User Modeling and User-Adapted Interaction manuscript No.
(will be inserted by the editor)

Applying reranking strategies to route recommendation using sequence-aware evaluation

Pablo Sánchez · **Alejandro Bellogín**

Received: date / Accepted: date

Abstract Venue recommendation approaches have become particularly useful nowadays due to the increasing number of users registered in Location-Based Social Networks (LBSNs), applications where it is possible to share the venues someone has visited and establish connections with other users in the system. Besides, the venue recommendation problem has certain characteristics that differ from traditional recommendation, and it can also benefit from other contextual aspects to not only recommend independent venues, but complete routes or venue sequences of related locations. Hence, in this paper we investigate the problem of route recommendation under the perspective of generating a sequence of meaningful locations for the users, by analyzing both their personal interests and the intrinsic relationships between the venues. We divide this problem into three stages, proposing general solutions to each case: first, we state a general methodology to derive user routes from LBSNs datasets that can be applied in as many scenarios as possible; second, we define a reranking framework that generate sequences of items from recommendation lists using different techniques; and third, we propose an evaluation metric that captures both accuracy and sequentiality at the same time. We report our experiments on several LBSNs datasets and by means of different recommendation quality metrics and algorithms. As a result, we have found that classical recommender systems are comparable to specifically tailored algorithms for this task, although exploiting the temporal dimension, in general, helps on improving the performance of these techniques; additionally, the proposed reranking strategies show promising results in terms of finding a tradeoff between relevance, sequentiality, and distance, essential dimensions in both venue and route recommendation tasks.

P. Sánchez
Departamento de Ingeniería Informática, Universidad Autónoma de Madrid, Madrid, Spain
E-mail: pablo.sanchezp@uam.es

A. Bellogín
Departamento de Ingeniería Informática, Universidad Autónoma de Madrid, Madrid, Spain
E-mail: alejandro.bellogin@uam.es

Keywords Travel sequences · Route recommendation · Temporal evaluation · Reranking

1 Introduction

Research on Point-of-Interest (POI) or venue recommendation – i.e., suggesting new places for users to visit like hotels, restaurants, museums, etc. – has grown in recent years, in part due to the increasing number of users in Location-Based Social Networks (LBSNs) such as Foursquare, Gowalla, or Yelp. Because of this, many recommendation techniques that exploit these information sources have been proposed, including the works of (Li et al 2015; Lian et al 2014; Zhang et al 2014; Zhang and Chow 2013; Gao et al 2013; Liu et al 2014). In general, all of these approaches incorporate different contextual factors such as geographical, social, or temporal aspects (Liu et al 2014, 2017); however, most of them are focused on recommending a set of venues (presumably) interesting for the user, neglecting the sequential nature of the POI recommendation problem. Furthermore, in some cases, the algorithms are tested under time-agnostic evaluation conditions, ignoring the temporal behavior of the users, which is critical and very informative in LBSNs.

At the same time, there are some works where route recommendation approaches have been proposed in the area of LBSNs (also known as trip or travel recommendation) – see the techniques presented by Fang et al (2014), Zhang et al (2015a) and Yu et al (2016) –, considering additional user constraints such as visiting time, POI availability, travel time, etc., while giving importance to the visit order of the venues. However, in these works it is not common to find comparisons between their approaches and other classic recommendation algorithms, nor the use of sequence-aware evaluation metrics in their experiments. On the other hand, the route recommendation problem has been addressed from adjacent, related areas such as Operational Research to recommend trips (not only independent POIs), defining the so-called Tourist Trip Design Problem (TTDP) (Gavalas et al 2014). However, most of these works are less focused at modeling the user preferences, since they either apply basic filters to take into account only some specific places, solve the problem for each user independently, or generate a sequence of venues by simultaneously considering the transition patterns between POIs, their popularity, and a given length (Gunawan et al 2016; Ayala et al 2017; Chen et al 2016). In most cases, these works are not aware of the large body of research available in the Recommender Systems (RS) and User Modeling communities.

Furthermore, the analysis of sequential information is not new in RS, since there are many applications in which past user sessions (equivalent to the user routes in our case) are determinant to make future recommendations, such as music recommendation or online purchases (Quadrana et al 2018). However, while in those domains it is common to find datasets with user sessions, and the definition of these sessions is straightforward, this is not the case when working in the venue or route recommendation domain, since public datasets typically consist of raw user-POI check-ins or GPS coordinates, where, as mentioned before, few works address this task using a temporal evaluation setting, and even less create or exploit routes so that consecutive check-ins belong to the same route, once a set of constraints are satisfied. As a consequence,

there is no unified approach on how to build routes from check-ins– besides the works of Choudhury et al (2010) and Lim et al (2015) –, which we believe could be useful to develop new methods and adapt recommendation algorithms using routes as user sessions, as well as to evaluate them according to the ones observed in the test set.

Regarding the evaluation of POI recommenders, there are several aspects that need further consideration from the research community. On the one hand, the evaluation of RS at large is still an open problem, where the current understanding is that ranking metrics (like Precision, Recall, or NDCG) are more established and better suited than error metrics (Gunawardana and Shani 2015), even though they are prone to biases (Bellogín et al 2017; Jannach et al 2015), but other dimensions such as novelty, diversity, or freshness are equally, or more, important nowadays (Castells et al 2015; Sánchez and Bellogín 2018b). Related problems such as new or cold-start users, trust and confidence in the system, and privacy, remain unresolved even though they are relevant to the community, and prevalent in POI recommendation (Gunawardana and Shani 2015). On the other hand, temporal and sequential aspects have been recently studied and introduced in the evaluation process, such as in the survey presented by Campos et al (2014) or in recent approaches by Monti et al (2018) and Sánchez and Bellogín (2018a), where the amount of agreement between the provided recommendations and the sequence followed by the user (in test) has been included as a key input to the evaluation metrics.

Therefore, based on the aforementioned problems still unsolved in venue and route recommendation, in this paper we address the following research questions:

RQ1: **How do classical collaborative filtering algorithms compare against approaches tailored to venue and route recommendation?** These methods (collaborative filtering approaches and solutions to the venue and route recommendation problem) have been surveyed and categorized as belonging to different groups in the past, but no explicit comparison between them has been made available, nor using real-world datasets with a large number of users and interactions. In principle, domain-focused algorithms from the TTDP literature would perform better, but they are usually less personalized, whereas classical collaborative filtering approaches understand the user better but fail at modeling the domain and its inherent characteristics. To address this question, we empirically compare state-of-the-art recommenders that belong to these two families on diverse datasets. We explore various evaluation dimensions such as accuracy, novelty, and diversity, and also include other popular and useful dimensions in the venue and route recommendation tasks, such as the total distance of the suggested sequence and its similarity in terms of categories.

RQ2: **Is it possible to improve the results of these algorithms by reranking the recommendations so as to create meaningful sequences?** We explore reranking strategies and propose a generic framework that takes into account different hypotheses to build sequences based on the recommendations produced by any algorithm. This framework would allow to generate sequences aiming to maximize different criteria relevant for route and venue recommender systems, while, at the same time, it would take into account the personalization component provided by the recommendation techniques.

RQ3: **How does the performance change when the user sequences in test are compared against the generated recommendations in a formal, principled way?** We address this question by presenting an evaluation metric that captures the similarity between the user sequences available in the test set and the recommendation lists, by adapting the Longest Common Subsequence technique (Apostolico 1997) so that it can be integrated in any ranking-based metric. Once such a metric is available, we can measure how effective the recommendation algorithms are in terms of predicting the actual user sequences, together with the reranking strategies developed for the previous research question.

Hence, the main contributions of this paper include:

- A framework to build venue sequences or routes from raw check-in data. In fact, even though there are some steps in this framework tailored to the venue recommendation domain, it is generic enough to be applied to any dataset containing temporal or sequential information.
- The definition of and experimentation with item reranking strategies to generate venue sequences from non-sequential recommenders.
- A new family of evaluation metrics where sequentiality has been incorporated into classical evaluation metrics in order to favor those recommenders that return the venues in the same order as the one followed by users in test. As before, these novel sequence-aware metrics could be applied within any time-aware dataset, not only those based on LBSNs.

The remainder of the paper is organized as follows: first, Section 2 summarizes recent approaches for the standard problem of POI recommendation as well as the problem of recommending personalized routes, presenting the specific characteristics and main issues that these algorithms face at the moment. Then, Section 3 defines our generic and configurable framework to generate routes (as sequences of venues) from datasets formed by user interactions with different POIs or venues represented as check-ins throughout time. Section 4 introduces our novel approach based on reranking to create routes by reordering the top candidates suggested by any recommendation algorithm using several strategies based on different hypotheses. Section 5 reformulates standard evaluation metrics to include the order of the sequences in the analysis of the recommendation quality. Finally, Sections 6 and 7 present in detail the methodology and different datasets we have used in the experiments, as well as the recommenders tested, together with a detailed study of their performance on different dimensions (relevance, novelty, diversity, distance, and sequentiality). The paper ends in Section 8 with the main conclusions and possible future work directions, although additional results and details not presented in the main text are included in an appendix at the end of the paper.

2 State of the art

As in other traditional recommendation domains, venue or POI recommendation approaches aim to maximize the number of relevant POIs they suggest to the users, to

alleviate the problem of an ever-increasing number of venues which should fit the user preferences. However, despite their commonalities, this type of recommendation has some inherent properties that differ from the classic recommendation problem. Firstly, the sparsity of the datasets used by POI recommendation algorithms are considerably higher than those used in traditional recommendation; for example, the density of the Movielens10M dataset is 1.34% whereas the global check-in dataset from Foursquare is 0.0034% and a popular dataset from Gowalla is 0.0047% (Harper and Konstan 2016; Yang et al 2016; Cho et al 2011). This might be attributed to a very large number of items (POIs) in each city and the fact that users do not interact with most of them, either because they are not close to the most visited areas of the user or because the user is not linked to that city (she is a tourist). In fact, this problem in worldwide datasets is even worse, since POIs from different cities are, by definition, different, hence, the more cities are included in a dataset, the larger the number of items, in contrast to other domains such as music or movies where an expansion of the dataset could overlap with the items already included.

Secondly, when performing venue or POI recommendations, the geographical influence is perhaps the most important feature to be taken into account, since it is generally assumed that users tend to prefer venues that are close to each other (Miller 2004); because of this, most current approaches propose different models to exploit such feature. Nevertheless, there are other contextual factors that can be used to improve the quality of recommendations, including social and temporal information or even the weather (Liu et al 2017; Braunhofer et al 2014). Along with the geographical influence, temporal information has proven to be truly useful as it helps to discriminate better the user behavior at different granularity levels (seasonal trends, different moments of the day, etc.). On the other hand, while geographical and temporal information are normally available in datasets based on LBSNs (both geographical locations and timestamps are included in popular datasets gathered, for instance, from Foursquare and Gowalla), weather data is more difficult to capture. It normally requires external sources of information, although it has been demonstrated that it affects the behavior of tourists (Braunhofer et al 2014) and that some weather factors have an impact on users' check-in behavior (Trattner et al 2018). However, a proper investigation about the difference between exploiting weather forecasts and actual weather conditions in place is still missing in the literature (Trattner et al 2018).

Regarding the geographical component, there are several proposed approaches that model this aspect; for example, Liu et al (2014) and Li et al (2015) use the influence of neighbor POIs of a target POI to compute the score of a given user with a weighted Matrix Factorization (MF) scheme and the Bayesian Personalized Ranking (BPR) optimization algorithm (Rendle et al 2009), respectively. The USG model from Ye et al (2011) estimates the probability of visiting the target POI using the history of POIs already visited by the target user, which is then combined with a user-based and a friend-based collaborative filtering algorithm. On the other hand, Zhang and Chow (2013) and Zhang et al (2014) apply the Kernel Density Estimation (KDE) technique to learn distributions to model the probability of a user visiting a venue, combined with a friend-based collaborative-filtering model. In the first article, the authors use the KDE technique to estimate a one dimensional distance probability distribution using the distances between the target POI and the rest of the POIs visited by the user. In the

second one, they use the same technique but over a two-dimensional space (latitude and longitude of the venues visited by the user), and they compute the probability of visiting the target POI based on its location, which is estimated according to the check-in probability distribution for the user.

Nevertheless, the venues suggested by these types of recommenders are not usually related to each other (beyond that they are all hypothetically interesting *at the same time* for the target user). In general, these algorithms aim to suggest as many relevant venues for the user as possible, but these venues are not expected to be visited one after the other, as a route, which is in contrast with the solutions proposed to the Tourist Trip Design Problem (TTDP). The TTDP extends the classic POI recommendation problem so that instead of recommending independent POIs, it intends to recommend a sequence of POIs to be visited in a specific order (Ayala et al 2017); however, as mentioned in the previous section, these approaches are usually less personalized or the user preferences are considered as a pre-filtering stage. Indeed, there are several variants of the TTDP, one of the most important ones is the Orienteering Problem (OP), a generalization of the Traveling Salesman Problem in which the goal is to maximize the score of visiting a subset of nodes in a graph so that the travel cost does not exceed a maximum bound (Gunawan et al 2016). Temporal information has also been incorporated to this problem, under the OP with Time Windows which considers that a node can be accessed only in a specific time window, or the Time Dependent OP where the travel time between two nodes may vary, instead of assuming to have a constant value (Gunawan et al 2016).

