collaborative filtering. *Inf. Process. Manage.*, 56(1):192–211, 2019. doi: 10.1016/j.ipm.2018.10.003. URL https://doi.org/10.1016/j.ipm.2018.10.003. 196, 233

Seyed Amin Mirlohi Falavarjani, Fattane Zarrinkalam, Jelena Jovanovic, Ebrahim Bagheri, and Ali A. Ghorbani. The reflection of offline activities on users' online social behavior: An observational study. *Inf. Process. Manag.*, 56(6), 2019. doi: 10.1016/j.ipm.2019.102070. URL https://doi.org/10.1016/j.ipm.2019.102070. 197, 233

Ning Zheng, Xiaoming Jin, and Lianghao Li. Cross-region collaborative filtering for new point-of-interest recommendation. In Leslie Carr, Alberto H. F. Laender, Bernadette Farias Lóscio, Irwin King, Marcus Fontoura, Denny Vrandecic, Lora Aroyo, José Palazzo M. de Oliveira, Fernanda Lima, and Erik Wilde, editors, *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume*, pages 45–46. International World Wide Web Conferences Steering Committee / ACM, 2013. doi: 10.1145/2487788.2487804. URL https://doi.org/10.1145/2487788.2487804. 202

Shenglin Zhao, Irwin King, and Michael R. Lyu. Capturing geographical influence in POI recommendations. In Minho Lee, Akira Hirose, Zeng-Guang Hou, and Rhee Man Kil, editors, *Neural Information Processing - 20th International Conference, ICONIP 2013, Daegu, Korea, November 3-7, 2013. Proceedings, Part II*, volume 8227 of *Lecture Notes in Computer Science*, pages 530–537. Springer, 2013. doi: 10.1007/978-3-642-42042-9_66. URL https://doi.org/10.1007/978-3-642-42042-9_66. 202

Seyyed Mohammadreza Rahimi and Xin Wang. Location recommendation based on periodicity of human activities and location categories. In Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu, editors, *Advances in Knowledge Discovery and Data Mining, 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17, 2013, Proceedings, Part II*, volume 7819 of *Lecture Notes in Computer Science*, pages 377–389. Springer, 2013. doi: 10.1007/978-3-642-37456-2_32. URL https://doi.org/10.1007/978-3-642-37456-2_32. 202

Xuelian Long and James Joshi. A hits-based POI recommendation algorithm for location-based social networks. In Jon G. Rokne and Christos Faloutsos, editors, *Advances in Social Networks Analysis and Mining 2013, ASONAM '13, Niagara, ON, Canada - August 25 - 29, 2013*, pages 642–647. ACM, 2013. doi: 10.1145/2492517.2492652. URL https://doi.org/10.1145/2492517.2492652. 202

Bin Liu and Hui Xiong. Point-of-interest recommendation in location based social networks with topic and location awareness. In *Proceedings of the 13th SIAM International Conference on Data Mining, May 2-4, 2013. Austin, Texas, USA*, pages 396–404. SIAM, 2013. doi: 10.1137/1.9781611972832.44. URL https://doi.org/10.1137/1.9781611972832.44. 202

Xin Liu, Yong Liu, Karl Aberer, and Chunyan Miao. Personalized point-of-interest recommendation by mining users' preference transition. In Qi He, Arun Iyengar, Wolfgang Nejdl, Jian Pei, and Rajeev Rastogi, editors, *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, pages 733–738. ACM, 2013b. doi: 10.1145/2505515.2505639. URL https://doi.org/10.1145/2505515.2505639. 202

Gregory Ference, Mao Ye, and Wang-Chien Lee. Location recommendation for out-of-town users in location-based social networks. In Qi He, Arun Iyengar, Wolfgang Nejdl, Jian Pei, and Rajeev Rastogi, editors, *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, pages 721–726. ACM, 2013. doi: 10.1145/2505515.2505637. URL https://doi.org/10.1145/2505515.2505637. 202

Chun Lu, Philippe Laublet, and Milan Stankovic. Ricochet: Context and complementarity-aware, ontology-based pois recommender system. In Maria Maleshkova, Ruben Verborgh, Steffen Stadtmüller, and Pedro A. Szekely, editors, *Proceedings of the Second Workshop on Services and Applications over Linked APIs and Data co-located with the 10th Extended Semantic Web Conference (ESWC 2014), Crete, Greece, May 26, 2014*, volume 1165 of *CEUR Workshop Proceedings*, pages 10–17. CEUR-WS.org, 2014. URL http://ceur-ws.org/Vol-1165/salad2014-3.pdf. 202

# REFERENCES

Pavlos Kosmides, Chara Remoundou, Konstantinos Demestichas, Ioannis Loumiotis, Evgenia Adamopoulou, and Michael Theologou. A location recommender system for location-based social networks. In *International Conference on Mathematics and Computers in Sciences and in Industry, MCSI ,2014*, pages 277–280. IEEE, Sep. 2014. doi: 10.1109/MCSI.2014.39. 202

Yi-Liang Zhao, Liqiang Nie, Xiangyu Wang, and Tat-Seng Chua. Personalized recommendations of locally interesting venues to tourists via cross-region community matching. *ACM TIST*, 5(3):50:1–50:26, 2014. doi: 10.1145/2532439. URL http://doi.acm.org/10.1145/2532439. 202

Iury Nunes and Leandro Balby Marinho. A personalized geographic-based diffusion model for location recommendations in LBSN. In Jussara M. Almeida, Álvaro R. Pereira Jr., Ricardo Baeza-Yates, and Fabrício Benevenuto, editors, *9th Latin American Web Congress, LA-WEB 2014, Ouro Preto, Minas Gerais, Brazil, 22-24 October, 2014*, pages 59–67. IEEE Computer Society, 2014. doi: 10.1109/LAWeb.2014.22. URL https://doi.org/10.1109/LAWeb.2014.22. 202

Henan Wang, Guoliang Li, and Jianhua Feng. Group-based personalized location recommendation on social networks. In Lei Chen, Yan Jia, Timos K. Sellis, and Guanfeng Liu, editors, *Web Technologies and Applications - 16th Asia-Pacific Web Conference, APWeb 2014, Changsha, China, September 5-7, 2014. Proceedings*, volume 8709 of *Lecture Notes in Computer Science*, pages 68–80. Springer, 2014. doi: 10.1007/978-3-319-11116-2_7. URL https://doi.org/10.1007/978-3-319-11116-2_7. 202

Benyou Zou, Cuiping Li, Liwen Tan, and Hong Chen. Location-based recommendation using incremental tensor factorization model. In Xudong Luo, Jeffrey Xu Yu, and Zhi Li, editors, *Advanced Data Mining and Applications - 10th International Conference, ADMA 2014, Guilin, China, December 19-21, 2014. Proceedings*, volume 8933 of *Lecture Notes in Computer Science*, pages 227–238. Springer, 2014. doi: 10.1007/978-3-319-14717-8_18. URL https://doi.org/10.1007/978-3-319-14717-8_18. 202

Tetsuya Fukuda and Masayoshi Aritsugi. A feasibility study of POI recommendation based on bursts of visits. In Gabriele Anderst-Kotsis and Maria Indrawan-Santiago, editors, *Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services, iiWAS 2015, Brussels, Belgium, December 11-13, 2015*, pages 40:1–40:5. ACM, 2015. doi: 10.1145/2837185.2837270. URL https://doi.org/10.1145/2837185.2837270. 202

Masoud Sattari, Ismail Hakki Toroslu, Pinar Karagoz, Panagiotis Symeonidis, and Yannis Manolopoulos. Extended feature combination model for recommendations in location-based mobile services. *Knowl. Inf. Syst.*, 44 (3):629–661, 2015. doi: 10.1007/s10115-014-0776-5. URL https://doi.org/10.1007/s10115-014-0776-5. 202

