Universidad Autónoma de Madrid
Escuela Politécnica Superior
Departamento de Ingeniería Informática

Recommender System Performance Evaluation and Prediction: An Information Retrieval Perspective

Dissertation written by

Alejandro Bellogín Kouki

under the supervision of

Pablo Castells Azpilicueta

and

Iván Cantador Gutiérrez

Madrid, October 2012

PhD thesis title: Recommender System Performance Evaluation and Prediction: An Information Retrieval Perspective

Author: **Alejandro Bellogín Kouki**

Affiliation: Departamento de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid, Spain

Supervisors: **Pablo Castells Azpilicueta**
Universidad Autónoma de Madrid, Spain

Iván Cantador Gutiérrez
Universidad Autónoma de Madrid, Spain

Date: October 2012

Committee: **Alberto Suárez González**
Universidad Autónoma de Madrid, Spain

Gonzalo Martínez Muñoz
Universidad Autónoma de Madrid, Spain

Arjen de Vries
Delft University of Technology, The Netherlands

Jun Wang
University College London, United Kingdom

Lourdes Araújo Serna
Universidad Nacional de Educación a Distancia, Spain

Juan Manuel Fernández Luna
Universidad de Granada, Spain

Enrique Amigó Cabrera
Universidad Nacional de Educación a Distancia, Spain

Contents

List of figures

List of tables

Abstract

Personalised recommender systems aim to help users access and retrieve relevant information or items from large collections, by automatically finding and suggesting products or services of likely interest based on observed evidence of the users' preferences. For many reasons, user preferences are difficult to guess, and therefore recommender systems have a considerable variance in their success ratio in estimating the user's tastes and interests. In such a scenario, self-predicting the chances that a recommendation is accurate before actually submitting it to a user becomes an interesting capability from many perspectives. Performance prediction has been studied in the context of search engines in the Information Retrieval field, but there is little if any prior research of this problem in the recommendation domain.

This thesis investigates the definition and formalisation of performance prediction methods for recommender systems. Specifically, we study adaptations of search performance predictors from the Information Retrieval field, and propose new predictors based on theories and models from Information Theory and Social Graph Theory. We show the instantiation of information-theoretical performance prediction methods on both rating and access log data, and the application of social-based predictors to social network structures.

Recommendation performance prediction is a relevant problem per se, because of its potential application to many uses. Thus, we primarily evaluate the quality of the proposed solutions in terms of the correlation between the predicted and the observed performance on test data. This assessment requires a clear recommender evaluation methodology against which the predictions can be contrasted. Given that the evaluation of recommender systems is an open area to a significant extent, the thesis addresses the evaluation methodology as a part of the researched problem. We analyse how the variations in the evaluation procedure may alter the apparent behaviour of performance predictors, and we propose approaches to avoid misleading observations.

In addition to the stand-alone assessment of the proposed predictors, we research the use of the predictive capability in the context of one of its common applications, namely the dynamic adjustment of recommendation methods and components. We research approaches where the combination leans towards the algorithm or the component that is predicted to perform best in each case, aiming to enhance the performance of the resulting dynamic configuration. The thesis reports positive empirical evidence confirming both a significant predictive power for the proposed methods in different experiments, and consistent improvements in the performance of dynamic recommenders employing the proposed predictors.

Resumen

Los sistemas de recomendación personalizados tienen como objetivo ayudar a los usuarios en el acceso y recuperación de información u objetos relevantes en vastas colecciones mediante la sugerencia automática de productos o servicios de potencial interés, basándose en la evidencia observada de las preferencias de los usuarios. Las preferencias de usuario son difíciles de predecir por muchos motivos y, por tanto, los sistemas de recomendación tienen una variabilidad considerable en su tasa de acierto al intentar estimar los gustos e intereses de cada usuario. En este escenario la auto-predicción de las probabilidades de que una recomendación sea acertada antes de proporcionarla al usuario se convierte en una capacidad interesante desde múltiples perspectivas. La predicción de eficacia ha sido estudiada en el contexto de los motores de búsqueda en el campo de la Recuperación de Información, pero apenas se ha investigado en el dominio de la recomendación.

Esta tesis investiga la definición y formalización de métodos de predicción de eficacia para sistemas de recomendación. Concretamente, se estudian adaptaciones de predictores de eficacia de búsqueda en el campo de la Recuperación de Información, y se proponen nuevos predictores basados en modelos y técnicas de la Teoría de la Información y la Teoría de Grafos Sociales. Se propone la instanciación de métodos de teoría de información para predicción de eficacia tanto en datos de valoraciones de usuario explícitas como en registros de accesos, así como la aplicación de predictores sociales sobre estructuras de red social.