Finally, even though the problem of venue recommendation has been studied from different perspectives (see the surveys collected by Liu et al (2017) and Zhao et al (2018a), for example), there are few works devoted to the problem of personalized route (as a sequence of venues) recommendation. Specifically, Ayala et al (2017) proposed three approaches to solve the TTDP incorporating information of public transport by adjusting the visit plans according to the real travel times. Chen et al (2016) defined a probabilistic model combining ranking scores and transition probabilities for their route recommender. Lim et al (2018) created PersTour, an algorithm that combines the user interests, POI popularity, and trip constraints in order to recommend trip itineraries. Laß et al (2017) proposed TourRec, a context-aware tour recommendation application for tourists that first scores each POI for every user by combining the venue popularity and the user preferences and it then finds a route by applying a Dijkstra-based algorithm. Finally, Choudhury et al (2010) proposed a method to generate a graph of POIs from photo streams extracted from Flickr, which is then used to build travel itineraries automatically by exploiting the popularity of the POIs. Regarding other approaches that specifically use LBSNs, Fang et al (2014) proposed a two-step model that first infers the scores of the venues taking into account temporal information and later tries to connect the POIs by optimizing the trip that satisfies the user constraints. Zhang et al (2015a), on the other hand, presented a probabilistic model that considers the user time budget, the time-window of the POIs, and the user preferences, whereas Yu et al (2016) developed an app that uses collaborative filtering approaches together with the popularity of the POIs and the frequency at which two POIs are visited in sequence to suggest travel packages to the users accessing their system via mobile phones.

This limited number of proposals on the problem of route recommendation is reasonable, since the task is more difficult than simply recommending independent venues to a user. Even more if we consider that public datasets with enough information are not always available, especially regarding explicit routes followed by the users. In the next section we present a framework to alleviate this issue in typical scenarios, i.e., when using datasets with raw check-ins, which are the most popular ones and used in the community nowadays.

3 A general framework to build routes from check-in datasets

In the route recommendation domain, users follow travel sequences or routes by visiting POIs that are related with each other (for example, 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 (routes) 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 (also called trips, itineraries, or trajectories in the literature). It should be noted, however, that existing datasets in those domains such as YOOCHOOSE (Ben-Shimon et al 2015) and Tmall (Liu et al 2016), 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 see Figure 1, together with Table 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

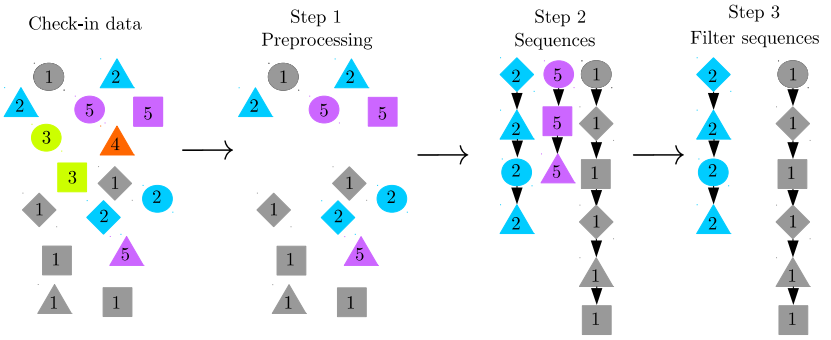


Fig. 1: Visual description of the framework presented in Section 3 to obtain sequences from raw check-ins. Each number and color represent a different user in the system and each shape depicts the type of the POI (museums, restaurants, hotels, etc.) a user interacted with (not necessarily a different POI each time).

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 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).

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.

Table 1: Summary of the parameters used in the framework presented in Section 3.

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

  - 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 groundtruth 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 still 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. 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) 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 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. In Section 5, we introduce a novel evaluation metric that captures exactly this aspect when comparing the recommendations against the groundtruth; but before that, in the next section we present a general recommendation approach that aims to produce meaningful routes or travel sequences from sequence-agnostic algorithms.

4 A novel approach for sequential venue recommender systems based on reranking

4.1 Item reranking in retrieval and recommendation

In the fields of Recommendation Systems and Information Retrieval, some models frequently tend to recover very similar items in top-N rankings. To solve this prob-

lem, 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 2019), 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.

4.2 Using item reranking to generate sequences of venues

In this paper, 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 $f_{obj}(u,i,R)$ 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 to be returned. 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 $f_{rec}(u,i,R)$ (although since

most recommendation algorithms typically ignore the previously ranked items, the notation could be simplified to $f_{rec}(u,i)$) and $f_{seq}(u,i,R)$, respectively:

$$f_{obj}(u,i,R) = \lambda \cdot f_{rec}(u,i) + (1-\lambda) \cdot f_{seq}(u,i,R) \quad (1)$$

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 λ.

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 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: $f^{rnd}_{seq}(u,i,R) = \text{rnd} \in [0,1]$.
  - **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.): $f^{rec}_{seq}(u,i,R) = r(u,i)$. 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; let us denote such last item as i_{n-1}, i.e., $R = \{i_1, \cdots, i_{n-1}\}$. 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: $f^{dist}_{seq}(u,i,R) = 1/dist(i_{n-1},i)$.
  - **Feature-based Markov Chain:** reranker that selects those venues whose features maximize the transition probability with respect to target item i's features: $f^{feat}_{seq}(u,i,R) = p(i^a|i^a_{n-1})$, 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}: $f^{item}_{seq}(u,i,R) = p(i|i_{n-1})$.
- Dependent on the whole sequence: the score depends on the entire sequence being generated, i.e., the current list R and the potential following item i. We define three different algorithms:
  - **LCS-based:** reranker that maximizes the Longest Common Subsequence (an algorithm that will be defined in detail in the next section) between the

Table 2: Summary of the rerankers defined in Section 4.2.

Family	Name	Abbr.	Description
Independent	Random	f_{seq}^{rnd}	Items reranked randomly
	Recommender-based	f_{seq}^{rec}	Items reranked according to the score provided 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 following the same order as in the test set

sequence of item features in the current recommended list R assuming item i is recommended at the end $(R+i)$ and the item features built from the training set of the user (u^a) ordered by timestamp: $f_{seq}^{lcs}(u,i,R) = lcs((R+i)^a, u^a)$.

– **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 $\{(R+i)\}_m$, 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: $f_{seq}^{stree}(u,i,R) = \delta_{ST(u^a)}(\{(R+i)^a\}_m)$, 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: $f_{seq}^{oracle}(u,i,R) = order_{test}(u,i)$. 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 2016): 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

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.

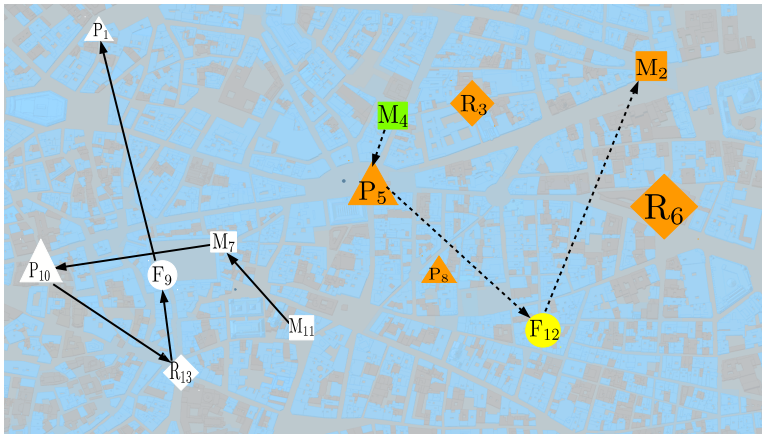
4.3 Solving ties and coverage issues of reranking strategies

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 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 when they 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 work 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.



f^{lcs}_{seq} $\mathrm{M}_4 \to \mathrm{P}_5 \to \mathrm{R}_3 \to \mathrm{P}_8$ f^{dist}_{seq} $\mathrm{M}_4 \to \mathrm{P}_5 \to \mathrm{P}_8 \to \mathrm{R}_3 \to \mathrm{M}_2 \to \mathrm{R}_6$

f^{stree}_{seq} $\mathrm{M}_4 \to \mathrm{P}_5 \to \mathrm{R}_3$ f^{rec}_{seq} $\mathrm{M}_4 \to \mathrm{R}_6 \to \mathrm{P}_5 \to \mathrm{R}_3 \to \mathrm{M}_2 \to \mathrm{P}_8$

f^{oracle}_{seq} $\mathrm{M}_4 \to \mathrm{P}_5 \to \mathrm{M}_2$

Fig. 2: Comparison of 5 sequences of venues generated by different reranking strategies as presented in Section 4.2. 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).

4.4 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^{feat}_{seq} and f^{item}_{seq}) 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^{lcs}_{seq} and f^{stree}_{seq} is subtle but important: whereas f^{lcs}_{seq} aims to find those items that produce the longest common subsequence when added to the recommendation list, f^{stree}_{seq} 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 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 data (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 items. In contrast, the f_{seq}^{lcs} reranker is able to add another POI following the Longest Common Subsequence with P_8, since it allows gaps when searching for the subsequence. The other sequences obtained by the reranking approaches are more straightforward: they either exploit the popularity (size) of the POIs (f_{seq}^{rec} approach using the Popularity recommender as reranker) or the distance between them (f_{seq}^{dist}). Finally, the oracle reranker f_{seq}^{oracle} returns the items in the test set in the order followed by the user, except for F_{12} because it does not belong to the set of candidate items.

5 Sequential recommender systems evaluation

When analyzing the performance of recommenders systems, the community has usually focused on maximizing the number of relevant items the algorithm is able to recommend. However, 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, for others (such as route or playlist recommendation) the order in which items are returned may be quite important to maximize the user satisfaction.

In fact, except for the work presented in (Chen et al 2016), where the authors propose a metric based on F_1 that takes into account the pairwise order between POIs, and the evaluation done in (Menon et al 2017) that used the same metric based on F_1 on pairs of points, we have not found other approaches where the evaluation metrics explicitly compare the order of the recommendations against the visited venues. Nonetheless, this topic is gaining interest nowadays; in Sánchez and Bellogín (2018b) we experimented with different temporal models to assess the freshness of the items when evaluating the recommendations received by an algorithm in a general context, whereas Monti et al (2018) proposed a framework for sequence-based recommender systems, although 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 2018) 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 find a subsequence of elements (not necessarily consecutive) whose length is the maximum possible between two sequences (Apostolico 1997). This algorithm is common in the fields of text editing and molecular sequence comparisons, and we have used it for recommendation to define a user similarity approach aware of sequential patterns in (Bellogín and Sánchez 2017a) which was extended in (Sánchez and Bellogín 2019) to also work with content data; here we propose to use it for evaluation instead, as we shall describe next.

Our proposal, hence, is to incorporate the LCS technique into well-known ranking evaluation metrics such as Precision or NDCG (Gunawardana and Shani 2015). 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, 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 visited venues 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 metric could be adapted in such a way that sequentiality is integrated seamlessly in their computation. In particular, to compute the LCS between two strings X and Y with lengths l_x and l_y, a dynamic programming approach is normally used that fills a matrix $C_{(l_x+1)\times(l_y+1)}$ following this equation:

$$C[i,j] = \begin{cases} 0 & \text{if i = 0 or j = 0} \\ C[i-1,j-1]+1 & \text{if } i,j>0 \text{ and } x_i = y_j \\ \max(C[i,j-1], C[i-1,j]) & \text{if } i,j>0 \text{ and } x_i \neq y_j \end{cases} \quad (2)$$

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.

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 2), since that means that the subsequence found is now larger than before and, hence, the sequentiality will be considered. However, this process would not

Data: $r(u), te(u), C$
Result: value of a given metric m on the subsequence found by LCS

```
s ← 0;
(i, j) ← dim(C);
while i > 1 AND j > 1 do
    x_i ← r(u)[i];
    y_j ← te(u)[j];
    if x_i = y_j then                                   // There is a matching
        s ← s + m(r(u), te(u), x_i, i);
        i ← i − 1;
        j ← j − 1;
    else if C[i − 1][j] > C[i][j − 1] then
        i ← i − 1;
    else
        j ← j − 1;
    end
end
```

Algorithm 1: Using backtracking in Longest Common Subsequence to update evaluation metrics based on sequentiality.

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 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 1) that uses the computed matrix C to find which elements belong to the common subsequence.

When performing the backtracking step as in Algorithm 1, we assume any evaluation metric can be divided in a user-item basis in such a way that $m(r,u,x_i,i)$ provides the contribution that item x_i presented in the recommendation list r 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: $m_P = 1/|r|$ for Precision, $m_R = 1/|u|$ 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 metrics and the proposed adaptation (based on LCS) for sequence-aware metrics, we show in Figure 3 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 that P_s is equivalent to P only when the relevant items are returned in the same order – that is,

[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.

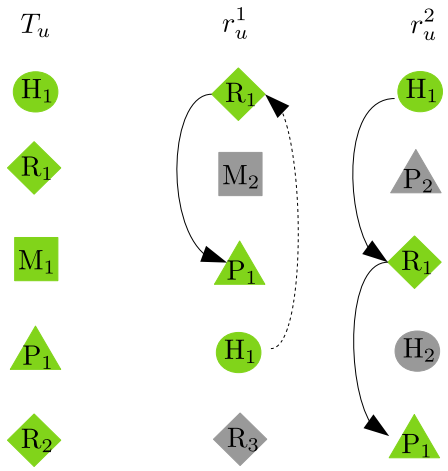


Fig. 3: Comparison between two lists that return the same number of relevant items (3, marked in green), but in one case the list does not present all the items in the correct order (r_u^1), hence $LCS(T_u, r_u^1) = 2$, whereas in the other case, the order is the same and the value of LCS is maximum: $LCS(T_u, r_u^2) = 3$. As a consequence, P_s $(r_u^2) = P(r_u^2) = 3/5$, whereas P_s $(r_u^1) = 2/5$ and $P(r_u^1) = 3/5$.

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 route, despite having gaps with respect to the test set, should also be considered interesting for the user.