Jan Zahálka, Stevan Rudinac, and Marcel Worring. Interactive multimodal learning for venue recommendation. *IEEE Trans. Multimedia*, 17(12):2235–2244, 2015. doi: 10.1109/TMM.2015.2480007. URL https://doi.org/10.1109/TMM.2015.2480007. 202

Steven Mudda and Silvia Giordano. REGULA: utilizing the regularity of human mobility for location recommendation. In Farnoush Banaei Kashani, Chengyang Zhang, and Abdeltawab M. Hendawi, editors, *Proceedings of the 6th ACM SIGSPATIAL International Workshop on GeoStreaming, IWGS 2015, Bellevue, WA, USA, November 3-6, 2015*, pages 69–77. ACM, 2015. doi: 10.1145/2833165.2833172. URL https://doi.org/10.1145/2833165.2833172. 202

Jean-Benoît Griesner, Talel Abdessalem, and Hubert Naacke. POI recommendation: Towards fused matrix factorization with geographical and temporal influences. In Hannes Werthner, Markus Zanker, Jennifer Golbeck, and Giovanni Semeraro, editors, *Proceedings of the 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria, September 16-20, 2015*, pages 301–304. ACM, 2015. doi: 10.1145/2792838.2799679. URL http://doi.acm.org/10.1145/2792838.2799679. 202

Kunhui Lin, Jingjin Wang, Zhongnan Zhang, Yating Chen, and Zhentuan Xu. Adaptive location recommendation algorithm based on location-based social networks. In *2015 10th International Conference on Computer Science Education (ICCSE)*, pages 137–142, July 2015. doi: 10.1109/ICCSE.2015.7250231. 202

Ziyu Lu, Hao Wang, Nikos Mamoulis, Wenting Tu, and David W. Cheung. Personalized location recommendation by aggregating multiple recommenders in diversity. In Panagiotis Bouros, Neal Lathia, Matthias Renz, Francesco Ricci, and Dimitris Sacharidis, editors, *Proceedings of the Workshop on Location-Aware Recommendations, LocalRec 2015, co-located with the 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 19, 2015*, volume 1405 of *CEUR Workshop Proceedings*, pages 28–35. CEUR-WS.org, 2015b. URL http://ceur-ws.org/Vol-1405/paper-05.pdf. 202

Xiangyu Wang, Yi-Liang Zhao, Liqiang Nie, Yue Gao, Weizhi Nie, Zheng-Jun Zha, and Tat-Seng Chua. Semantic-based location recommendation with multimodal venue semantics. *IEEE Trans. Multimedia*, 17(3):409–419, 2015. doi: 10.1109/TMM.2014.2385473. URL https://doi.org/10.1109/TMM.2014.2385473. 202

Meng Qi, Xin Li, Lejian Liao, Dandan Song, and William K. Cheung. Deriving an effective hypergraph model for point of interest recommendation. In Songmao Zhang, Martin Wirsing, and Zili Zhang, editors, *Knowledge Science, Engineering and Management - 8th International Conference, KSEM 2015, Chongqing, China, October 28-30, 2015, Proceedings*, volume 9403 of *Lecture Notes in Computer Science*, pages 771–777. Springer, 2015. doi: 10.1007/978-3-319-25159-2_71. URL https://doi.org/10.1007/978-3-319-25159-2_71. 202

Xin Li, Guandong Xu, Enhong Chen, and Lin Li. MARS: A multi-aspect recommender system for point-of-interest. In Johannes Gehrke, Wolfgang Lehner, Kyuseok Shim, Sang Kyun Cha, and Guy M. Lohman, editors, *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, pages 1436–1439. IEEE Computer Society, 2015c. doi: 10.1109/ICDE.2015.7113395. URL https://doi.org/10.1109/ICDE.2015.7113395. 202

Xinqiang Zhao, Xin Li, Lejian Liao, Dandan Song, and William K. Cheung. Crafting a time-aware point-of-interest recommendation via pairwise interaction tensor factorization. In Songmao Zhang, Martin Wirsing, and Zili Zhang, editors, *Knowledge Science, Engineering and Management - 8th International Conference, KSEM 2015, Chongqing, China, October 28-30, 2015, Proceedings*, volume 9403 of *Lecture Notes in Computer Science*, pages 458–470. Springer, 2015. doi: 10.1007/978-3-319-25159-2_41. URL https://doi.org/10.1007/978-3-319-25159-2_41. 202

Jia-Dong Zhang, Chi-Yin Chow, and Yu Zheng. Orec: An opinion-based point-of-interest recommendation framework. In James Bailey, Alistair Moffat, Charu C. Aggarwal, Maarten de Rijke, Ravi Kumar, Vanessa Murdock, Timos K. Sellis, and Jeffrey Xu Yu, editors, *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, pages 1641–1650. ACM, 2015b. doi: 10.1145/2806416.2806516. URL https://doi.org/10.1145/2806416.2806516. 202

Huayu Li, Richang Hong, Shiai Zhu, and Yong Ge. Point-of-interest recommender systems: A separate-space perspective. In Charu C. Aggarwal, Zhi-Hua Zhou, Alexander Tuzhilin, Hui Xiong, and Xindong Wu, editors, *2015 IEEE International Conference on Data Mining, ICDM 2015, Atlantic City, NJ, USA, November 14-17, 2015*, pages 231–240. IEEE Computer Society, 2015d. doi: 10.1109/ICDM.2015.27. URL https://doi.org/10.1109/ICDM.2015.27. 203

Ming Li, Günther Sagl, Lucy W. Mburu, and Hongchao Fan. A contextualized and personalized model to predict user interest using location-based social networks. *Computers, Environment and Urban Systems*, 58:97–106, 2016b. doi: 10.1016/j.compenvurbsys.2016.03.006. URL https://doi.org/10.1016/j.compenvurbsys.2016.03.006. 203

Khadija Vakeel and Sanjog Ray. A motivation-aware approach for point of interest recommendations. In Daniel R. Fesenmaier, Tsvi Kuflik, and Julia Neidhardt, editors, *Proceedings of the Workshop on Recommenders in Tourism co-located with 10th ACM Conference on Recommender Systems (RecSys 2016), Boston, MA, USA, September 15, 2016*, volume 1685 of *CEUR Workshop Proceedings*, pages 24–29. CEUR-WS.org, 2016. URL http://ceur-ws.org/Vol-1685/paper4.pdf. 203

Joan Capdevila, Marta Arias, and Argimiro Arratia. Geosrs: A hybrid social recommender system for geolocated data. *Inf. Syst.*, 57:111–128, 2016. doi: 10.1016/j.is.2015.10.003. URL https://doi.org/10.1016/j.is.2015.10.003. 203

Jian Li, Guanjun Liu, Changjun Jiang, and ChunGang Yan. A hybrid method of recommending pois based on context and personal preference confidence. In Ashiq Anjum and Xinghui Zhao, editors, *Proceedings of the 3rd IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, BDCAT 2016, Shanghai, China, December 6-9, 2016*, pages 287–292. ACM, 2016c. doi: 10.1145/3006299.3006330. URL https://doi.org/10.1145/3006299.3006330. 203

Md. Ahsan Habib, Md. Abdur Rakib, and Muhammad Abdul Hasan. Location, time, and preference aware restaurant recommendation method. In *2016 19th International Conference on Computer and Information Technology (ICCIT)*, pages 315–320, Dec 2016. doi: 10.1109/ICCITECHN.2016.7860216. 203

Zijun Yao, Yanjie Fu, Bin Liu, Yanchi Liu, and Hui Xiong. POI recommendation: A temporal matching between POI popularity and user regularity. In Francesco Bonchi, Josep Domingo-Ferrer, Ricardo A. Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu, editors, *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, pages 549–558. IEEE, 2016. doi: 10.1109/ICDM.2016.0066. URL https://doi.org/10.1109/ICDM.2016.0066. 203

Basma H. Albanna, Mahmoud Attia Sakr, Sherin M. Moussa, and Ibrahim F. Moawad. Interest aware location-based recommender system using geo-tagged social media. *ISPRS Int. J. Geo-Information*, 5(12):245, 2016. doi: 10.3390/ijgi5120245. URL https://doi.org/10.3390/ijgi5120245. 203

Xiaoyan Zhu and Ripei Hao. Context-aware location recommendations with tensor factorization. In *2016 IEEE/CIC International Conference on Communications in China, ICCC 2016, Chengdu, China, July 27-29, 2016*, pages 1–6. IEEE, 2016. doi: 10.1109/ICCChina.2016.7636832. URL https://doi.org/10.1109/ICCChina.2016.7636832. 203

Jialiang Chen, Xin Li, William K. Cheung, and Kan Li. Effective successive POI recommendation inferred with individual behavior and group preference. *Neurocomputing*, 210:174–184, 2016c. doi: 10.1016/j.neucom.2015.10.146. URL https://doi.org/10.1016/j.neucom.2015.10.146. 203

Zhengwu Yuan and Haiguang Li. Location recommendation algorithm based on temporal and geographical similarity in location-based social networks. In *2016 12th World Congress on Intelligent Control and Automation (WCICA)*, pages 1697–1702, June 2016. doi: 10.1109/WCICA.2016.7578804. 203

Luepol Pipanmaekaporn and Suwatchai Kamonsantiroj. Mining semantic location history for collaborative POI recommendation in online social networks. In Irfan Awan and Muhammad Younas, editors, *2nd International Conference on Open and Big Data, OBD 2016, Vienna, Austria, August 22-24, 2016*, pages 31–38. IEEE Computer Society, 2016. doi: 10.1109/OBD.2016.12. URL https://doi.org/10.1109/OBD.2016.12. 203

Budsabawan Jueajan, Kanittha Naleg, Luepol Pipanmekaporn, and Suwatchai Kamolsantiroj. Development of location-aware place recommendation system on android smart phones. In *2016 Fifth ICT International Student Project Conference (ICT-ISPC)*, pages 125–128, May 2016. doi: 10.1109/ICT-ISPC.2016.7519252. 203

Bahaeddin Eravci, Neslihan Bulut, Çagri Etemoglu, and Hakan Ferhatosmanoglu. Location recommendations for new businesses using check-in data. In Carlotta Domeniconi, Francesco Gullo, Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu, editors, *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain*, pages 1110–1117. IEEE Computer Society, 2016. doi: 10.1109/ICDMW.2016.0160. URL https://doi.org/10.1109/ICDMW.2016.0160. 203

Hanbing Zhang, Yan Yang, and Zhaogong Zhang. CTS: combine temporal influence and spatial influence for time-aware POI recommendation. In Wanxiang Che, Qilong Han, Hongzhi Wang, Weipeng Jing, Shaoliang Peng, Junyu Lin, Guanglu Sun, Xianhua Song, Hongtao Song, and Zeguang Lu, editors, *Social Computing - Second International Conference of Young Computer Scientists, Engineers and Educators, ICYCSEE 2016, Harbin, China, August 20-22, 2016, Proceedings, Part I*, volume 623 of *Communications in Computer and Information Science*, pages 272–286. Springer, 2016a. doi: 10.1007/978-981-10-2053-7_25. URL https://doi.org/10.1007/978-981-10-2053-7_25. 203

Hao Guo, Xin Li, Ming He, Xiangyu Zhao, Guiquan Liu, and Guandong Xu. Cosolorec: Joint factor model with content, social, location for heterogeneous point-of-interest recommendation. In Franz Lehner and Nora Fteimi, editors, *Knowledge Science, Engineering and Management - 9th International Conference, KSEM 2016, Passau, Germany, October 5-7, 2016, Proceedings*, volume 9983 of *Lecture Notes in Computer Science*, pages 613–627, 2016. doi: 10.1007/978-3-319-47650-6_48. URL https://doi.org/10.1007/978-3-319-47650-6_48. 203

Madhuri Debnath, Praveen Kumar Tripathi, and Ramez Elmasri. Preference-aware successive POI recommendation with spatial and temporal influence. In Emma S. Spiro and Yong-Yeol Ahn, editors, *Social Informatics - 8th International Conference, SocInfo 2016, Bellevue, WA, USA, November 11-14, 2016, Proceedings, Part I*, volume 10046 of *Lecture Notes in Computer Science*, pages 347–360, 2016. doi: 10.1007/978-3-319-47880-7_21. URL https://doi.org/10.1007/978-3-319-47880-7_21. 203

Huayu Li, Richang Hong, Zhiang Wu, and Yong Ge. A spatial-temporal probabilistic matrix factorization model for point-of-interest recommendation. In Sanjay Chawla Venkatasubramanian and Wagner Meira Jr., editors, *Proceedings of the 2016 SIAM International Conference on Data Mining, Miami, Florida, USA, May 5-7, 2016*, pages 117–125. SIAM, 2016d. doi: 10.1137/1.9781611974348.14. URL https://doi.org/10.1137/1.9781611974348.14. 203

Xiefeng Xu, Pengpeng Zhao, Guanfeng Liu, Caidong Gu, Jiajie Xu, Jian Wu, and Zhiming Cui. A hybrid method for POI recommendation: Combining check-in count, geographical information and reviews. In Feifei Li, Kyuseok Shim, Kai Zheng, and Guanfeng Liu, editors, *Web Technologies and Applications - 18th Asia-Pacific Web Conference, APWeb 2016, Suzhou, China, September 23-25, 2016. Proceedings, Part II*, volume 9932 of *Lecture Notes in Computer Science*, pages 162–173. Springer, 2016a. doi: 10.1007/978-3-319-45817-5_13. URL https://doi.org/10.1007/978-3-319-45817-5_13. 203

## REFERENCES

Stathis Maroulis, Ioannis Boutsis, and Vana Kalogeraki. Context-aware point of interest recommendation using tensor factorization. In James Joshi, George Karypis, Ling Liu, Xiaohua Hu, Ronay Ak, Yinglong Xia, Weijia Xu, Aki-Hiro Sato, Sudarsan Rachuri, Lyle H. Ungar, Philip S. Yu, Rama Govindaraju, and Toyotaro Suzumura, editors, *2016 IEEE International Conference on Big Data, BigData 2016, Washington DC, USA, December 5-8, 2016*, pages 963–968. IEEE Computer Society, 2016. doi: 10.1109/BigData.2016.7840694. URL https://doi.org/10.1109/BigData.2016.7840694. 203

Shenglin Zhao, Michael R. Lyu, and Irwin King. Aggregated temporal tensor factorization model for point-of-interest recommendation. In Akira Hirose, Seiichi Ozawa, Kenji Doya, Kazushi Ikeda, Minho Lee, and Derong Liu, editors, *Neural Information Processing - 23rd International Conference, ICONIP 2016, Kyoto, Japan, October 16-21, 2016, Proceedings, Part III*, volume 9949 of *Lecture Notes in Computer Science*, pages 450–458, 2016b. doi: 10.1007/978-3-319-46675-0_49. URL https://doi.org/10.1007/978-3-319-46675-0_49. 203

Cong Zheng, Haihong E, Meina Song, and Junde Song. TGTM: temporal-geographical topic model for point-of-interest recommendation. In Shamkant B. Navathe, Weili Wu, Shashi Shekhar, Xiaoyong Du, X. Sean Wang, and Hui Xiong, editors, *Database Systems for Advanced Applications - 21st International Conference, DASFAA 2016, Dallas, TX, USA, April 16-19, 2016, Proceedings, Part I*, volume 9642 of *Lecture Notes in Computer Science*, pages 348–363. Springer, 2016. doi: 10.1007/978-3-319-32025-0_22. URL https://doi.org/10.1007/978-3-319-32025-0_22. 203

Guandong Xu, Bin Fu, and Yanhui Gu. Point-of-interest recommendations via a supervised random walk algorithm. *IEEE Intelligent Systems*, 31(1):15–23, 2016b. doi: 10.1109/MIS.2016.4. URL https://doi.org/10.1109/MIS.2016.4. 203