Finally, it is important to emphasize that this evaluation metric would not be, in any case, biased towards the previously defined LCS-based reranking approach or any other sequence-aware recommendation methods. It is true that it is designed to produce higher values if the recommended list follows the order in the test set, but besides that, it is agnostic as to how such ranking was generated. This is because of the following reasons: a) first, the metric checks the test of the user against the recommendation list, whereas a reranking strategy or a recommendation algorithm would compare a recommendation list against the user historical preferences; b) second, there is no learning process, in contrast to situations where an optimization method is used with a metric and the same metric is later used in the evaluation; c) and, third, the definition of what a sequence is, is *universal*, the LCS algorithm provides us one possible technique to (more or less) efficiently obtain the sequences, but the same result could be obtained if LCS is not used in the evaluation (or in the reranking strategy) but another method to obtain sequences (such as brute force) is used instead.

Table 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 ($C_{\bar{r}}$).

Dataset	$\|U\|$	$\|I\|$	$\|C_r\|$	$\|C_{\bar{r}}\|$
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

6 Experimental setup

6.1 Datasets

For the experiments, we used three different datasets built from two data sources: the Global-scale check-in dataset from Yang et al (2016), 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 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 2017; Li et al 2017; Liu et al 2014) because it is a realistic, natural partition of the data; besides, items in this domain are by definition unique in each city and, hence, the benefit of combining information from many other venues of other cities (or even countries) is reduced.

6.1.1 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[2]. In this dataset, most of the check-ins come from Turkey and 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.

6.1.2 Semantic trails

In this dataset, the authors integrated different data sources to provide semantically annotated user trails. They did this by starting from the dataset described in the previous section, grouping the check-ins into sequences of activities, and then enriching it with

[2] https://sites.google.com/site/yangdingqi/home/foursquare-dataset

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 paper and can be obtained here[3]. In particular, 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). 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.

6.1.3 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[4].

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 it can be observed in Table 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 is very different to the other ones; in fact, probably these items reflect more faithfully interesting items 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; this mapping is included in our code repository.

[3] https://figshare.com/articles/Semantic_Trails_Datasets/7429076

[4] https://github.com/igobrilhante/TripBuilder

Table 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	$\lvert\mathbf{U}\rvert$	$\lvert\mathbf{V}\rvert$	$\lvert\mathbf{C_r}\rvert$	$\lvert\mathbf{C_{\bar{r}}}\rvert$	$\frac{\lvert\mathbf{C_r}\rvert}{\lvert\mathbf{U}\rvert\cdot\lvert\mathbf{V}\rvert}$ %	$\frac{\lvert\mathbf{C_{\bar{r}}}\rvert}{\lvert\mathbf{U}\rvert\cdot\lvert\mathbf{V}\rvert}$ %	$\lvert\mathbf{S}\rvert$
NYC	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
TOK	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
ROM	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
PJ	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

6.1.4 Datasets processing

The three datasets described earlier have been processed to obtain 4 cities where the recommenders will be trained and evaluated: New York and Tokyo from the Global-scale check-in dataset, Petaling Jaya from Semantic trails, and Rome 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 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; 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 3, 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 ($p_b^t = 60, p_b^i = 3$). 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.

6.2 Evaluation methodology

As discussed at the end of Section 3, 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 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 6.4, 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 metric (see Section 5) is able to deal with repeated items.

Unless stated otherwise, the results are shown using the Train Items 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 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 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[5]. For relevance, we show the results using Precision (P) and NDCG, together with their sequential counterparts: P_s and $NDCG_s$, according to their definitions in Section 5. On the other hand, for novelty and diversity, we show the results in terms of EPC (where it assumes that the higher the popularity of the POI, the lower its novelty) and Gini (which accounts for how evenly distributed the recommended items are, so that the higher the obtained value, the higher the diversity) (Vargas and Castells 2014). 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 2017; 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

[5] 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.

Table 5: Summary of the evaluation metrics used in the experiments, including their abbreviation and description.

Metric	Abbr.	Description
Precision	P	Counts how many items from the test set are recommended
Normalized Discounted Cumulative Gain	NDCG	Compares the recommendation list against an ideal ranking where the items in the test set are ranked first
Test Feature Precision	TFP	Counts how many categories from the items in the test set are recommended, with some restrictions; see Section 6.3
Sequential Precision	P_s	Counts how many items from the test set are recommended taking into account the order of those items as they appear in the test set
Sequential NDCG	$NDCG_s$	Compares the recommendation list against an ideal ranking where the items in the test set are ranked according to the order followed by the user in the test set
Sequential Feature Precision	FP_s	An extension of Test Feature Precision where the order of the items as they appear in the test set is considered
Distance	Dist	Measures the pairwise distance between each item in the recommended list and the next one
Gini	Gini	Measures the diversity of the recommended list by considering how uniform the distribution of recommended items is; see (Vargas and Castells 2014)
Expected Popularity Complement	EPC	Measures the novelty of the recommended items by assuming that the larger the popularity of an item, the lower its novelty; see (Vargas and Castells 2014)

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 5, 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 P_s but matching categories instead of POIs, we call it Sequential Feature Precision or 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.

We include in Table 5 a summary of the evaluation metrics used in the experiments, together with a brief description and their abbreviation.

6.4 Recommenders

We experiment with 27 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 TTDP since they need 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

Table 6: 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 uses 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}
	User-based nearest neighbor	UB	Y	$k = \{40, 60, 80, 100, 120\}$, sim = {Jac, Cos }
	Item-based nearest neighbor	IB	Y	$k = \{40, 60, 80, 100, 120\}$, sim = {Jac, Cos }
	MF using Alternate Least Squares	HKV	Y	$k = \{10, 50, 100\}$, $\alpha = \{0.1, 1\}$, $\lambda = \{0.1, 1\}$
	MF with Bayesian Personalized Ranking	BPR	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 = {Jac, Cos }, $\lambda = \{0.05, 0.1\}$
	Backward-Forward	BF	N	$k = \{40, 60, 80, 100, 120\}$, sim = {Jac, Cos }, Wgt = {T, F }, Nrm = {Def, Std, Rks }, $L_m^- = L_m^+ = \{5, 10\}$
	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
	Hybrid: Pop + UB + AvgDis	PGN	Y	$k = \{40, 60, 80, 100, 120\}$, sim = {Jac, Cos }
	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, $\varepsilon = 0.3$, 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

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.

Below, we briefly introduce each recommendation algorithm used, including its main reference paper; we also include details about where its implementation is taken from (either our own, or using public libraries, mainly RankSys[6] and MyMediaLite[7]). All the code for these experiments is available in the following Bitbucket repository: PabloSanchezP/SeReRSys.

- Basic: simple baselines, useful to test biases on the recommendations:
  - Rnd: a random recommender; RankSys implementation.
  - Pop: recommender that returns the items ordered by descending popularity; RankSys implementation.
  - Train: recommender that suggests the items already interacted by the user in the training set, ordered by frequency and, in case of ties, by popularity; our own implementation.

[6] Java 8 Recommender Systems framework, available at https://github.com/RankSys/RankSys and described in (Vargas 2015).

[7] MyMediaLite Recommender System Library, available at http://www.mymedialite.net and described in (Gantner et al 2011).

  - $\overline{\text{Train}}$: 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: a hybrid recommender system where a user-based nearest neighbor formulation is used with content-based similarities between users, where the users are represented as in CBUI. Based on the collaborative-via-content approach (Balabanovic and Shoham 1997); our own implementation.
  - UB: a user-based nearest neighbor algorithm with k neighbors (k-nn) using collaborative similarities between users (Ning et al 2015), we follow a variation that performs better in terms of ranking metrics because it does not normalize the score by the sum of similarities (Aiolli 2013); RankSys implementation.
  - IB: an item-based nearest neighbor algorithm with k neighbors (k-nn) using collaborative similarities between items (Ning et al 2015), without normalizing by the sum of similarities; RankSys implementation.
  - HKV: a matrix factorization approach as described in (Hu et al 2008) that uses Alternate Least Squares in the minimization formula; RankSys implementation.
  - BPR: the Bayesian Personalized Ranking from (Rendle et al 2009) using a matrix factorization technique; MyMediaLite implementation.
- 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 (Sánchez and Bellogín 2018b); our own implementation.
  - TD: a modified version of UB with an exponential time decay (controlled by parameter λ), so that the neighbors' scores are more penalized if the timestamp when the neighbor interacted with the target item is much older than the last interaction of the target user. We use a variation from the one defined in (Campos et al 2014) where we do not normalize by the sum of similarities; our own implementation.
  - BF: a sequence-aware approach presented in (Bellogín and Sánchez 2017b) that extends the UB recommender so that each neighbor selects the candidate items according to the last common interaction with respect to the target user, the different lists are combined using rank fusion techniques (parameters Wgt and Nrm); our own implementation.

  - MC: first order Factorized Markov Chain recommender algorithm (Rendle et al 2010); implementation provided by other authors[8].
  - FPMC: Factorized Personalized Markov Chain recommender, that is, a combination of Markov Chain and Matrix Factorization techniques (Rendle et al 2010); same implementation as MC.
  - Fossil: Factorized Sequential Prediction with Item Similarity Models from (He and McAuley 2016), this model combines Factorized Item Similarity Models and high-order Markov Chains; same implementation as MC.
  - Caser: Convolutional Neural Network proposed in (Tang and Wang 2018) that combines the sequential patterns inferred from the data and the general preferences of the users to make recommendations; implementation provided by the authors[9].
- 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[10].
  - RankGeoFM: Matrix Factorization method using BPR as proposed in (Li et al 2015), 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[11].
- 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 are

[8] The authors of (He and McAuley 2016) provide implementation of some related algorithms, they are all available at https://drive.google.com/file/d/0B9Ck8jw-TZUEeEhSWXU2WWloc0k/view, although we adapted such code to generate a ranking from any recommendation algorithm, since the original code only used AUC as evaluation metric and did not produce rankings.

[9] We use the Python implementation available here: https://github.com/graytowne/caser_pytorch.

[10] The authors of (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/ .

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

Table 7: Optimal parameters found in each city, see Table 6 for the explored range of the parameters.

Rec	NYC	TOK	ROM	PJ
CBCF: k, Wgt	100, Yes	120, No	120, No	120, No
UB: k, sim	100, Jac	120, Cos	120, Jac	120, Jac
IB: k, sim	60, Jac	120, Jac	100, Jac	100, Jac
HKV: k, α, λ	50, 0.1, 1	10, 0.1, 0.1	10, 0.1, 1	10, 1, 1
BPR: k, $\lambda_u = \lambda_i$, λ_0	100, 0.005, 0	50, 0.005, 0	100, 0.1, 0	50, 0.1, 0
TD: k, sim, λ	120, Jac, 0.1	120, Cos, 0.1	120, Cos, 0.05	120, Cos, 0.05
BF: k, sim, Wgt, Nrm, $L_m^- = L_m^+$	120, Jac, F, Def, 10	100, Cos, F, Def, 5	120, Jac, T, Def, 10	120, Cos, T, Def, 5
MC: k, λ	2, 0.1	10, 0.1	2, 0.2	20, 0.2
FPMC: k, λ	2, 0.2	10, 0.2	20, 0.2	20, 0.1
Fossil: k, λ, 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, Jac	100, Cos	100, Cos	100, Jac
IRenMF: k, α, λ_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, α, 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)

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 6.2.
  - 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 the range of the parameters tested in the experiments, whereas Table 7 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. Note that we also include whether the recommender uses an aggregated training or not (as explained in Section 6.2); we based this decision according to the nature of each algorithm and, in some cases, because better results were obtained for the reported combination.

7 Results

7.1 Performance of venue recommender systems

Table 8 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 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 7.3.

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 7.3 for more details).

In the following sections, we analyze these and other results according to different evaluation dimensions.

7.1.1 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.

When comparing Precision 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

Table 8: 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
	UB	0.139	0.409	**0.340**	0.136	0.396	**0.300**	40.8	0.008	0.933
	IB	0.121	0.383	0.326	0.119	0.373	0.298	**24.2**	0.047	0.971
	HKV	0.121	0.382	0.323	0.120	0.372	0.289	35.8	0.004	0.949
	BPR	**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**
	BF	0.142	0.417	†**0.349**	0.138	0.402	**0.307**	41.8	0.004	0.926
	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**

mentioned in Section 5); 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 4 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 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

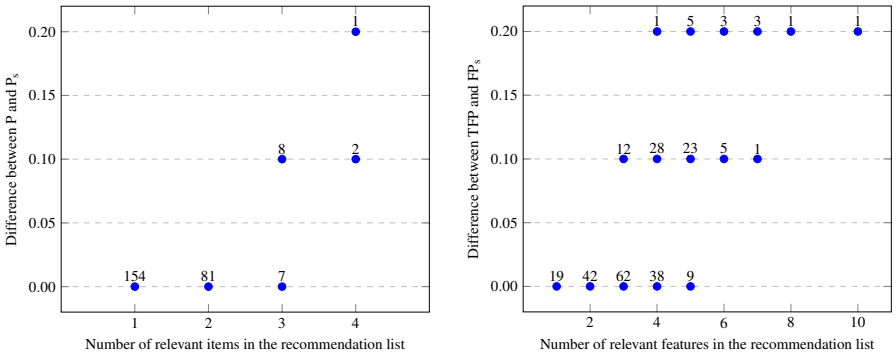


Fig. 4: Difference between sequential (P_s and FP_s) and non-sequential (P and TFP) metric values and the number of users with such difference, depending on the number of relevant items or features returned by IRenMF in NYC.

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 BF) 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. 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.

7.1.2 Behavior of evaluated recommenders

According to the presented results in Table 8, 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 7 where only in one case 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. 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 POI recommendation datasets in contrast with the original papers where these algorithms were proposed, and in some cases the authors even filtered out those users with a low number of interactions.

From the Temporal family, only BF is able to get a performance close to the TPop recommender in terms of relevance and it is the best one in the FP_s 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

Table 9: Performance for New York, Rome, and Petaling Jaya, only showing the best recommender for each family. Notation and cutoffs like in Table 8.

City	Family	Rec	Accuracy			Seq. Accuracy			Non Accuracy		
			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	BPR	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
	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	BPR	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	BPR	†**0.131**	†**0.418**	0.274	0.128	0.408	0.270	30.0	0.001	0.917
	Temporal	BF	0.130	0.416	†**0.328**	†**0.129**	†**0.409**	†**0.310**	†**24.7**	0.004	†**0.938**
	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**

two MF approaches – HKV and BPR– 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, BPR 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 6.4).

7.1.3 Analysis on other cities

So far, we have only explored the results for the city of New York; now in Table 9 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.1, we show and discuss the results for all the cities described in Section 6.1, but in the rest of the paper 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 4). First, the BPR recommender is the best one for the Classic 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 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[12], 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 on check-ins, despite their pervasiveness in the field, since they could provide a different perspective of user behavior.

7.1.4 Short summary

Based on the conclusions drawn so far, we can provide an answer to **RQ1** regarding 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 **RQ3** and 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

[12] 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.

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.

7.2 Performance of reranking strategies

In this section, we present experiments for the proposed reranking strategies. For this, we take Equation 1 with $\lambda = 0$, thus, only the sequence-aware reranker component is considered (later in Section 7.3 we analyze the results at different values of λ). Moreover, we rerank the first 20 items returned by each recommender for every user using the 8 components presented in Section 4; 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 10 shows the results obtained for the rerankers described in Section 4 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 6.4), 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: NDCG_s (sequence-aware item-level relevance), FP_s (sequence-aware category-level relevance), and Dist (distance of the recommended route). As before, complete results are presented in Appendix A.2 as separate tables, showing the results for all the cities presented in previous sections on every evaluation metric (see Tables 12, 13, 14, 15, and 16).

7.2.1 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 FP_s, but also for 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

Table 10: Effect of the reranking strategies ($\lambda = 0$, 20 candidate items reranked) on each family of recommendation algorithms under three evaluation metrics.

		NYC			ROM			PJ		
Family	Reranker	$NDCG_s$	FP_s	Dist	$NDCG_s$	FP_s	Dist	$NDCG_s$	FP_s	Dist
Basic	Baseline	0.402	0.284	43.9	0.447	0.464	5.0	0.404	0.245	35.0
	f^{rnd}_{seq}	0.383	0.297	28.0	0.402	0.452	5.9	0.387	0.274	29.5
	f^{dist}_{seq}	0.396	▲**0.308**	▲**4.1**	0.469	†**0.474**	▲**1.4**	†**0.409**	▲**0.296**	▲**7.2**
	f^{feat}_{seq}	0.400	0.267	33.3	0.422	0.371	5.0	0.402	0.267	33.2
	f^{item}_{seq}	0.399	0.279	37.8	†**0.473**	0.469	1.8	0.408	0.262	19.5
	f^{rec}_{seq}	†**0.406**	0.298	42.4	0.422	0.452	6.0	0.407	0.271	26.3
	f^{lcs}_{seq}	0.395	0.285	17.8	0.440	0.446	2.3	0.403	0.274	14.8
	f^{stree}_{seq}	0.402	0.289	38.4	0.446	0.466	3.2	0.403	0.263	25.9
	f^{oracle}_{seq}	▲**0.468**	0.296	43.2	▲**0.614**	▲**0.482**	4.2	▲**0.456**	0.247	34.2
Classic	Baseline	0.404	0.285	45.3	0.447	0.460	6.3	0.408	0.270	30.0
	f^{rnd}_{seq}	0.382	0.292	30.3	0.403	0.450	5.9	0.394	0.278	30.7
	f^{dist}_{seq}	0.395	▲**0.309**	▲**4.2**	0.468	†**0.475**	▲**1.4**	†**0.410**	▲**0.294**	▲**7.4**
	f^{feat}_{seq}	0.398	0.267	33.5	0.424	0.373	5.0	0.402	0.269	34.0
	f^{item}_{seq}	0.400	0.276	38.0	†**0.476**	0.468	1.8	0.409	0.268	18.5
	f^{rec}_{seq}	†**0.406**	0.300	42.4	0.422	0.452	6.0	0.407	0.273	26.5
	f^{lcs}_{seq}	0.395	0.284	17.9	0.440	0.447	2.3	0.405	0.279	13.2
	f^{stree}_{seq}	0.404	0.294	38.6	0.447	0.465	3.7	0.405	0.275	22.1
	f^{oracle}_{seq}	▲**0.468**	0.300	44.3	▲**0.612**	▲**0.482**	4.9	▲**0.455**	0.269	29.0
Temporal	Baseline	0.404	0.284	43.2	0.447	†**0.469**	4.9	0.409	0.310	24.7
	f^{rnd}_{seq}	0.385	0.300	30.3	0.409	0.449	6.0	0.393	▲**0.329**	25.9
	f^{dist}_{seq}	0.395	▲**0.308**	▲**4.1**	0.464	0.468	▲**1.4**	0.405	0.327	▲**5.4**
	f^{feat}_{seq}	0.399	0.266	33.3	0.421	0.375	5.0	0.397	0.294	23.1
	f^{item}_{seq}	0.399	0.277	37.7	†**0.474**	0.465	1.9	†**0.410**	0.291	18.7
	f^{rec}_{seq}	†**0.406**	0.299	42.5	0.422	0.452	6.1	0.408	0.308	25.4
	f^{lcs}_{seq}	0.395	0.288	18.1	0.441	0.447	2.3	0.401	0.319	9.8
	f^{stree}_{seq}	0.404	0.294	38.2	0.445	0.468	3.1	0.403	0.322	15.7
	f^{oracle}_{seq}	▲**0.469**	0.296	42.8	▲**0.608**	▲**0.482**	4.1	▲**0.453**	0.313	23.7
Geo	Baseline	0.405	0.306	43.9	0.427	0.457	5.6	0.406	0.286	30.0
	f^{rnd}_{seq}	0.378	0.307	22.5	0.397	0.447	5.9	0.390	0.307	25.1
	f^{dist}_{seq}	0.385	0.315	▲**3.6**	0.456	†**0.468**	▲**1.4**	0.405	▲**0.315**	▲**5.8**
	f^{feat}_{seq}	0.393	0.281	32.8	0.414	0.364	5.3	0.397	0.282	26.8
	f^{item}_{seq}	0.402	0.291	37.1	†**0.467**	0.466	2.1	†**0.412**	0.270	18.8
	f^{rec}_{seq}	†**0.405**	0.311	42.0	0.417	0.453	6.0	0.407	0.290	25.8
	f^{lcs}_{seq}	0.390	0.311	11.9	0.426	0.440	2.2	0.401	0.308	10.5
	f^{stree}_{seq}	0.402	▲**0.321**	31.3	0.431	0.458	3.5	0.404	0.302	19.4
	f^{oracle}_{seq}	▲**0.464**	0.314	41.7	▲**0.586**	▲**0.472**	4.6	▲**0.449**	0.287	28.8
Tour	Baseline	0.391	0.279	44.9	†**0.477**	0.473	2.0	0.403	0.240	28.4
	f^{rnd}_{seq}	0.364	0.305	23.9	0.400	0.448	5.7	0.390	0.291	30.8
	f^{dist}_{seq}	0.381	0.311	▲**4.2**	0.467	†**0.474**	▲**1.4**	†**0.412**	▲**0.309**	▲**7.1**
	f^{feat}_{seq}	0.374	0.277	20.9	0.420	0.359	5.0	0.401	0.278	31.6
	f^{item}_{seq}	0.397	0.283	38.1	0.477	0.470	1.8	0.406	0.271	16.9
	f^{rec}_{seq}	†**0.403**	0.289	41.4	0.427	0.451	5.8	0.408	0.273	26.6
	f^{lcs}_{seq}	0.382	▲**0.312**	12.0	0.438	0.446	2.1	0.406	0.290	13.9
	f^{stree}_{seq}	0.386	0.295	32.4	0.457	0.466	2.4	0.403	0.272	21.5
	f^{oracle}_{seq}	▲**0.442**	0.285	44.4	▲**0.600**	▲**0.482**	3.0	▲**0.455**	0.244	28.0

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^{rnd}_{seq}, 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^{oracle}_{seq}) 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 FP_s in NYC for the Temporal family increases from 0.284 to 0.299 when applying the f_{seq}^{rec} reranker – an improvement of 4.9% –, and the performance on $NDCG_s$ 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.

7.2.2 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 $NDCG_s$ and 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 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 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 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 FP_s and $NDCG_s$. 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 $NDCG_s$ and FP_s.

7.2.3 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

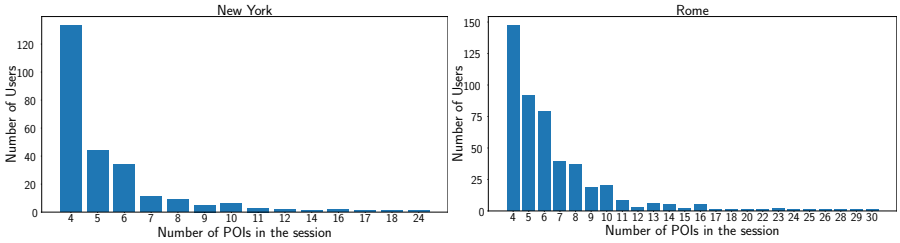


Fig. 5: Frequency of the number of venues each user visited in test for New York and Rome.

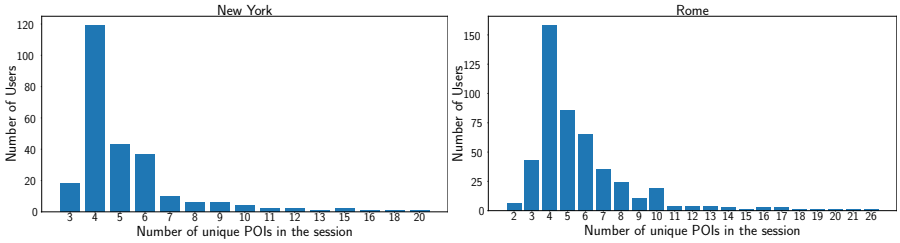


Fig. 6: Frequency of the number of venues each user visited in test for New York and Rome after removing repeated items in a per-user basis.

(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.

7.2.4 Short summary

Overall, these results confirm that it is possible to improve the performance of algorithms by reranking the recommendations (**RQ2**), 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 (FP_s), while keeping the same (or lower but very close) item-level sequential relevance ($NDCG_s$). 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.

7.3 Discussion

In previous sections, we have presented and analyzed the performance of different recommendation techniques applied to the task of route recommendation. We now aim to discuss in more detail some aspects that may have a large impact on the evaluation of these techniques. 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 (λ in Equation 1) on

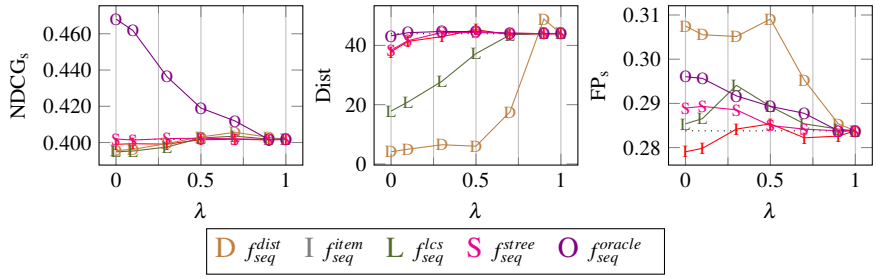


Fig. 7: Sensitivity to λ parameter on three evaluation dimensions ($NDCG_s$@10 left, Dist@5 middle, FP_s@10 right) for the Pop recommender using the following rerankers (with markers) on New York: f^{dist}_{seq}, f^{item}_{seq}, f^{lcs}_{seq}, f^{stree}_{seq}, and f^{oracle}_{seq}. The performance for the baseline (no reranker) is included as the horizontal dashed line.

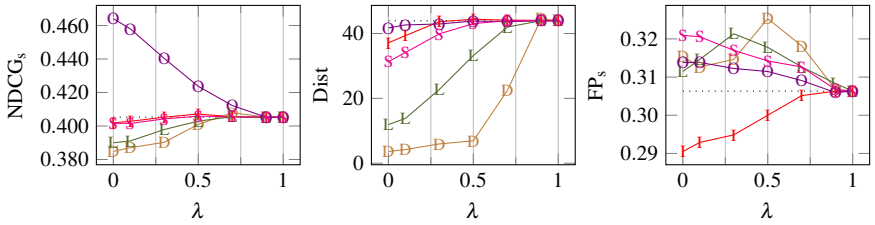


Fig. 8: Same information as in Figure 7 but for the IRenMF recommender.

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 paper, we moved those additional results to Appendix A.3

Figure 5 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 6.2, is 4), this effect is further emphasized when repeated venues in the test set are removed (see Figure 6), 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.

Moreover, we show in Figures 7 and 8 the evolution of different λ values to analyze the impact of the sequence-aware reranker component according to Equation 1 (recall that the results presented in the previous sections were for $\lambda = 0$). Figure 7 shows the results for the Pop recommender on New York whereas Figure 8 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 λ, 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 λ to obtain a tradeoff in all the dimensions would be, as expected, $\lambda = 0.5$. This value would allow us, for instance, to improve 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 $NDCG_s$ (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).

8 Conclusions and future work

In this paper we have presented an analysis on the performance of a comprehensive set of recommenders from different families in the area of recommending sequences of venues (routes) obtained from user check-ins. Firstly, we have defined a generic framework to obtain routes from independent check-ins using different constraints, which can actually be applied to other domains as long as temporal information is available. We have also formalized a set of sequential metrics integrating the Longest Common Subsequence (LCS) algorithm into ranking evaluation metrics. Our third main contribution is the proposal, adaptation, and experimentation of several reranking strategies – originally used within the scope of classical recommendation to increase the diversity of the results – to generate more meaningful sequences of venues.