Gonzalo Rojas, Diego Seco, and Francisco Serrano. Boosting point-of-interest recommendation with multigranular time representations. *J. UCS*, 22(8):1148–1174, 2016. URL http://www.jucs.org/jucs_22_8/boosting_point_of_interest. 203

Mu-Yao Fang and Bi-Ru Dai. Power of bosom friends, POI recommendation by learning preference of close friends and similar users. In Sanjay Madria and Takahiro Hara, editors, *Big Data Analytics and Knowledge Discovery - 18th International Conference, DaWaK 2016, Porto, Portugal, September 6-8, 2016, Proceedings*, volume 9829 of *Lecture Notes in Computer Science*, pages 179–192. Springer, 2016. doi: 10.1007/978-3-319-43946-4_12. URL https://doi.org/10.1007/978-3-319-43946-4_12. 203

Da-Chuan Zhang, Mei Li, and Chang-Dong Wang. Point of interest recommendation with social and geographical influence. In James Joshi, George Karypis, Ling Liu, Xiaohua Hu, Ronay Ak, Yinglong Xia, Weijia Xu, Aki-Hiro Sato, Sudarsan Rachuri, Lyle H. Ungar, Philip S. Yu, Rama Govindaraju, and Toyotaro Suzumura, editors, *2016 IEEE International Conference on Big Data, BigData 2016, Washington DC, USA, December 5-8, 2016*, pages 1070–1075. IEEE Computer Society, 2016b. doi: 10.1109/BigData.2016.7840709. URL https://doi.org/10.1109/BigData.2016.7840709. 203

Jianmin Wang, Ruhuo Tan, Ri-Peng Zhang, and Fang You. A recommender system research based on location-based social networks. In Gabriele Meiselwitz, editor, *Social Computing and Social Media - 8th International Conference, SCSM 2016, Held as Part of HCI International 2016, Toronto, ON, Canada, July 17-22, 2016. Proceedings*, volume 9742 of *Lecture Notes in Computer Science*, pages 81–90. Springer, 2016. doi: 10.1007/978-3-319-39910-2_8. URL https://doi.org/10.1007/978-3-319-39910-2_8. 203

Rahul Katarya, Manika Ranjan, and Om Prakash Verma. Location based recommender system using enhanced random walk model. In *2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, pages 33–37, Dec 2016. doi: 10.1109/PDGC.2016.7913179. 203

Haiyan Guan, Hongyan Qian, and Yanchao Zhao. Location privacy protected recommendation system in mobile cloud. In Xingming Sun, Alex X. Liu, Han-Chieh Chao, and Elisa Bertino, editors, *Cloud Computing and Security - Second International Conference, ICCCS 2016, Nanjing, China, July 29-31, 2016, Revised Selected Papers, Part I*, volume 10039 of *Lecture Notes in Computer Science*, pages 409–420, 2016. doi: 10.1007/978-3-319-48671-0_36. URL https://doi.org/10.1007/978-3-319-48671-0_36. 203

Xiaoyan Zhu, Ripei Hao, Haotian Chi, and Xiaojiang Du. Personalized location recommendations with local feature awareness. In *2016 IEEE Global Communications Conference, GLOBECOM 2016, Washington, DC, USA, December 4-8, 2016*, pages 1–6. IEEE, 2016. doi: 10.1109/GLOCOM.2016.7842140. URL https://doi.org/10.1109/GLOCOM.2016.7842140. 203

Ruheng Lv, Yufeng Wang, Qun Jin, and Jianhua Ma. ELR-DC: an efficient recommendation scheme for location based social networks. In Xingang Liu, Tie Qiu, Bin Guo, Kaixuan Lu, Zhaolong Ning, Mianxiong Dong, and Yayong Li, editors, *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Chengdu, China, December 15-18, 2016*, pages 567–572. IEEE, 2016. doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2016.127. URL https://doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData.2016.127. 203

Carlos Rios, Silvia Noemi Schiaffino, and Daniela Lis Godoy. Selecting and weighting users in collaborative filtering-based poi recommendation. *Acta Polytechnica Hungarica*, 14(3):13–32, 2017. ISSN 1785-8860. 203

Pavlos Kefalas and Yannis Manolopoulos. A time-aware spatio-textual recommender system. *Expert Syst. Appl.*, 78:396–406, 2017. doi: 10.1016/j.eswa.2017.01.060. URL https://doi.org/10.1016/j.eswa.2017.01.060. 203

Jarana Manotumruksa, Craig Macdonald, and Iadh Ounis. A personalised ranking framework with multiple sampling criteria for venue recommendation. In Ee-Peng Lim, Marianne Winslett, Mark Sanderson, Ada Wai-Chee Fu, Jimeng Sun, J. Shane Culpepper, Eric Lo, Joyce C. Ho, Debora Donato, Rakesh Agrawal, Yu Zheng, Carlos Castillo, Aixin Sun, Vincent S. Tseng, and Chenliang Li, editors, *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 1469–1478. ACM, 2017b. doi: 10.1145/3132847.3132985. URL https://doi.org/10.1145/3132847.3132985. 203

Hai tao Zheng, Yingmin Zhou, Nan Liang, Xi Xiao, Arun Kumar Sangaiah, and Congzhi Zhao. Exploiting user mobility for time-aware poi recommendation in social networks. *IEEE Access*, pages 1–1, 2017. ISSN 2169-3536. doi: 10.1109/ACCESS.2017.2764074. 203

Hung-Yi Gau, Yi-Shu Lu, and Jiun-Long Huang. A grid-based successive point-of-interest recommendation method. In *2017 10th International Conference on Ubi-media Computing and Workshops (Ubi-Media)*, pages 1–6, Aug 2017. doi: 10.1109/UMEDIA.2017.8074153. 203

Bin Xia, Zhen Ni, Tao Li, Qianmu Li, and Qifeng Zhou. Vrer: Context-based venue recommendation using embedded space ranking SVM in location-based social network. *Expert Syst. Appl.*, 83:18–29, 2017b. doi: 10.1016/j.eswa.2017.04.020. URL https://doi.org/10.1016/j.eswa.2017.04.020. 203

Shuiqiao Yang, Guangyan Huang, Yang Xiang, Xiangmin Zhou, and Chi-Hung Chi. Modeling user preferences on spatiotemporal topics for point-of-interest recommendation. In Xiaoqing (Frank) Liu and Umesh Bellur, editors, *2017 IEEE International Conference on Services Computing, SCC 2017, Honolulu, HI, USA, June 25-30, 2017*, pages 204–211. IEEE Computer Society, 2017b. doi: 10.1109/SCC.2017.33. URL https://doi.org/10.1109/SCC.2017.33. 203

Shokirkhon Oppokhonov, Seyoung Park, and Isaac K. E. Ampomah. Current location-based next POI recommendation. In Amit P. Sheth, Axel Ngonga, Yin Wang, Elizabeth Chang, Dominik Slezak, Bogdan Franczyk, Rainer Alt, Xiaohui Tao, and Rainer Unland, editors, *Proceedings of the International Conference on Web Intelligence, Leipzig, Germany, August 23-26, 2017*, pages 831–836. ACM, 2017. doi: 10.1145/3106426.3106528. URL http://doi.acm.org/10.1145/3106426.3106528. 203

Bilian Chen, Shenbao Yu, Jing Tang, Mengda He, and Yifeng Zeng. Using function approximation for personalized point-of-interest recommendation. *Expert Syst. Appl.*, 79: 225–235, 2017. doi: 10.1016/j.eswa.2017.01.037. URL https://doi.org/10.1016/j.eswa.2017.01.037. 203

Hao Wang, Yanmei Fu, Qinyong Wang, Hongzhi Yin, Changying Du, and Hui Xiong. A location-sentiment-aware recommender system for both home-town and out-of-town users. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 1135–1143. ACM, 2017b. doi: 10.1145/3097983.3098122. URL http://doi.acm.org/10.1145/3097983.3098122. 204

Qing Guo, Zhu Sun, Jie Zhang, Qi Chen, and Yin-Leng Theng. Aspect-aware point-of-interest recommendation with geo-social influence. In Mária Bieliková, Eelco Herder, Federica Cena, and Michel C. Desmarais, editors, *Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization, UMAP 2017, Bratislava, Slovakia, July 09 - 12, 2017*, pages 17–22. ACM, 2017. doi: 10.1145/3099023.3099066. URL https://doi.org/10.1145/3099023.3099066. 204

Heba M. Wagih, Hoda M. O. Mokhtar, and Samy S. Ghoniemy. Location recommendation based on social trust. In Hai Zhuge and Xiaoping Sun, editors, *13th International Conference on Semantics, Knowledge and Grids, SKG 2017, Beijing, China, August 13-14, 2017*, pages 50–55. IEEE, 2017. doi: 10.1109/SKG.2017.00017. URL https://doi.org/10.1109/SKG.2017.00017. 204

Wenjing Luan, Guanjun Liu, Changjun Jiang, and Liang Qi. Partition-based collaborative tensor factorization for POI recommendation. *IEEE CAA J. Autom. Sinica*, 4 (3):437–446, 2017. doi: 10.1109/JAS.2017.7510538. URL https://doi.org/10.1109/JAS.2017.7510538. 204

Yuankai Ying, Ling Chen, and Gencai Chen. A temporal-aware POI recommendation system using context-aware tensor decomposition and weighted HITS. *Neurocomputing*, 242:195–205, 2017. doi: 10.1016/j.neucom.2017.02.067. URL https://doi.org/10.1016/j.neucom.2017.02.067. 204

Logesh Ravi and V. Subramaniyaswamy. Learning recency and inferring associations in location based social network for emotion induced point-of-interest recommendation. *J. Inf. Sci. Eng.*, 33(6):1629–1647, 2017b. URL http://jise.iis.sinica.edu.tw/JISESearch/pages/View/PaperView.jsf?keyId=159_2101. 204

Jungkyu Han and Hayato Yamana. Familiarity-aware POI recommendation in urban neighborhoods. *JIP*, 25:386–396, 2017. doi: 10.2197/ipsjjip.25.386. URL https://doi.org/10.2197/ipsjjip.25.386. 204

Shenglin Zhao, Irwin King, and Michael R. Lyu. Geo-pairwise ranking matrix factorization model for point-of-interest recommendation. In Derong Liu, Shengli Xie, Yuanqing Li, Dongbin Zhao, and El-Sayed M. El-Alfy, editors, *Neural Information Processing - 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part V*, volume 10638 of *Lecture Notes in Computer Science*, pages 368–377. Springer, 2017c. doi: 10.1007/978-3-319-70139-4_37. URL https://doi.org/10.1007/978-3-319-70139-4_37. 204

Seyyed Mohammadreza Rahimi, Xin Wang, and Behrouz Far. Behavior-based location recommendation on location-based social networks. In Jinho Kim, Kyuseok Shim, Longbing Cao, Jae-Gil Lee, Xuemin Lin, and Yang-Sae Moon, editors, *Advances in Knowledge Discovery and Data Mining - 21st Pacific-Asia Conference, PAKDD 2017, Jeju, South Korea, May 23-26, 2017, Proceedings, Part II*, volume 10235 of *Lecture Notes in Computer Science*, pages 273–285, 2017. doi: 10.1007/978-3-319-57529-2_22. URL https://doi.org/10.1007/978-3-319-57529-2_22. 204

Man-Rui Li, Ling Huang, and Chang-Dong Wang. Geographical and overlapping community modeling based on business circles for POI recommendation. In Yi Sun, Huchuan Lu, Lihe Zhang, Jian Yang, and Hua Huang, editors, *Intelligence Science and Big Data Engineering - 7th International Conference, IScIDE 2017, Dalian, China, September 22-23, 2017, Proceedings*, volume 10559 of *Lecture Notes in Computer Science*, pages 665–675. Springer, 2017c. doi: 10.1007/978-3-319-67777-4_60. URL https://doi.org/10.1007/978-3-319-67777-4_60. 204

Yi-Ning Xu, Lei Xu, Ling Huang, and Chang-Dong Wang. Social and content based collaborative filtering for point-of-interest recommendations. In Derong Liu, Shengli Xie, Yuanqing Li, Dongbin Zhao, and El-Sayed M. El-Alfy, editors, *Neural Information Processing - 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part V*, volume 10638 of *Lecture Notes in Computer Science*, pages 46–56. Springer, 2017a. doi: 10.1007/978-3-319-70139-4_5. URL https://doi.org/10.1007/978-3-319-70139-4_5. 204

Jun Zeng, Feng Li, Junhao Wen, and Wei Zhou. A point of interest recommendation approach by fusing geographical and reputation influence on location based social networks. In Imed Romdhani, Lei Shu, Takahiro Hara, Zhangbing Zhou, Timothy J. Gordon, and Deze Zeng, editors, *Collaborative Computing: Networking, Applications and Worksharing - 13th International Conference, CollaborateCom 2017, Edinburgh, UK, December 11-13, 2017, Proceedings*, volume 252 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 232–242. Springer, 2017. doi: 10.1007/978-3-030-00916-8_22. URL https://doi.org/10.1007/978-3-030-00916-8_22. 204

Qianfang Xu, Jiachun Wang, and Bo Xiao. Personalized location recommendation for location-based social networks. In *2017 IEEE/CIC International Conference on Communications in China, ICCC 2017, Qingdao, China, October 22-24, 2017*, pages 1–6. IEEE, 2017b. doi: 10.1109/ICCChina.2017.8330459. URL https://doi.org/10.1109/ICCChina.2017.8330459. 204

Yangkai Shi and Wenjun Jiang. Point-of-interest recommendations: Capturing the geographical influence from local trajectories. In *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC), Guangzhou, China, December 12-15, 2017*, pages 1122–1129. IEEE, 2017. doi: 10.1109/ISPA/IUCC.2017.00169. URL https://doi.org/10.1109/ISPA/IUCC.2017.00169. 204

Lye Guang Xing, Ileladewa Adeoye Abiodun, Cheng Wai Khuen, and Tan Teik Boon. A personalized recommendation framework with user trajectory analysis applied in location-based social network (lbsn). In *2017 IEEE 3rd International Conference on Engineering Technologies and Social Sciences (ICETSS)*, pages 1–6, 2017. 204

Deepali J. Erande and Archana Chaugule. Personalized point of interest recommendation using check-in history and friend's interest. In *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, pages 1–5, 2017. 204

Yuanyi Chen, Zengwei Zheng, Lin Sun, Dan Chen, and Minyi Guo. Fine-gained location recommendation based on user

## REFERENCES

textual reviews in lbsns. In Shijian Li, editor, *Green, Pervasive, and Cloud Computing - 13th International Conference, GPC 2018, Hangzhou, China, May 11-13, 2018, Revised Selected Papers*, volume 11204 of *Lecture Notes in Computer Science*, pages 196–211. Springer, 2018b. doi: 10.1007/978-3-030-15093-8_14. URL https://doi.org/10.1007/978-3-030-15093-8_14. 204

Shudong Liu and Lei Wang. A self-adaptive point-of-interest recommendation algorithm based on a multi-order markov model. *Future Generation Comp. Syst.*, 89:506–514, 2018. doi: 10.1016/j.future.2018.07.008. URL https://doi.org/10.1016/j.future.2018.07.008. 204

Lianggui Liu, Wei Li, Lingmin Wang, and Huiling Jia. PCRM: increasing POI recommendation accuracy in location-based social networks. *TIIS*, 12(11):5344–5356, 2018. doi: 10.3837/tiis.2018.11.010. URL https://doi.org/10.3837/tiis.2018.11.010. 204