Our results show that those recommendation approaches that use temporal or geographical information are capable of obtaining better results in terms of relevance. This is in line with results from the literature, however, this is the first work, to the best of our knowledge, where so many algorithms from different families have been tested under a common, realistic evaluation framework. Furthermore, we have also shown that it is possible to define sequence-aware ranking metrics, in such a way that those recommenders suggesting the venues in the same order as the user test are rewarded. However, due to the high sparsity of POI datasets, we have observed that it is more illustrative to analyze these metrics not only at the item level (i.e., matching exactly

the recommended venues) but also at the category level, extending and integrating already defined metrics in the literature.

On the other hand, the proposed reranking techniques have shown their usefulness at improving the quality of the recommendations provided originally by the algorithms, aiming, in principle, to provide more meaningful sequences for the users. Although in terms of traditional performance, measured by NDCG or its sequential counterpart, they have not achieved a substantial improvement with respect to the recommendations without reranking, they have demonstrated that by reordering a small number of items (20 in the reported experiments) from the recommendation list, we can enrich the performance of the recommenders in terms of distance and category accuracy.

Nonetheless, the task of route recommendation is far from being solved, and our experiments evidence several aspects that should be addressed in the future. First, the high degree of sparsity in these datasets produces that simple models often work better or at the same level than more complex algorithms; this effect is emphasized by the well-known popularity bias and the repetition bias, an experimental setting that is often ignored in the area but that in venue and route recommendation tasks has a strong impact on the results, since users tend to visit the same venues more than once; in both cases, simple baselines (either returning the most popular items or those venues previously seen by the user) achieve very good results. Moreover, sparsity has a large impact on related evaluation problems such as cold-start or new users or items; hence, a thorough investigation on this issue should be devoted in the context of sequence-aware evaluation. Second, we have shown that alternative evaluation criteria beyond relevance, such as geographical distance, should be considered if we expect the recommended venues to be followed by the user; however, it is a challenging problem to provide relevant but close venues, as we have presented some algorithms based only on geographical information that recommend very short but not relevant enough routes, while other methods suggest longer routes that fit better the user preferences. For this problem, we have experimented with our proposed reranking strategies, with some positive and promising results, but we envision this could be an area where new methods, probably based on combining existing approaches, could optimize all these criteria at the same time.

Finally, the fields of venue and route recommendation would clearly benefit from having realistic datasets made with routes followed by actual tourists in a particular city; so far, most of the works in the area (including ours) transform in some way datasets based on check-ins at different LBSNs, however, as we have discussed, this process is not perfect and it is difficult to neutralize the inherent biases in these systems (Papalexakis et al 2014; Zhang et al 2018). At the same time, it is not clear how many of the users of LBSNs are actually tourists, since the performance and movemement patterns of users categorized as tourists or locals may differ (Le et al 2014). We, as a surrogate, decided to experiment with datasets coming from different sources and domains to prevent this effect, but a more realistic dataset would allow to experiment and draw conclusions closer to the final users of any venue recommender system. With the growing proliferation of datasets with cellphone, taxi, or bus usage throughout cities (Zhang et al 2015b), new opportunities to collect such realistic datasets open up. Once several information sources become available to produce this type of datasets, it will be interesting to compare – in contrast or in addition to the

Table 11: Performance for all the cities, only showing the best recommender for each family. Notation and cutoffs like in Table 8.

			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	BPR	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
	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	BPR	0.128	0.386	†**0.389**	0.126	0.374	†**0.375**	22.7	0.001	0.852
	Temporal	BF	†**0.131**	0.388	0.378	†**0.129**	0.376	0.363	24.6	0.002	0.877
	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	BPR	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	BPR	†**0.131**	†**0.418**	0.274	0.128	0.408	0.270	30.0	0.001	0.917
	Temporal	BF	0.130	0.416	†**0.328**	†**0.129**	†**0.409**	†**0.310**	†**24.7**	0.004	†**0.938**
	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**

methodology followed here – the performance of several recommender systems on the same city, since in that situation the observations gathered by each dataset could be (potentially) very different.

Acknowledgements This work has been funded by the Ministerio de Ciencia, Innovación y Universidades (reference: TIN2016-80630-P) and by the European Social Fund (ESF), within the 2017 call for predoctoral contracts. The authors thank the reviewers for their thoughtful comments and suggestions.

A Appendix

A.1 Complete performance results of venue recommender systems

Table 11 shows the results in the four cities for the best recommender for each family (we do not show the performance of all recommenders in all cities due to space constraints). 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 BPR 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 BPR approach, on the other hand, is tailored to optimize the ranking by modeling the implicit 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 Complete performance results of reranking strategies

In Tables 12, 13, 14, 15, and 16 we present the complete results for the same reranker strategies as in Table 10, 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 the paper) is the following: the distance of the route obtained by the recommender from the Basic family (Pop) in PJ 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 by balancing a tradeoff between relevance and the rest of the dimensions.

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 $NDCG_s$ and FP_s (see for example the results in PJ for most of the families, but especially for Tour) or at least in one of the metrics (for example, in NYC this reranker is the best in terms of FP_s). The performance of the rerankers based on subsequences (f_{seq}^{lcs} and f_{seq}^{stree}) actually depends on each dataset, although it seems they work better in PJ; 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 PJ, 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.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 the paper (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 17, by allowing the algorithms to recommend items the user has previously interacted with – we denote this methodology as Items in Train (as opposed to Train Items), 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 Train Items 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 (Train Items against Items in Train) 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 BPR in Tokyo, UB is the best algorithm in the Classic family, whereas in Petaling Jaya, the switch is between BPR and BF to UB and Caser.

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 (Items in Train) 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 (Train Items), 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).

References

Abdollahpouri H, Burke R, Mobasher B (2019) Managing popularity bias in recommender systems with personalized re-ranking. In: Barták R, Brawner KW (eds) Proceedings of the Thirty-Second International Florida Artificial Intelligence Research Society Conference, Sarasota, Florida, USA, May 19-22 2019., AAAI Press, pp 413–418, URL https://aaai.org/ocs/index.php/FLAIRS/FLAIRS19/paper/view/18199

Aiolli F (2013) Efficient top-n recommendation for very large scale binary rated datasets. In: Yang Q, King I, Li Q, Pu P, Karypis G (eds) Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013, ACM, pp 273–280, DOI 10.1145/2507157.2507189, URL http://doi.acm.org/10.1145/2507157.2507189

Apostolico A (1997) String editing and longest common subsequences. In: Rozenberg G, Salomaa A (eds) Handbook of Formal Languages: Volume 2. Linear Modeling: Background and Application, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 361–398, DOI 10.1007/978-3-662-07675-0_8, URL http://dx.doi.org/10.1007/978-3-662-07675-0_8

Ayala VAA, Gülsen KC, Alzogbi A, Färber M, Muñiz M, Lausen G (2017) A delay-robust touristic plan recommendation using real-world public transportation information. In: Neidhardt J, Fesenmaier DR, Kuflik T, Wörndl W (eds) 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., CEUR-WS.org, CEUR Workshop Proceedings, vol 1906, pp 9–17, URL http://ceur-ws.org/Vol-1906/paper2.pdf

Balabanovic M, Shoham Y (1997) Content-based, collaborative recommendation. Commun ACM 40(3):66–72, DOI 10.1145/245108.245124, URL http://doi.acm.org/10.1145/245108.245124

Bellogín A, Sánchez P (2017a) Collaborative filtering based on subsequence matching: A new approach. Inf Sci 418:432–446, DOI 10.1016/j.ins.2017.08.016, URL https://doi.org/10.1016/j.ins.2017.08.016

Bellogín A, Sánchez P (2017b) Revisiting neighbourhood-based recommenders for temporal scenarios. In: Bieliková M, Bogina V, Kuflik T, Sasson R (eds) Proceedings of the 1st Workshop on Temporal Reasoning in Recommender Systems co-located with 11th International Conference on Recommender Systems (RecSys 2017), Como, Italy, August 27-31, 2017., CEUR-WS.org, CEUR Workshop Proceedings, vol 1922, pp 40–44, URL http://ceur-ws.org/Vol-1922/paper8.pdf

Bellogín A, Castells P, Cantador I (2017) Statistical biases in information retrieval metrics for recommender systems. Inf Retr Journal 20(6):606–634, DOI 10.1007/s10791-017-9312-z, URL https://doi.org/10.1007/s10791-017-9312-z

Ben-Shimon D, Tsikinovsky A, Friedmann M, Shapira B, Rokach L, Hoerle J (2015) Recsys challenge 2015 and the YOOCHOOSE dataset. In: Werthner H, Zanker M, Golbeck J, Semeraro G (eds) Proceedings of the 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria, September 16-20, 2015, ACM, pp 357–358, URL https://dl.acm.org/citation.cfm?id=2798723

Bothorel C, Lathia N, Picot-Clémente R, Noulas A (2018) Location recommendation with social media data. In: Brusilovsky P, He D (eds) Social Information Access - Systems and Technologies, Lecture

Notes in Computer Science, vol 10100, Springer, pp 624–653, DOI 10.1007/978-3-319-90092-6_16, URL https://doi.org/10.1007/978-3-319-90092-6_16

Braunhofer M, Elahi M, Ricci F, Schievenin T (2014) Context-aware points of interest suggestion with dynamic weather data management. In: Xiang Z, Tussyadiah I (eds) Information and Communication Technologies in Tourism 2014, ENTER 2014, Proceedings of the International Conference in Dublin, Ireland, January 21-24, 2014, Springer, pp 87–100

Brilhante IR, de Macêdo JAF, Nardini FM, Perego R, Renso C (2013) Where shall we go today?: planning touristic tours with tripbuilder. In: He Q, Iyengar A, Nejdl W, Pei J, Rastogi R (eds) 22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013, ACM, pp 757–762, DOI 10.1145/2505515.2505643, URL https://doi.org/10.1145/2505515.2505643

Burke R, O'Mahony MP, Hurley NJ (2015) Robust collaborative recommendation. In: Ricci F, Rokach L, Shapira B (eds) Recommender Systems Handbook, Springer, pp 961–995, DOI 10.1007/978-1-4899-7637-6_28, URL https://doi.org/10.1007/978-1-4899-7637-6_28

Campos PG, Díez F, Cantador I (2014) Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. User Model User-Adapt Interact 24(1-2):67–119, DOI 10.1007/s11257-012-9136-x, URL https://doi.org/10.1007/s11257-012-9136-x

Cañamares R, Castells P (2017) A probabilistic reformulation of memory-based collaborative filtering: Implications on popularity biases. In: Kando N, Sakai T, Joho H, Li H, de Vries AP, White RW (eds) Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017, ACM, pp 215–224, DOI 10.1145/3077136.3080836, URL http://doi.acm.org/10.1145/3077136.3080836

Castells P, Hurley NJ, Vargas S (2015) Novelty and diversity in recommender systems. In: Ricci F, Rokach L, Shapira B (eds) Recommender Systems Handbook, Springer, pp 881–918, DOI 10.1007/978-1-4899-7637-6_26, URL https://doi.org/10.1007/978-1-4899-7637-6_26

Chen D, Ong CS, Xie L (2016) Learning points and routes to recommend trajectories. In: Mukhopadhyay S, Zhai C, Bertino E, Crestani F, Mostafa J, Tang J, Si L, Zhou X, Chang Y, Li Y, Sondhi P (eds) Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016, ACM, pp 2227–2232, DOI 10.1145/2983323.2983672, URL http://doi.acm.org/10.1145/2983323.2983672

Cho E, Myers SA, Leskovec J (2011) Friendship and mobility: user movement in location-based social networks. In: Apté C, Ghosh J, Smyth P (eds) Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011, ACM, pp 1082–1090, DOI 10.1145/2020408.2020579, URL https://doi.org/10.1145/2020408.2020579

Choudhury MD, Feldman M, Amer-Yahia S, Golbandi N, Lempel R, Yu C (2010) Automatic construction of travel itineraries using social breadcrumbs. In: Chignell MH, Toms EG (eds) HT'10, Proceedings of the 21st ACM Conference on Hypertext and Hypermedia, Toronto, Ontario, Canada, June 13-16, 2010, ACM, pp 35–44, DOI 10.1145/1810617.1810626, URL http://doi.acm.org/10.1145/1810617.1810626

Fang SH, Lu EH, Tseng VS (2014) Trip recommendation with multiple user constraints by integrating point-of-interests and travel packages. In: Zaslavsky AB, Chrysanthis PK, Becker C, Indulska J, Mokbel MF, Nicklas D, Chow C (eds) IEEE 15th International Conference on Mobile Data Management, MDM 2014, Brisbane, Australia, July 14-18, 2014 - Volume 1, IEEE Computer Society, pp 33–42, DOI 10.1109/MDM.2014.10, URL https://doi.org/10.1109/MDM.2014.10

Gantner Z, Rendle S, Freudenthaler C, Schmidt-Thieme L (2011) Mymedialite: a free recommender system library. In: Mobasher B, Burke RD, Jannach D, Adomavicius G (eds) Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011, ACM, pp 305–308, DOI 10.1145/2043932.2043989, URL http://doi.acm.org/10.1145/2043932.2043989

Gao H, Tang J, Hu X, Liu H (2013) Exploring temporal effects for location recommendation on location-based social networks. In: Yang Q, King I, Li Q, Pu P, Karypis G (eds) Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013, ACM, pp 93–100, DOI 10.1145/2507157.2507182, URL http://doi.acm.org/10.1145/2507157.2507182

Gavalas D, Konstantopoulos C, Mastakas K, Pantziou GE (2014) A survey on algorithmic approaches for solving tourist trip design problems. J Heuristics 20(3):291–328, DOI 10.1007/s10732-014-9242-5, URL https://doi.org/10.1007/s10732-014-9242-5

de Gemmis M, Lops P, Musto C, Narducci F, Semeraro G (2015) Semantics-aware content-based recommender systems. In: Ricci F, Rokach L, Shapira B (eds) Recommender Systems Handbook, Springer, pp 119–159, DOI 10.1007/978-1-4899-7637-6_4, URL https://doi.org/10.1007/978-1-4899-7637-6_4