Jean-Benoît Griesner, Talel Abdessalem, Hubert Naacke, and Pierre Dosne. Algeospf: A hierarchical factorization model for POI recommendation. In Ulrik Brandes, Chandan Reddy, and Andrea Tagarelli, editors, *IEEE/ACM 2018 International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2018, Barcelona, Spain, August 28-31, 2018*, pages 87–90. IEEE Computer Society, 2018. doi: 10.1109/ASONAM.2018.8508249. URL https://doi.org/10.1109/ASONAM.2018.8508249. 204

Ziqing Zhu, Jiuxin Cao, and Chenghao Weng. Location-time-sociality aware personalized tourist attraction recommendation in LBSN. In Weiming Shen, Junzhou Luo, Jean-Paul A. Barthès, Fang Dong, Jinghui Zhang, and Haibin Zhu, editors, *22nd IEEE International Conference on Computer Supported Cooperative Work in Design, CSCWD 2018, Nanjing, China, May 9-11, 2018*, pages 636–641. IEEE, 2018b. doi: 10.1109/CSCWD.2018.8465179. URL https://doi.org/10.1109/CSCWD.2018.8465179. 204

Xiangguo Zhao, Zhongyu Ma, and Zhen Zhang. A novel recommendation system in location-based social networks using distributed ELM. *Memetic Computing*, 10(3):321–331, 2018c. doi: 10.1007/s12293-017-0227-4. URL https://doi.org/10.1007/s12293-017-0227-4. 204

Shuning Xing, Fangai Liu, Xiaohui Zhao, and Tianlai Li. Points-of-interest recommendation based on convolution matrix factorization. *Appl. Intell.*, 48(8):2458–2469, 2018. doi: 10.1007/s10489-017-1103-0. URL https://doi.org/10.1007/s10489-017-1103-0. 204

Hanane Amirat, Abderrahim Benslimane, Philippe Fournier-Viger, and Nasreddine Lagraa. Locrec: Rule-based successive location recommendation in LBSN. In *2018 IEEE International Conference on Communications, ICC 2018, Kansas City, MO, USA, May 20-24, 2018*, pages 1–6. IEEE, 2018. doi: 10.1109/ICC.2018.8422183. URL https://doi.org/10.1109/ICC.2018.8422183. 204

Hao Wang, Wentao Ouyang, Huawei Shen, and Xueqi Cheng. ULE: learning user and location embeddings for POI recommendation. In *Third IEEE International Conference on Data Science in Cyberspace, DSC 2018, Guangzhou, China, June 18-21, 2018*, pages 99–106. IEEE, 2018c. doi: 10.1109/DSC.2018.00023. URL https://doi.org/10.1109/DSC.2018.00023. 204

Guoqiong Liao, Shan Jiang, Zhiheng Zhou, Changxuan Wan, and Xiping Liu. POI recommendation of location-based social networks using tensor factorization. In *19th IEEE International Conference on Mobile Data Management, MDM 2018, Aalborg, Denmark, June 25-28, 2018*, pages 116–124. IEEE Computer Society, 2018. doi: 10.1109/MDM.2018.00028. URL https://doi.org/10.1109/MDM.2018.00028. 204

Tao Xu, Yutao Ma, and Qian Wang. Cross-urban point-of-interest recommendation for non-natives. *Int. J. Web Service Res.*, 15(3):82–102, 2018b. doi: 10.4018/IJWSR.2018070105. URL https://doi.org/10.4018/IJWSR.2018070105. 204

Jinxin Liu, Xu Jiao, Youzhi Jin, Xinyu Liu, and Li Liu. Research and implementation of poi recommendation system integrating temporal feature. In *2018 IEEE 3rd International Conference on Big Data Analysis (ICBDA)*, pages 287–292, March 2018. doi: 10.1109/ICBDA.2018.8367694. 204

Junfei Wang, Darshan Bagul, Jun Chu, Lu Meng, and Sargur Srihari. Mining place-time affinity to improve poi recommendation. In *2018 International Conference on Information and Computer Technologies (ICICT)*, pages 22–26, March 2018. doi: 10.1109/INFOCT.2018.8356834. 204

Lei Guo, Haoran Jiang, and Xinhua Wang. Location regularization-based POI recommendation in location-based social networks. *Information*, 9(4):85, 2018. doi: 10.3390/info9040085. URL https://doi.org/10.3390/info9040085. 204

Chang Su, Ning Li, and Xian-Zhong Xie. Point-of-interest recommendation based on spatial clustering in lbsn. In *2018 4th Annual International Conference on Network and Information Systems for Computers (ICNISC)*, pages 7–12, April 2018. doi: 10.1109/ICNISC.2018.00011. 204

Ramesh Baral and Tao Li. Exploiting the roles of aspects in personalized POI recommender systems. *Data Min. Knowl. Discov.*, 32(2):320–343, 2018. doi: 10.1007/s10618-017-0537-7. URL https://doi.org/10.1007/s10618-017-0537-7. 204

Jingyuan Gao and Yan Yang. ABPR- A new way of point-of-interest recommendation via geographical and category influence. In Qinglei Zhou, Qiguang Miao, Hongzhi Wang, Wei Xie, Yan Wang, and Zeguang Lu, editors, *Data Science - 4th International Conference of Pioneering Computer Scientists, Engineers and Educators, ICPCSEE 2018, Zhengzhou, China, September 21-23, 2018, Proceedings, Part II*, volume 902 of *Communications in Computer and Information Science*, pages 96–107. Springer, 2018. doi: 10.1007/978-981-13-2206-8_9. URL https://doi.org/10.1007/978-981-13-2206-8_9. 204

Elahe Naserianhanzaei, Xinheng Wang, and Keshav P. Dahal. APPR: additive personalized point-of-interest recommendation. In *IEEE Global Communications Conference, GLOBECOM 2018, Abu Dhabi, United Arab Emirates, December 9-13, 2018*, pages 1–7. IEEE, 2018. doi: 10.1109/GLOCOM.2018.8647203. URL https://doi.org/10.1109/GLOCOM.2018.8647203. 204

Haifeng Zhu, Pengpeng Zhao, Zhixu Li, Jiajie Xu, Lei Zhao, and Victor S. Sheng. Exploiting implicit social relationship for point-of-interest recommendation. In Yi Cai, Yoshiharu Ishikawa, and Jianliang Xu, editors, *Web and Big Data - Second International Joint Conference, APWeb-WAIM 2018, Macau, China, July 23-25, 2018, Proceedings, Part II*, volume 10988 of *Lecture Notes in Computer Science*, pages 280–297. Springer, 2018c. doi: 10.1007/978-3-319-96893-3_21. URL https://doi.org/10.1007/978-3-319-96893-3_21. 204

Siyuan Zhang and Hong Cheng. Exploiting context graph attention for POI recommendation in location-based social networks. In Jian Pei, Yannis Manolopoulos, Shazia W. Sadiq, and Jianxin Li, editors, *Database Systems for Advanced Applications - 23rd International Conference, DASFAA 2018, Gold Coast, QLD, Australia, May 21-24, 2018, Proceedings, Part I*, volume 10827 of *Lecture Notes in Computer Science*, pages 83–99. Springer, 2018. doi: 10.1007/978-3-319-91452-7_6. URL https://doi.org/10.1007/978-3-319-91452-7_6. 204

Alireza Pourali, Fattane Zarrinkalam, and Ebrahim Bagheri. Point-of-interest recommendation using heterogeneous link prediction. In Michael H. Böhlen, Reinhard Pichler, Norman May, Erhard Rahm, Shan-Hung Wu, and Katja Hose, editors, *Proceedings of the 21th International Conference on Extending Database Technology, EDBT 2018, Vienna, Austria, March 26-29, 2018*, pages 481–484. OpenProceedings.org, 2018. doi: 10.5441/002/edbt.2018.52. URL https://doi.org/10.5441/002/edbt.2018.52. 204

Ruifeng Ding, Zhenzhong Chen, and Xiaolei Li. Spatial-temporal distance metric embedding for time-specific POI recommendation. *IEEE Access*, 6:67035–67045, 2018b. doi: 10.1109/ACCESS.2018.2869994. URL https://doi.org/10.1109/ACCESS.2018.2869994. 204

Zhiyuan Zhang and Yun Liu. A list-wise matrix factorization based poi recommendation by fusing multi-tag, social and geographical influences. *Journal of Internet Technology*, 19 (1):127–136, 2018. URL https://jit.ndhu.edu.tw/article/view/1632. 204

Rong Yang, Xiaofeng Han, and Xingzhong Zhang. A multi-factor recommendation algorithm for POI recommendation. In Xiaofeng Meng, Ruixuan Li, Kanliang Wang, Baoning Niu, Xin Wang, and Gansen Zhao, editors, *Web Information Systems and Applications - 15th International Conference, WISA 2018, Taiyuan, China, September 14-15, 2018, Proceedings*, volume 11242 of *Lecture Notes in Computer Science*, pages 445–454. Springer, 2018. doi: 10.1007/978-3-030-02934-0_41. URL https://doi.org/10.1007/978-3-030-02934-0_41. 205

Omer Tal and Yang Liu. TCENR: A hybrid neural recommender for location based social networks. In Hanghang Tong, Zhenhui Jessie Li, Feida Zhu, and Jeffrey Yu, editors, *2018 IEEE International Conference on Data Mining Workshops, ICDM Workshops, Singapore, Singapore, November 17-20, 2018*, pages 1186–1191. IEEE, 2018. doi: 10.1109/ICDMW.2018.00170. URL https://doi.org/10.1109/ICDMW.2018.00170. 205

Akash Gupta, Neha Tandon, and Sonia Khetarpaul. Influence-time-proximity driven locations recommendation model: An integrated approach. In *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), Kochi, India, October 17-20, 2019*, pages 1381–1386. IEEE, 2019. doi: 10.1109/TENCON.2019.8929590. URL https://doi.org/10.1109/TENCON.2019.8929590. 205

Xi Wang, Iadh Ounis, and Craig Macdonald. Comparison of sentiment analysis and user ratings in venue recommendation. In Leif Azzopardi, Benno Stein, Norbert Fuhr, Philipp Mayr, Claudia Hauff, and Djoerd Hiemstra, editors, *Advances in Information Retrieval - 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14-18, 2019, Proceedings, Part I*, volume 11437 of *Lecture Notes in Computer Science*, pages 215–228. Springer, 2019. doi: 10.1007/978-3-030-15712-8_14. URL https://doi.org/10.1007/978-3-030-15712-8_14. 205

Guoming Zhang, Lianyong Qi, Xuyun Zhang, Xiaolong Xu, and Wanchun Dou. Context-aware point-of-interest recommendation algorithm with interpretability. In Xinheng Wang, Honghao Gao, Muddesar Iqbal, and Geyong Min, editors, *Collaborative Computing: Networking, Applications and Worksharing - 15th EAI International Conference, CollaborateCom 2019, London, UK, August 19-22, 2019, Proceedings*, volume 292 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 745–759. Springer, 2019b. doi: 10.1007/978-3-030-30146-0_50. URL https://doi.org/10.1007/978-3-030-30146-0_50. 205

Lei Guo, Yufei Wen, and Fangai Liu. Location perspective-based neighborhood-aware POI recommendation in location-based social networks. *Soft Comput.*, 23(22): 11935–11945, 2019a. doi: 10.1007/s00500-018-03748-9. URL https://doi.org/10.1007/s00500-018-03748-9. 205

Xu Jiao, Yingyuan Xiao, Wenguang Zheng, Hongya Wang, and Ching-Hsien Hsu. A novel next new point-of-interest recommendation system based on simulated user travel decision-making process. *Future Generation Comp. Syst.*, 100:982–993, 2019a. doi: 10.1016/j.future.2019.05.065. URL https://doi.org/10.1016/j.future.2019.05.065. 205

Jiajun Zhou, Bo Liu, Yaofeng Chen, and Fuqiang Lin. Ufc: A unified poi recommendation framework. *Arabian Journal for Science and Engineering*, 44:9321–9332, 2019a. doi: https://doi.org/10.1007/s13369-019-04011-5. 205

Guoqiang Zhou, Shuai Zhang, Yi Fan, Jingjin Li, Wenbo Yao, and Hongfang Liu. Recommendations based on user effective point-of-interest path. *Int. J. Machine Learning & Cybernetics*, 10(10):2887–2899, 2019b. doi: 10.1007/s13042-018-00910-5. URL https://doi.org/10.1007/s13042-018-00910-5. 205

Ramesh Baral, S. Sitharama Iyengar, Xiaolong Zhu, Tao Li, and Pawel Sniatala. Hirecs: A hierarchical contextual location recommendation system. *IEEE Trans. Comput. Social Systems*, 6(5):1020–1037, 2019. doi: 10.1109/TCSS.2019.2938239. URL https://doi.org/10.1109/TCSS.2019.2938239. 205

Hossein A. Rahmani, Mohammad Aliannejadi, Rasoul Mirzaei Zadeh, Mitra Baratchi, Mohsen Afsharchi, and Fabio Crestani. Category-aware location embedding for point-of-interest recommendation. In Yi Fang, Yi Zhang, James Allan, Krisztian Balog, Ben Carterette, and Jiafeng Guo, editors, *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR 2019, Santa Clara, CA, USA, October 2-5, 2019*, pages 173–176. ACM, 2019. doi: 10.1145/3341981.3344240. URL https://doi.org/10.1145/3341981.3344240. 205

Zhibin Zhang, Cong Zou, Ruifeng Ding, and Zhenzhong Chen. VCG: exploiting visual contents and geographical influence for point-of-interest recommendation. *Neurocomputing*, 357:53–65, 2019c. doi: 10.1016/j.neucom.2019.04.079. URL https://doi.org/10.1016/j.neucom.2019.04.079. 205

Pei-Yi Hao, Weng-Hang Cheang, and Jung-Hsien Chiang. Real-time event embedding for POI recommendation. *Neurocomputing*, 349:1–11, 2019. doi: 10.1016/j.neucom.2019.04.022. URL https://doi.org/10.1016/j.neucom.2019.04.022. 205

Wenjie Cai, Yufeng Wang, Ruheng Lv, and Qun Jin. An efficient location recommendation scheme based on clustering and data fusion. *Computers & Electrical Engineering*, 77:289–299, 2019. doi: 10.1016/j.compeleceng.2019.06.006. URL https://doi.org/10.1016/j.compeleceng.2019.06.006. 205

Junjie Yin, Yun Li, Zheng Liu, Jian Xu, Bin Xia, and Qianmu Li. ADPR: an attention-based deep learning point-of-interest recommendation framework. In *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*, pages 1–8. IEEE, 2019. doi: 10.1109/IJCNN.2019.8852309. URL https://doi.org/10.1109/IJCNN.2019.8852309. 205

Sein Jang, Jeong-Hun Kim, and Aziz Nasridinov. Flexible POI recommendation based on user situation. In *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), iThings/GreenCom/CPSCom/SmartData 2019, Atlanta, GA, USA, July 14-17, 2019*, pages 1257–1260. IEEE, 2019. doi: 10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00211. URL https://doi.org/10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00211. 205

Tongcun Liu, Jianxin Liao, Zhigen Wu, Yulong Wang, and Jingyu Wang. A geographical-temporal awareness hierarchical attention network for next point-of-interest recommendation. In Abdulmotaleb El-Saddik, Alberto Del Bimbo, Zhongfei Zhang, Alexander G. Hauptmann, K. Selçuk Candan, Marco Bertini, Lexing Xie, and Xiao-Yong Wei, editors, *Proceedings of the 2019 on International Conference on Multimedia Retrieval, ICMR 2019, Ottawa, ON, Canada, June 10-13, 2019*, pages 7–15. ACM, 2019a. doi: 10.1145/3323873.3325024. URL https://doi.org/10.1145/3323873.3325024. 205