Gunawan A, Lau HC, Vansteenwegen P (2016) Orienteering problem: A survey of recent variants, solution approaches and applications. European Journal of Operational Research 255(2):315–332, DOI 10.1016/j.ejor.2016.04.059, URL https://doi.org/10.1016/j.ejor.2016.04.059

Gunawardana A, Shani G (2015) Evaluating recommender systems. In: Ricci F, Rokach L, Shapira B (eds) Recommender Systems Handbook, Springer, pp 265–308, DOI 10.1007/978-1-4899-7637-6_8, URL https://doi.org/10.1007/978-1-4899-7637-6_8

Guo G, Zhang J, Sun Z, Yorke-Smith N (2015) Librec: A java library for recommender systems. In: Cristea AI, Masthoff J, Said A, Tintarev N (eds) 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., CEUR-WS.org, CEUR Workshop Proceedings, vol 1388, URL http://ceur-ws.org/Vol-1388/demo_paper1.pdf

Gusfield D (1997) Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology. Cambridge University Press, DOI 10.1017/cbo9780511574931, URL https://doi.org/10.1017/cbo9780511574931

Harper FM, Konstan JA (2016) The movielens datasets: History and context. TiiS 5(4):19:1–19:19, DOI 10.1145/2827872, URL http://doi.acm.org/10.1145/2827872

He J, Li X, Liao L (2017) Category-aware next point-of-interest recommendation via listwise bayesian personalized ranking. In: Sierra C (ed) Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017, ijcai.org, pp 1837–1843, DOI 10.24963/ijcai.2017/255, URL https://doi.org/10.24963/ijcai.2017/255

He R, McAuley J (2016) Fusing similarity models with markov chains for sparse sequential recommendation. In: Bonchi F, Domingo-Ferrer J, Baeza-Yates RA, Zhou Z, Wu X (eds) IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain, IEEE, pp 191–200, DOI 10.1109/ICDM.2016.0030, URL https://doi.org/10.1109/ICDM.2016.0030

Hu Y, Koren Y, Volinsky C (2008) 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, IEEE Computer Society, pp 263–272, DOI 10.1109/ICDM.2008.22, URL https://doi.org/10.1109/ICDM.2008.22

Jannach D, Lerche L, Kamehkhosh I, Jugovac M (2015) What recommenders recommend: an analysis of recommendation biases and possible countermeasures. User Model User-Adapt Interact 25(5):427–491, DOI 10.1007/s11257-015-9165-3, URL https://doi.org/10.1007/s11257-015-9165-3

Jansen BJ, Spink A, Blakely C, Koshman S (2007) Defining a session on web search engines. JASIST 58(6):862–871, DOI 10.1002/asi.20564, URL https://doi.org/10.1002/asi.20564

Jeung H, Yiu ML, Jensen CS (2011) Trajectory pattern mining. In: Zheng Y, Zhou X (eds) Computing with Spatial Trajectories, Springer, pp 143–177, DOI 10.1007/978-1-4614-1629-6_5, URL https://doi.org/10.1007/978-1-4614-1629-6_5

Kaminskas M, Bridge D (2017) Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems. TiiS 7(1):2:1–2:42, DOI 10.1145/2926720, URL https://doi.org/10.1145/2926720

Kumar G, Jerbi H, O'Mahony MP (2017) Towards the recommendation of personalised activity sequences in the tourism domain. In: Neidhardt J, Fesenmaier DR, Kuflik T, Wörndl W (eds) 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., CEUR-WS.org, CEUR Workshop Proceedings, vol 1906, pp 26–30, URL http://ceur-ws.org/Vol-1906/paper4.pdf

Laß C, Herzog D, Wörndl W (2017) Context-aware tourist trip recommendations. In: Neidhardt J, Fesenmaier DR, Kuflik T, Wörndl W (eds) 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., CEUR-WS.org, CEUR Workshop Proceedings, vol 1906, pp 18–25, URL http://ceur-ws.org/Vol-1906/paper3.pdf

Le A, Pelechrinis K, Krishnamurthy P (2014) Country-level spatial dynamics of user activity: a case study in location-based social networks. In: Menczer F, Hendler J, Dutton WH, Strohmaier M, Cattuto C, Meyer ET (eds) ACM Web Science Conference, WebSci '14, Bloomington, IN, USA, June 23-26, 2014, ACM, pp 71–80, DOI 10.1145/2615569.2615689, URL https://doi.org/10.1145/2615569.2615689

Li X, Cong G, Li X, Pham TN, Krishnaswamy S (2015) Rank-geofm: A ranking based geographical factorization method for point of interest recommendation. In: Baeza-Yates RA, Lalmas M, Moffat A, Ribeiro-Neto BA (eds) Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015, ACM, pp 433–442, DOI 10.1145/2766462.2767722, URL http://doi.acm.org/10.1145/2766462.2767722

Li X, Jiang M, Hong H, Liao L (2017) A time-aware personalized point-of-interest recommendation via high-order tensor factorization. ACM Trans Inf Syst 35(4):31:1–31:23, DOI 10.1145/3057283, URL http://doi.acm.org/10.1145/3057283

Lian D, Zhao C, Xie X, Sun G, Chen E, Rui Y (2014) Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation. In: Macskassy SA, Perlich C, Leskovec J, Wang W, Ghani R (eds) The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014, ACM, pp 831–840, DOI 10.1145/2623330.2623638, URL http://doi.acm.org/10.1145/2623330.2623638

Lim KH, Chan J, Leckie C, Karunasekera S (2015) Personalized tour recommendation based on user interests and points of interest visit durations. In: Yang Q, Wooldridge M (eds) Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015, AAAI Press, pp 1778–1784, URL http://ijcai.org/Abstract/15/253

Lim KH, Chan J, Leckie C, Karunasekera S (2018) Personalized trip recommendation for tourists based on user interests, points of interest visit durations and visit recency. Knowl Inf Syst 54(2):375–406, DOI 10.1007/s10115-017-1056-y, URL https://doi.org/10.1007/s10115-017-1056-y

Liu G, Nguyen TT, Zhao G, Zha W, Yang J, Cao J, Wu M, Zhao P, Chen W (2016) Repeat buyer prediction for e-commerce. In: Krishnapuram B, Shah M, Smola AJ, Aggarwal CC, Shen D, Rastogi R (eds) Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, ACM, pp 155–164, DOI 10.1145/2939672.2939674, URL https://doi.org/10.1145/2939672.2939674

Liu Y, Wei W, Sun A, Miao C (2014) Exploiting geographical neighborhood characteristics for location recommendation. In: Li J, Wang XS, Garofalakis MN, Soboroff I, Suel T, Wang M (eds) Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014, ACM, pp 739–748, DOI 10.1145/2661829.2662002, URL http://doi.acm.org/10.1145/2661829.2662002

Liu Y, Pham T, Cong G, Yuan Q (2017) An experimental evaluation of point-of-interest recommendation in location-based social networks. PVLDB 10(10):1010–1021, URL http://www.vldb.org/pvldb/vol10/p1010-liu.pdf

Maillet F, Eck D, Desjardins G, Lamere P (2009) Steerable playlist generation by learning song similarity from radio station playlists. In: Hirata K, Tzanetakis G, Yoshii K (eds) Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, Kobe International Conference Center, Kobe, Japan, October 26-30, 2009, International Society for Music Information Retrieval, pp 345–350, URL http://ismir2009.ismir.net/proceedings/OS4-2.pdf

Menon AK, Chen D, Xie L, Ong CS (2017) Revisiting revisits in trajectory recommendation. In: Yang J, Sun Z, Bozzon A, Zhang J, Larson M (eds) Proceedings of International Workshop on Citizens for Recommender Systems, CitRec@RecSys 2017, 31 August 2017, Como, Italy, ACM, pp 2:1–2:6, DOI 10.1145/3127325.3127326, URL http://doi.acm.org/10.1145/3127325.3127326

Miller HJ (2004) Tobler's first law and spatial analysis. Annals of the Association of American Geographers 94(2):284–289, DOI 10.1111/j.1467-8306.2004.09402005.x, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8306.2004.09402005.x

Monti D, Palumbo E, Rizzo G, Morisio M (2018) Sequeval: A framework to assess and benchmark sequence-based recommender systems. CoRR abs/1810.04956, URL http://arxiv.org/abs/1810.04956, 1810.04956

Ning X, Desrosiers C, Karypis G (2015) A comprehensive survey of neighborhood-based recommendation methods. In: Ricci F, Rokach L, Shapira B (eds) Recommender Systems Handbook, Springer, pp 37–76, DOI 10.1007/978-1-4899-7637-6_2, URL https://doi.org/10.1007/978-1-4899-7637-6_2

Palumbo E, Rizzo G, Troncy R, Baralis E (2017) Predicting your next stop-over from location-based social network data with recurrent neural networks. In: Neidhardt J, Fesenmaier DR, Kuflik T, Wörndl W (eds) 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., CEUR-WS.org, CEUR Workshop Proceedings, vol 1906, pp 1–8, URL http://ceur-ws.org/Vol-1906/paper1.pdf

Papalexakis EE, Pelechrinis K, Faloutsos C (2014) Spotting misbehaviors in location-based social networks using tensors. In: Chung C, Broder AZ, Shim K, Suel T (eds) 23rd International World Wide Web

Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014, Companion Volume, ACM, pp 551–552, DOI 10.1145/2567948.2576950, URL https://doi.org/10.1145/2567948.2576950

Quadrana M, Cremonesi P, Jannach D (2018) Sequence-aware recommender systems. In: Mitrovic T, Zhang J, Chen L, Chin D (eds) Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization, UMAP 2018, Singapore, July 08-11, 2018, ACM, pp 373–374, DOI 10.1145/3209219.3209270, URL http://doi.acm.org/10.1145/3209219.3209270

Radlinski F, Dumais ST (2006) Improving personalized web search using result diversification. In: Efthimiadis EN, Dumais ST, Hawking D, Järvelin K (eds) 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, ACM, pp 691–692, DOI 10.1145/1148170.1148320, URL https://doi.org/10.1145/1148170.1148320

Renda ME, Straccia U (2003) Web metasearch: Rank vs. score based rank aggregation methods. In: Lamont GB, Haddad H, Papadopoulos GA, Panda B (eds) Proceedings of the 2003 ACM Symposium on Applied Computing (SAC), March 9-12, 2003, Melbourne, FL, USA, ACM, pp 841–846, DOI 10.1145/952532.952698, URL http://doi.acm.org/10.1145/952532.952698

Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2009) BPR: bayesian personalized ranking from implicit feedback. In: Bilmes JA, Ng AY (eds) UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009, AUAI Press, pp 452–461, URL https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=1630&proceeding_id=25

Rendle S, Freudenthaler C, Schmidt-Thieme L (2010) Factorizing personalized markov chains for next-basket recommendation. In: Rappa M, Jones P, Freire J, Chakrabarti S (eds) Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010, ACM, pp 811–820, DOI 10.1145/1772690.1772773, URL http://doi.acm.org/10.1145/1772690.1772773

Said A, Bellogín A (2014) Comparative recommender system evaluation: benchmarking recommendation frameworks. In: Kobsa A, Zhou MX, Ester M, Koren Y (eds) Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014, ACM, pp 129–136, DOI 10.1145/2645710.2645746, URL http://doi.acm.org/10.1145/2645710.2645746

Said A, Bellogín A (2015) Replicable evaluation of recommender systems. In: Werthner H, Zanker M, Golbeck J, Semeraro G (eds) Proceedings of the 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria, September 16-20, 2015, ACM, pp 363–364, URL https://dl.acm.org/citation.cfm?id=2792841

Sánchez P, Bellogín A (2018a) Challenges on evaluating venue recommendation approaches: Position paper. In: Neidhardt J, Wörndl W, Kuflik T, Zanker M (eds) 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., CEUR-WS.org, CEUR Workshop Proceedings, vol 2222, pp 37–40, URL http://ceur-ws.org/Vol-2222/paper8.pdf

Sánchez P, Bellogín A (2018b) Time-aware novelty metrics for recommender systems. In: Pasi G, Piwowarski B, Azzopardi L, Hanbury A (eds) Advances in Information Retrieval - 40th European Conference on IR Research, ECIR 2018, Grenoble, France, March 26-29, 2018, Proceedings, Springer, Lecture Notes in Computer Science, vol 10772, pp 357–370, DOI 10.1007/978-3-319-76941-7_27, URL https://doi.org/10.1007/978-3-319-76941-7_27

Sánchez P, Bellogín A (2019) Building user profiles based on sequences for content and collaborative filtering. Inf Process Manage 56(1):192–211, DOI 10.1016/j.ipm.2018.10.003, URL https://doi.org/10.1016/j.ipm.2018.10.003

Santos RLT, Macdonald C, Ounis I (2010) Exploiting query reformulations for web search result diversification. In: Rappa M, Jones P, Freire J, Chakrabarti S (eds) Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010, ACM, pp 881–890, DOI 10.1145/1772690.1772780, URL https://doi.org/10.1145/1772690.1772780

Santos RLT, MacDonald C, Ounis I (2015) Search result diversification. Foundations and Trends in Information Retrieval 9(1):1–90, DOI 10.1561/1500000040, URL https://doi.org/10.1561/1500000040

Spiliopoulou M, Mobasher B, Berendt B, Nakagawa M (2003) A framework for the evaluation of session reconstruction heuristics in web-usage analysis. INFORMS Journal on Computing 15(2):171–190, DOI 10.1287/ijoc.15.2.171.14445, URL https://doi.org/10.1287/ijoc.15.2.171.14445

Tang J, Wang K (2018) Personalized top-n sequential recommendation via convolutional sequence embedding. In: Chang Y, Zhai C, Liu Y, Maarek Y (eds) Proceedings of the Eleventh ACM International

Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018, ACM, pp 565–573, DOI 10.1145/3159652.3159656, URL https://doi.org/10.1145/3159652.3159656

Trattner C, Oberegger A, Marinho LB, Parra D (2018) Investigating the utility of the weather context for point of interest recommendations. J of IT & Tourism 19(1-4):117–150, DOI 10.1007/s40558-017-0100-9, URL https://doi.org/10.1007/s40558-017-0100-9

Vargas S (2015) Novelty and diversity evaluation and enhancement in recommender systems. PhD thesis, PhD thesis, Universidad Autónoma de Madrid, Spain

Vargas S, Castells P (2011) Rank and relevance in novelty and diversity metrics for recommender systems. In: Mobasher B, Burke RD, Jannach D, Adomavicius G (eds) Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011, ACM, pp 109–116, DOI 10.1145/2043932.2043955, URL http://doi.acm.org/10.1145/2043932.2043955

Vargas S, Castells P (2014) Improving sales diversity by recommending users to items. In: Kobsa A, Zhou MX, Ester M, Koren Y (eds) Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014, ACM, pp 145–152, DOI 10.1145/2645710.2645744, URL https://doi.org/10.1145/2645710.2645744

Vargas S, Castells P, Vallet D (2011) Intent-oriented diversity in recommender systems. In: Ma W, Nie J, Baeza-Yates RA, Chua T, Croft WB (eds) Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011, ACM, pp 1211–1212, DOI 10.1145/2009916.2010124, URL https://doi.org/10.1145/2009916.2010124

Wasilewski J, Hurley N (2018) Intent-aware item-based collaborative filtering for personalised diversification. In: Mitrovic T, Zhang J, Chen L, Chin D (eds) Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization, UMAP 2018, Singapore, July 08-11, 2018, ACM, pp 81–89, DOI 10.1145/3209219.3209234, URL https://doi.org/10.1145/3209219.3209234

Yang D, Zhang D, Qu B (2016) Participatory cultural mapping based on collective behavior data in location-based social networks. ACM TIST 7(3):30:1–30:23, DOI 10.1145/2814575, URL http://doi.acm.org/10.1145/2814575

Ye M, Yin P, Lee W, Lee DL (2011) Exploiting geographical influence for collaborative point-of-interest recommendation. In: Ma W, Nie J, Baeza-Yates RA, Chua T, Croft WB (eds) Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011, ACM, pp 325–334, DOI 10.1145/2009916.2009962, URL http://doi.acm.org/10.1145/2009916.2009962

Yu Z, Xu H, Yang Z, Guo B (2016) Personalized travel package with multi-point-of-interest recommendation based on crowdsourced user footprints. IEEE Trans Human-Machine Systems 46(1):151–158, DOI 10.1109/THMS.2015.2446953, URL https://doi.org/10.1109/THMS.2015.2446953

Zamani H, Schedl M, Lamere P, Chen C (2018) An analysis of approaches taken in the ACM recsys challenge 2018 for automatic music playlist continuation. CoRR abs/1810.01520, URL http://arxiv.org/abs/1810.01520, 1810.01520

Zhang C, Liang H, Wang K, Sun J (2015a) Personalized trip recommendation with POI availability and uncertain traveling time. In: Bailey J, Moffat A, Aggarwal CC, de Rijke M, Kumar R, Murdock V, Sellis TK, Yu JX (eds) Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015, ACM, pp 911–920, DOI 10.1145/2806416.2806558, URL http://doi.acm.org/10.1145/2806416.2806558

Zhang D, Zhao J, Zhang F, He T (2015b) Urbancps: a cyber-physical system based on multi-source big infrastructure data for heterogeneous model integration. In: Bayen AM, Branicky MS (eds) Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems, ICCPS 2015, Seattle, WA, USA, April 14-16, 2015, ACM, pp 238–247, DOI 10.1145/2735960.2735985, URL https://doi.org/10.1145/2735960.2735985

Zhang J, Chow C (2013) igslr: personalized geo-social location recommendation: a kernel density estimation approach. In: Knoblock CA, Schneider M, Kröger P, Krumm J, Widmayer P (eds) 21st SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2013, Orlando, FL, USA, November 5-8, 2013, ACM, pp 324–333, DOI 10.1145/2525314.2525339, URL http://doi.acm.org/10.1145/2525314.2525339

Zhang J, Chow C, Li Y (2014) LORE: exploiting sequential influence for location recommendations. In: Huang Y, Schneider M, Gertz M, Krumm J, Sankaranarayanan J (eds) Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Dallas/Fort Worth, TX, USA, November 4-7, 2014, ACM, pp 103–112, DOI 10.1145/2666310.2666400, URL

http://doi.acm.org/10.1145/2666310.2666400
Zhang K, Pelechrinis K, Lappas T (2018) Effects of promotions on location-based social media: Evidence from foursquare. Int J Electronic Commerce 22(1):36–65, DOI 10.1080/10864415.2018.1396118, URL https://doi.org/10.1080/10864415.2018.1396118
Zhao S, Lyu MR, King I (2018a) Point-of-Interest Recommendation in Location-Based Social Networks. Springer Briefs in Computer Science, Springer, DOI 10.1007/978-981-13-1349-3, URL https://doi.org/10.1007/978-981-13-1349-3
Zhao WX, Zhou N, Sun A, Wen J, Han J, Chang EY (2018b) A time-aware trajectory embedding model for next-location recommendation. Knowl Inf Syst 56(3):559–579, DOI 10.1007/s10115-017-1107-4, URL https://doi.org/10.1007/s10115-017-1107-4
Ziegler C, McNee SM, Konstan JA, Lausen G (2005) Improving recommendation lists through topic diversification. In: Ellis A, Hagino T (eds) Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005, ACM, pp 22–32, DOI 10.1145/1060745.1060754, URL https://doi.org/10.1145/1060745.1060754

Author Biographies

Pablo Sánchez is a PhD student who is currently working in the Information Retrieval Group at Universidad Autónoma de Madrid (UAM), Spain, under the supervision of Dr. Alejandro Bellogín. He finished his Computer Science degree in 2016 and his Masters studies in 2017 at the same university; since then, he has been working in the areas of Information Retrieval and Recommender Systems, where he has published more than 10 papers so far in different journals and conferences.

Alejandro Bellogín is an Associate Professor at Universidad Autónoma de Madrid (UAM), Spain. Previously, he worked with Prof. Arjen de Vries associated to the Centrum Wiskunde & Informatica under a post-doctoral Marie Curie research grant. In 2012, he completed his PhD under the supervision of Prof. Pablo Castells and Dr. Iván Cantador at UAM. Dr. Bellogín has worked in several areas of user modeling and personalization, including recommender systems, evaluation, and reproducibility, where he has published over 50 conference and journal papers, while being involved in venues related to these areas, such as RecSys, WWW and UMAP, among others.

Table 12: Performance of the rerankers for the Basic family of recommenders. Same notation as in Table 8.

City	Reranker	Accuracy			Seq. Accuracy			Non Accuracy		
		P	NDCG	TFP	P_s	$NDCG_s$	FP_s	Dist	Gini	EPC
NYC	Baseline	0.144	0.417	0.326	0.139	0.402	0.284	43.9	0.001	0.904
	f^{rnd}_{seq}	0.132	0.396	0.324	0.128	0.383	0.297	28.0	▲**0.002**	0.935
	f^{dist}_{seq}	0.137	0.408	†**0.331**	0.134	0.396	▲**0.308**	▲**4.1**	0.001	▲**0.940**
	f^{feat}_{seq}	0.143	0.415	0.285	0.138	0.400	0.267	33.3	0.001	0.914
	f^{item}_{seq}	0.137	0.412	0.309	0.134	0.399	0.279	37.8	0.001	0.908
	f^{rec}_{seq}	†**0.147**	†**0.422**	0.331	†**0.142**	†**0.406**	0.298	42.4	0.001	0.913
	f^{lcs}_{seq}	0.137	0.406	0.313	0.136	0.395	0.285	17.8	0.001	0.918
	f^{stree}_{seq}	0.143	0.416	0.326	0.140	0.402	0.289	38.4	0.001	0.905
	f^{oracle}_{seq}	▲**0.163**	▲**0.479**	▲**0.332**	▲**0.163**	▲**0.468**	0.296	43.2	0.001	0.904
TOK	Baseline	0.127	0.385	†**0.385**	0.126	0.374	0.372	25.8	0.001	0.849
	f^{rnd}_{seq}	0.118	0.368	0.379	0.116	0.356	0.365	27.5	▲**0.001**	▲**0.894**
	f^{dist}_{seq}	0.122	0.378	0.383	0.121	0.367	▲**0.373**	▲**6.9**	0.001	0.893
	f^{feat}_{seq}	0.126	0.384	0.315	0.124	0.372	0.315	26.0	0.001	0.860
	f^{item}_{seq}	0.127	0.386	0.369	0.125	0.374	0.359	18.2	0.001	0.856
	f^{rec}_{seq}	†**0.129**	†**0.386**	0.358	†**0.127**	†**0.374**	0.346	24.5	0.001	0.858
	f^{lcs}_{seq}	0.123	0.378	0.351	0.122	0.367	0.344	10.4	0.001	0.880
	f^{stree}_{seq}	0.127	0.383	0.372	0.126	0.371	0.359	15.5	0.001	0.862
	f^{oracle}_{seq}	▲**0.138**	▲**0.419**	▲**0.386**	▲**0.138**	▲**0.408**	0.372	25.1	0.001	0.849
ROM	Baseline	0.227	0.508	0.517	0.202	0.447	0.464	5.0	0.050	0.848
	f^{rnd}_{seq}	0.186	0.449	0.511	0.171	0.402	0.452	5.9	▲**0.077**	▲**0.885**
	f^{dist}_{seq}	0.225	0.523	†**0.523**	0.211	0.469	†**0.474**	▲**1.4**	0.072	0.884
	f^{feat}_{seq}	0.199	0.476	0.423	0.181	0.422	0.371	5.0	0.048	0.876
	f^{item}_{seq}	†**0.232**	†**0.534**	0.520	†**0.214**	†**0.473**	0.469	1.8	0.067	0.872
	f^{rec}_{seq}	0.207	0.477	0.510	0.187	0.422	0.452	6.0	0.058	0.857
	f^{lcs}_{seq}	0.202	0.487	0.488	0.189	0.440	0.446	2.3	0.069	0.881
	f^{stree}_{seq}	0.218	0.502	0.520	0.198	0.446	0.466	3.2	0.063	0.861
	f^{oracle}_{seq}	▲**0.289**	▲**0.659**	▲**0.531**	▲**0.289**	▲**0.614**	▲**0.482**	4.2	0.052	0.850
PJ	Baseline	0.128	0.413	0.249	0.126	0.404	0.245	35.0	0.001	0.915
	f^{rnd}_{seq}	0.118	0.394	0.283	0.117	0.387	0.274	29.5	▲**0.002**	▲**0.939**
	f^{dist}_{seq}	†**0.132**	†**0.418**	▲**0.311**	†**0.130**	†**0.409**	▲**0.296**	▲**7.2**	0.002	0.937
	f^{feat}_{seq}	0.129	0.412	0.296	0.126	0.402	0.267	33.2	0.001	0.917
	f^{item}_{seq}	0.130	0.417	0.267	0.127	0.408	0.262	19.5	0.001	0.920
	f^{rec}_{seq}	0.131	0.415	0.277	0.129	0.407	0.271	26.3	0.002	0.924
	f^{lcs}_{seq}	0.130	0.413	0.293	0.128	0.403	0.274	14.8	0.002	0.926
	f^{stree}_{seq}	0.130	0.413	0.283	0.126	0.403	0.263	25.9	0.001	0.919
	f^{oracle}_{seq}	▲**0.144**	▲**0.463**	0.251	▲**0.144**	▲**0.456**	0.247	34.2	0.001	0.916

Table 13: Performance of the rerankers for the Classic family of recommenders. Same notation as in Table 8.