Jian Li, Guanjun Liu, Chungang Yan, and Changjun Jiang. LORI: A learning-to-rank-based integration method of location recommendation. *IEEE Trans. Comput. Social Systems*, 6(3):430–440, 2019b. doi: 10.1109/TCSS.2019.2907563. URL https://doi.org/10.1109/TCSS.2019.2907563. 205

# REFERENCES

Guanhua Zhan, Jian Xu, Zhifeng Huang, Qiang Zhang, Ming Xu, and Ning Zheng. A semantic sequential correlation based LSTM model for next POI recommendation. In *20th IEEE International Conference on Mobile Data Management, MDM 2019, Hong Kong, SAR, China, June 10-13, 2019*, pages 128–137. IEEE, 2019. doi: 10.1109/MDM.2019.00-65. URL https://doi.org/10.1109/MDM.2019.00-65. 205

Yi-Shu Lu, Wen-Yueh Shih, Hung-Yi Gau, Kuan-Chieh Chung, and Jiun-Long Huang. On successive point-of-interest recommendation. *World Wide Web*, 22(3):1151–1173, 2019. doi: 10.1007/s11280-018-0599-5. URL https://doi.org/10.1007/s11280-018-0599-5. 205

Xu Jiao, Yingyuan Xiao, Wenguang Zheng, Hongya Wang, and Youzhi Jin. R2SIGTP: a novel real-time recommendation system with integration of geography and temporal preference for next point-of-interest. In Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia, editors, *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 3560–3563. ACM, 2019b. doi: 10.1145/3308558.3314120. URL https://doi.org/10.1145/3308558.3314120. 205

X. Yu, X. Li, J. Li, and K. Gai. A geographical behavior-based point-of-interest recommendation. In *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, pages 166–171, 2019. 205

Yang Li, Yadan Luo, Zheng Zhang, Shazia W. Sadiq, and Peng Cui. Context-aware attention-based data augmentation for POI recommendation. In *35th IEEE International Conference on Data Engineering Workshops, ICDE Workshops 2019, Macao, China, April 8-12, 2019*, pages 177–184. IEEE, 2019c. doi: 10.1109/ICDEW.2019.00-14. URL https://doi.org/10.1109/ICDEW.2019.00-14. 205

Xingxin Liu, Xingyu Huang, Yue Wang, and Lin Zhang. Point-of-interest category recommendation based on group mobility modeling. In *Proceedings of the 24th International Conference on Intelligent User Interfaces: Companion, Marina del Ray, CA, USA, March 16-20, 2019*, pages 39–40. ACM, 2019b. doi: 10.1145/3308557.3308670. URL https://doi.org/10.1145/3308557.3308670. 205

Shuning Xing, Fang'ai Liu, Qianqian Wang, Xiaohui Zhao, and Tianlai Li. Content-aware point-of-interest recommendation based on convolutional neural network. *Appl. Intell.*, 49(3):858–871, 2019. doi: 10.1007/s10489-018-1276-1. URL https://doi.org/10.1007/s10489-018-1276-1. 205

Xu Yang, Billy Zimba, Tingting Qiao, Keyan Gao, and Xiaoya Chen. Exploring iot location information to perform point of interest recommendation engine: Traveling to a new geographical region. *Sensors*, 19(5):992, 2019. doi: 10.3390/s19050992. URL https://doi.org/10.3390/s19050992. 205

Song Chuang, Wen Junhao, and Li Shun. Personalized poi recommendation based on check-in data and geographical-regional influence. In *Proceedings of the 3rd International Conference on Machine Learning and Soft Computing*, ICMLSC 2019, pages 128–133, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366120. doi: 10.1145/3310986.3311034. URL https://doi.org/10.1145/3310986.3311034. 205

Mingxin Gan and Ling Gao. Discovering memory-based preferences for POI recommendation in location-based social networks. *ISPRS Int. J. Geo-Information*, 8(6):279, 2019. doi: 10.3390/ijgi8060279. URL https://doi.org/10.3390/ijgi8060279. 205

Qing Guo, Zhu Sun, Jie Zhang, and Yin-Leng Theng. Modeling heterogeneous influences for point-of-interest recommendation in location-based social networks. In Maxim Bakaev, Flavius Frasincar, and In-Young Ko, editors, *Web Engineering - 19th International Conference, ICWE 2019, Daejeon, South Korea, June 11-14, 2019, Proceedings*, volume 11496 of *Lecture Notes in Computer Science*, pages 72–80. Springer, 2019b. doi: 10.1007/978-3-030-19274-7_6. URL https://doi.org/10.1007/978-3-030-19274-7_6. 205

Lei Guo, Haoran Jiang, Xiyu Liu, and Changming Xing. Network embedding-aware point-of-interest recommendation in location-based social networks. *Complexity*, 2019: 3574194:1–3574194:18, 2019c. doi: 10.1155/2019/3574194. URL https://doi.org/10.1155/2019/3574194. 205

Jinghua Zhu and Xu Guo. Deep neural model for point-of-interest recommendation fused with graph embedding representation. In Edoardo S. Biagioni, Yao Zheng, and Siyao Cheng, editors, *Wireless Algorithms, Systems, and Applications - 14th International Conference, WASA 2019, Honolulu, HI, USA, June 24-26, 2019, Proceedings*, volume 11604 of *Lecture Notes in Computer Science*, pages 495–506. Springer, 2019. doi: 10.1007/978-3-030-23597-0_40. URL https://doi.org/10.1007/978-3-030-23597-0_40. 205

Yangyang Xu, Xuefei Li, Jing Li, Chunzhi Wang, Rong Gao, and Yonghong Yu. SSSER: spatiotemporal sequential and social embedding rank for successive point-of-interest recommendation. *IEEE Access*, 7:156804–156823, 2019. doi: 10.1109/ACCESS.2019.2950061. URL https://doi.org/10.1109/ACCESS.2019.2950061. 205

Seyyed Mohammadreza Rahimi, Behrouz Far, and Xin Wang. Behavior-based location recommendation on location-based social networks. *GeoInformatica*, 24(3):477–504, 2020. doi: 10.1007/s10707-019-00360-3. URL https://doi.org/10.1007/s10707-019-00360-3. 205

Khoa D. Doan, Guolei Yang, and Chandan K. Reddy. An attentive spatio-temporal neural model for successive point of interest recommendation. In Qiang Yang, Zhi-Hua Zhou, Zhiguo Gong, Min-Ling Zhang, and Sheng-Jun Huang, editors, *Advances in Knowledge Discovery and Data Mining - 23rd Pacific-Asia Conference, PAKDD 2019, Macau, China, April 14-17, 2019, Proceedings, Part III*, volume 11441 of *Lecture Notes in Computer Science*, pages 346–358. Springer, 2019. doi: 10.1007/978-3-030-16142-2_27. URL https://doi.org/10.1007/978-3-030-16142-2_27. 205

Chang Su, Hao Li, and Xianzhong Xie. Personalized ranking point of interest recommendation based on spatial-temporal distance metric in lbsns. In *Proceedings of the 8th International Conference on Software and Computer Applications, ICSCA '19, Penang, Malaysia, February 19-21, 2019*, pages 38–43. ACM, 2019b. doi: 10.1145/3316615.3316715. URL https://doi.org/10.1145/3316615.3316715. 205

Cécile Bothorel, Neal Lathia, Romain Picot-Clémente, and Anastasios Noulas. Location recommendation with social media data. In Peter Brusilovsky and Daqing He, editors, *Social Information Access - Systems and Technologies*, volume 10100 of *Lecture Notes in Computer Science*, pages 624–653. Springer, 2018. doi: 10.1007/978-3-319-90092-6_16. URL https://doi.org/10.1007/978-3-319-90092-6_16. 213