City	Reranker	Accuracy			Seq. Accuracy			Non Accuracy		
		P	NDCG	TFP	P_s	$NDCG_s$	FP_s	Dist	Gini	EPC
NYC	Baseline	0.145	0.418	0.330	0.141	0.404	0.285	45.3	0.001	0.905
	f^{rnd}_{seq}	0.130	0.393	0.320	0.127	0.382	0.292	30.3	▲**0.002**	0.935
	f^{dist}_{seq}	0.136	0.407	0.332	0.133	0.395	▲**0.309**	▲**4.2**	0.002	▲**0.941**
	f^{feat}_{seq}	0.142	0.413	0.287	0.137	0.398	0.267	33.5	0.002	0.915
	f^{item}_{seq}	0.138	0.413	0.307	0.136	0.400	0.276	38.0	0.001	0.908
	f^{rec}_{seq}	†**0.147**	†**0.422**	0.332	†**0.142**	†**0.406**	0.300	42.4	0.002	0.913
	f^{lcs}_{seq}	0.138	0.407	0.316	0.136	0.395	0.284	17.9	0.002	0.919
	f^{stree}_{seq}	0.145	0.418	▲**0.335**	0.141	0.404	0.294	38.6	0.001	0.906
	f^{oracle}_{seq}	▲**0.163**	▲**0.479**	0.334	▲**0.163**	▲**0.468**	0.300	44.3	0.001	0.905
TOK	Baseline	†**0.128**	†**0.386**	†**0.389**	†**0.126**	†**0.374**	▲**0.375**	22.7	0.001	0.852
	f^{rnd}_{seq}	0.116	0.366	0.371	0.115	0.356	0.357	27.0	▲**0.001**	▲**0.897**
	f^{dist}_{seq}	0.123	0.378	0.374	0.122	0.367	0.362	▲**6.8**	0.001	0.896
	f^{feat}_{seq}	0.125	0.384	0.315	0.124	0.372	0.315	26.0	0.001	0.860
	f^{item}_{seq}	0.127	0.385	0.370	0.126	0.373	0.360	17.4	0.001	0.856
	f^{rec}_{seq}	0.127	0.385	0.358	0.126	0.373	0.346	24.7	0.001	0.859
	f^{lcs}_{seq}	0.123	0.377	0.349	0.122	0.365	0.343	10.3	0.001	0.885
	f^{stree}_{seq}	0.127	0.382	0.371	0.124	0.369	0.360	13.9	0.001	0.866
	f^{oracle}_{seq}	▲**0.137**	▲**0.416**	▲**0.390**	▲**0.137**	▲**0.405**	0.375	22.4	0.001	0.852
ROM	Baseline	0.226	0.508	0.518	0.201	0.447	0.460	6.3	0.050	0.848
	f^{rnd}_{seq}	0.187	0.451	0.512	0.171	0.403	0.450	5.9	▲**0.077**	▲**0.886**
	f^{dist}_{seq}	0.223	0.521	†**0.525**	0.210	0.468	†**0.475**	▲**1.4**	0.071	0.884
	f^{feat}_{seq}	0.200	0.478	0.426	0.183	0.424	0.373	5.0	0.047	0.875
	f^{item}_{seq}	†**0.234**	†**0.536**	0.519	†**0.216**	†**0.476**	0.468	1.8	0.067	0.871
	f^{rec}_{seq}	0.207	0.477	0.509	0.187	0.422	0.452	6.0	0.058	0.857
	f^{lcs}_{seq}	0.200	0.487	0.487	0.188	0.440	0.447	2.3	0.068	0.880
	f^{stree}_{seq}	0.218	0.502	0.520	0.199	0.447	0.465	3.7	0.063	0.861
	f^{oracle}_{seq}	▲**0.287**	▲**0.657**	▲**0.530**	▲**0.287**	▲**0.612**	▲**0.482**	4.9	0.052	0.850
PJ	Baseline	0.131	0.418	0.274	0.128	0.408	0.270	30.0	0.001	0.917
	f^{rnd}_{seq}	0.124	0.402	0.288	0.122	0.394	0.278	30.7	▲**0.002**	▲**0.939**
	f^{dist}_{seq}	0.132	0.418	▲**0.309**	†**0.130**	†**0.410**	▲**0.294**	▲**7.4**	0.002	0.938
	f^{feat}_{seq}	0.128	0.411	0.298	0.125	0.402	0.269	34.0	0.001	0.918
	f^{item}_{seq}	0.130	†**0.419**	0.272	0.127	0.409	0.268	18.5	0.001	0.920
	f^{rec}_{seq}	0.131	0.415	0.279	0.129	0.407	0.273	26.5	0.002	0.924
	f^{lcs}_{seq}	†**0.132**	0.416	0.302	0.128	0.405	0.279	13.2	0.002	0.926
	f^{stree}_{seq}	0.131	0.415	0.297	0.128	0.405	0.275	22.1	0.001	0.919
	f^{oracle}_{seq}	▲**0.144**	▲**0.462**	0.275	▲**0.144**	▲**0.455**	0.269	29.0	0.001	0.917

Table 14: Performance of the rerankers for the Temporal family of recommenders. Same notation as in Table 8.

City	Reranker	Accuracy			Seq. Accuracy			Non Accuracy		
		P	NDCG	TFP	P_s	$NDCG_s$	FP_s	Dist	Gini	EPC
NYC	Baseline	0.145	0.418	0.321	0.140	0.404	0.284	43.2	0.001	0.904
	f^{rnd}_{seq}	0.131	0.395	0.325	0.130	0.385	0.300	30.3	▲**0.002**	0.935
	f^{dist}_{seq}	0.136	0.408	0.331	0.133	0.395	▲**0.308**	▲**4.1**	0.001	▲**0.940**
	f^{feat}_{seq}	0.143	0.414	0.284	0.138	0.399	0.266	33.3	0.001	0.914
	f^{item}_{seq}	0.138	0.413	0.309	0.135	0.399	0.277	37.7	0.001	0.907
	f^{rec}_{seq}	†**0.147**	†**0.422**	▲**0.332**	†**0.142**	†**0.406**	0.299	42.5	0.001	0.913
	f^{lcs}_{seq}	0.138	0.407	0.313	0.136	0.395	0.288	18.1	0.001	0.918
	f^{stree}_{seq}	0.145	0.418	0.327	0.141	0.404	0.294	38.2	0.001	0.905
	f^{oracle}_{seq}	▲**0.163**	▲**0.480**	0.328	▲**0.163**	▲**0.469**	0.296	42.8	0.001	0.904
TOK	Baseline	†**0.131**	†**0.388**	0.378	†**0.129**	†**0.376**	0.363	24.6	0.002	0.877
	f^{rnd}_{seq}	0.120	0.371	0.391	0.120	0.361	0.366	24.5	0.004	0.917
	f^{dist}_{seq}	0.125	0.378	▲**0.395**	0.123	0.367	▲**0.368**	▲**6.2**	▲**0.004**	▲**0.924**
	f^{feat}_{seq}	0.126	0.384	0.320	0.124	0.372	0.318	25.9	0.001	0.872
	f^{item}_{seq}	0.129	0.388	0.364	0.127	0.375	0.352	19.3	0.002	0.870
	f^{rec}_{seq}	0.129	0.388	0.370	0.128	0.376	0.357	25.1	0.002	0.871
	f^{lcs}_{seq}	0.125	0.379	0.361	0.123	0.367	0.350	9.3	0.003	0.905
	f^{stree}_{seq}	0.129	0.383	0.378	0.127	0.371	0.360	13.9	0.003	0.890
	f^{oracle}_{seq}	▲**0.139**	▲**0.420**	0.380	▲**0.139**	▲**0.409**	0.363	24.0	0.002	0.877
ROM	Baseline	0.227	0.508	0.516	0.201	0.447	†**0.469**	4.9	0.052	0.849
	f^{rnd}_{seq}	0.188	0.453	0.507	0.175	0.409	0.449	6.0	▲**0.080**	▲**0.887**
	f^{dist}_{seq}	0.221	0.518	†**0.520**	0.207	0.464	0.468	▲**1.4**	0.075	0.886
	f^{feat}_{seq}	0.199	0.476	0.431	0.180	0.421	0.375	5.0	0.050	0.874
	f^{item}_{seq}	†**0.234**	†**0.535**	0.518	†**0.214**	†**0.474**	0.465	1.9	0.068	0.871
	f^{rec}_{seq}	0.207	0.476	0.508	0.187	0.422	0.452	6.1	0.060	0.857
	f^{lcs}_{seq}	0.204	0.489	0.488	0.191	0.441	0.447	2.3	0.071	0.880
	f^{stree}_{seq}	0.218	0.501	0.519	0.198	0.445	0.468	3.1	0.065	0.863
	f^{oracle}_{seq}	▲**0.285**	▲**0.654**	▲**0.529**	▲**0.285**	▲**0.608**	▲**0.482**	4.1	0.054	0.851
PJ	Baseline	0.130	0.416	0.328	0.129	0.409	0.310	24.7	0.004	0.938
	f^{rnd}_{seq}	0.122	0.400	0.351	0.121	0.393	▲**0.329**	25.9	0.007	0.954
	f^{dist}_{seq}	0.127	0.413	▲**0.353**	0.126	0.405	0.327	▲**5.4**	▲**0.008**	▲**0.960**
	f^{feat}_{seq}	0.126	0.405	0.332	0.124	0.397	0.294	23.1	0.006	0.941
	f^{item}_{seq}	0.130	†**0.417**	0.308	0.128	†**0.410**	0.291	18.7	0.003	0.930
	f^{rec}_{seq}	†**0.132**	0.417	0.321	†**0.130**	0.408	0.308	25.4	0.004	0.936
	f^{lcs}_{seq}	0.127	0.410	0.342	0.125	0.401	0.319	9.8	0.007	0.953
	f^{stree}_{seq}	0.128	0.411	0.349	0.126	0.403	0.322	15.7	0.006	0.944
	f^{oracle}_{seq}	▲**0.143**	▲**0.460**	0.330	▲**0.143**	▲**0.453**	0.313	23.7	0.004	0.938

Table 15: Performance of the rerankers for the Geo family of recommenders. Same notation as in Table 8.

City	Reranker	Accuracy			Seq. Accuracy			Non Accuracy		
		P	NDCG	TFP	P_s	$NDCG_s$	FP_s	Dist	Gini	EPC
NYC	Baseline	†**0.147**	0.420	0.345	†**0.143**	0.405	0.306	43.9	0.002	0.916
	f^{rnd}_{seq}	0.126	0.388	0.346	0.125	0.378	0.307	22.5	▲**0.004**	0.946
	f^{dist}_{seq}	0.131	0.397	0.353	0.128	0.385	0.315	▲**3.6**	0.004	▲**0.959**
	f^{feat}_{seq}	0.137	0.406	0.311	0.133	0.393	0.281	32.8	0.004	0.930
	f^{item}_{seq}	0.142	0.416	0.330	0.138	0.402	0.291	37.1	0.002	0.915
	f^{rec}_{seq}	0.145	†**0.421**	0.348	0.141	†**0.405**	0.311	42.0	0.003	0.920
	f^{lcs}_{seq}	0.138	0.402	0.346	0.134	0.390	0.311	11.9	0.004	0.938
	f^{stree}_{seq}	0.146	0.416	▲**0.362**	0.142	0.402	▲**0.321**	31.3	0.003	0.921
	f^{oracle}_{seq}	▲**0.162**	▲**0.475**	0.348	▲**0.162**	▲**0.464**	0.314	41.7	0.002	0.917
TOK	Baseline	†**0.131**	†**0.389**	0.376	†**0.129**	†**0.377**	0.361	23.7	0.002	0.870
	f^{rnd}_{seq}	0.122	0.372	0.394	0.120	0.360	0.374	25.3	0.003	0.910
	f^{dist}_{seq}	0.127	0.382	▲**0.398**	0.125	0.370	▲**0.375**	▲**5.9**	▲**0.003**	▲**0.917**
	f^{feat}_{seq}	0.126	0.384	0.319	0.125	0.372	0.317	26.2	0.001	0.866
	f^{item}_{seq}	0.128	0.386	0.366	0.126	0.374	0.354	18.2	0.002	0.866
	f^{rec}_{seq}	0.130	0.388	0.365	0.128	0.376	0.354	25.2	0.002	0.870
	f^{lcs}_{seq}	0.125	0.379	0.355	0.123	0.367	0.346	9.0	0.003	0.902
	f^{stree}_{seq}	0.129	0.383	0.368	0.127	0.372	0.357	12.9	0.002	0.883
	f^{oracle}_{seq}	▲**0.144**	▲**0.428**	0.380	▲**0.144**	▲**0.417**	0.363	23.0	0.002	0.870
ROM	Baseline	0.211	0.486	0.516	0.187	0.427	0.457	5.6	0.082	0.865
	f^{rnd}_{seq}	0.178	0.440	0.507	0.167	0.397	0.447	5.9	▲**0.127**	▲**0.900**
	f^{dist}_{seq}	0.213	0.507	†**0.520**	0.201	0.456	†**0.468**	▲**1.4**	0.114	0.899
	f^{feat}_{seq}	0.189	0.464	0.411	0.174	0.414	0.364	5.3	0.077	0.890
	f^{item}_{seq}	†**0.225**	†**0.526**	0.519	†**0.207**	†**0.467**	0.466	2.1	0.081	0.872
	f^{rec}_{seq}	0.200	0.469	0.510	0.181	0.417	0.453	6.0	0.073	0.862
	f^{lcs}_{seq}	0.189	0.472	0.483	0.178	0.426	0.440	2.2	0.115	0.896
	f^{stree}_{seq}	0.202	0.481	0.514	0.186	0.431	0.458	3.5	0.103	0.880
	f^{oracle}_{seq}	▲**0.268**	▲**0.630**	▲**0.527**	▲**0.268**	▲**0.586**	▲**0.472**	4.6	0.083	0.866
PJ	Baseline	0.129	0.415	0.296	0.126	0.406	0.286	30.0	0.008	0.931
	f^{rnd}_{seq}	0.121	0.397	0.322	0.119	0.390	0.307	25.1	0.015	0.949
	f^{dist}_{seq}	0.127	0.413	▲**0.337**	0.125	0.405	▲**0.315**	▲**5.8**	▲**0.023**	▲**0.960**
	f^{feat}_{seq}	0.126	0.406	0.316	0.123	0.397	0.282	26.8	0.007	0.929
	f^{item}_{seq}	0.131	†**0.420**	0.279	†**0.130**	†**0.412**	0.270	18.8	0.003	0.922
	f^{rec}_{seq}	†**0.132**	0.416	0.301	0.130	0.407	0.290	25.8	0.003	0.931
	f^{lcs}_{seq}	0.128	0.410	0.331	0.126	0.401	0.308	10.5	0.013	0.943
	f^{stree}_{seq}	0.130	0.413	0.326	0.127	0.404	0.302	19.4	0.010	0.934
	f^{oracle}_{seq}	▲**0.141**	▲**0.456**	0.297	▲**0.141**	▲**0.449**	0.287	28.8	0.008	0.931

Table 16: Performance of the rerankers for the Tour family of recommenders. Same notation as in Table 8.

City	Reranker	Accuracy			Seq. Accuracy			Non Accuracy		
		P	NDCG	TFP	P_s	$NDCG_s$	FP_s	Dist	Gini	EPC
NYC	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
TOK	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
ROM	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
PJ	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 17: Performance for all the cities when items already seen by the user are also allowed in the recommendations. Notation as in Table 11.

			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	BF	0.160	0.443	0.340	0.151	0.424	0.306	40.1	0.004	0.922
	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	BF	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	BPR	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	Caser	0.151	0.450	0.324	0.148	0.439	0.310	22.2	0.007	0.942
	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**