La predicción de eficacia de recomendación es un problema relevante per se dados sus múltiples usos y aplicaciones potenciales. Por ello, en primer lugar se evalúa la calidad de las soluciones propuestas en términos de la correlación entre la eficacia estimada y la observada en los datos de test. Esta valoración requiere una metodología clara de evaluación de sistemas de recomendación con la que las predicciones puedan ser contrastadas. Dado que la evaluación de los sistemas de recomendación es aún un área de investigación en buena medida abierta, la tesis aborda la metodología de evaluación como parte del problema a investigar. Se analizan entonces cómo las variaciones en el procedimiento de evaluación pueden alterar la percepción del comportamiento de los predictores de eficacia, y se proponen aproximaciones para evitar observaciones engañosas.

Además de las valoraciones independientes de los predictores propuestos, investigamos el uso de su capacidad predictiva en el contexto de una de sus aplicaciones comunes, a saber, el ajuste dinámico de métodos híbridos para combinar algoritmos y componentes de recomendación. Se investigan aproximaciones donde la combinación se inclina hacia el algoritmo o la componente que se predice va a tener mejor eficacia en cada caso, a fin de mejorar la eficacia de la configuración dinámica resultante. La tesis presenta resultados empíricos positivos que confirman tanto un poder predictivo significativo para los métodos propuestos, como consistentes mejoras en la eficacia de recomendaciones dinámicas que utilizan los predictores propuestos.

Acknowledgements

This manuscript is the result of several years of work in the college, master, and finally PhD studies. In all this time I have received the support of my family, friends, and colleagues, to whom I am very grateful. These lines are a sign of my gratitude to all of them.

First, I would like to thank my supervisors Pablo Castells and Iván Cantador for their constant encouragement at any aspect I had to face these years, from the stressful talks to the (sometimes confusing) reviews received, and including those about the overwhelming first classes as a teacher and the tiresome paperwork. Thanks to Pablo for giving me the opportunity to pursue this PhD with him, for his effort and continous work during these years, and for allowing me to participate in conferences and projects where I have learnt so much about deadlines and priorities, and how to deal with them according to a given purpose. Special thanks also to Iván whose continuous support helped me to improve my writing and design skills, although there is still some room for improvement in this respect, especially in the latter. Besides, I started my research training with him, which let me grow professionally, and represented my first satisfactions in the form of papers.

Fernando Díez deserves a special mention, since without his counsel and early interest I probably would not belong to my research group, and I may not have done any research at all. I am also deeply grateful and in debt to all of the current and past members of the IRG/NETS group, especially to David Vallet, Miriam Fernández, Sergio López, Pedro Campos, Saúl Vargas, Nacho Fernández, Víctor Villasante, José Conde, and María Medina for all the great times spent together and the really interesting discussions about research, programming, teaching, and life in general. I would also like to thank the rest of occupants of the B-408 and previously those of B-402 (Alexandra Dumitrescu, Enrique Chavarriaga, César Álvarez, Yolanda Torres, David Alfaya, Víctor López, Alejandro García, Manuel Torres, Laura Hernández, Álvaro Valera, Luis Martín, Chema Martínez, and Sara Schvartzman), and the people from the VPULab/GTI, especially José María Martínez, Jesús Bescós, Víctor Valdés, Javier Molina, Fernando López, Marcos Escudero, and Álvaro García with whom I shared projects and meetings.

After my two internships at the University College London, I am very grateful to Jun Wang for his kind support, and also to Tamas Jambor and Jagadeesh Gorla with whom I had the pleasure of collaborating and discussing whenever I needed. Outside of my group, I also met many interesting people with whom I had the opportunity to

discuss about research, learn many things, and even collaborate together; thus, I want to extend my gratitude to all of them, in particular, my special thanks to Javier Parapar, Miguel Martínez, Denis Parra, Xavier Amatriain, Joaquín Pérez, Neal Lathia, Alan Said, Zeno Gantner, Yue Shi, Marcel Blattner, Anmol Bashin, Gunnar Schröder, Sandra García, Owen Phelan, José Ramón Pérez, Ronald Teijeira, Álvaro Barbero, Miguel Ángel Martínez, Alfonso Romero, Eduardo Graells, Marko Tkalčič, and Arkaitz Zubiaga. I am also in debt with the people behind the Apache Mahout open source project since I have used it almost from the beginning of my training process and it offered me the implementations of my first recommender systems.

Gracias a la gente que conocí en el máster (allá por el 2008-2009), sobre todo con los que colaboré de alguna u otra manera, y con los que compartí charlas y discusiones sobre las asignaturas tanto dentro como fuera de la facultad. Muchas gracias a Richard Rojas, Joaquín Fernández, Rafael Martínez, José Miguel Rojas, Clemente Borges y Sergio García. Y si agradezco a los que me acompañaron en el máster, no voy a ser menos con aquellos con los que hice la carrera, la 'doble', los que sobrevivimos a la primera promoción de una carrera donde aprendí mucho, tanto académica, como personalmente. Muchas personas merecen este reconocimiento, pero se lo dedico en especial a Juanda, mi compañero de prácticas durante casi toda la carrera y una de las personas con las que más he aprendido: gracias. También gracias a Roberto, Maite, Jorge, Willy, Irene, Edu, Elena, Jesús, Fran y María por conseguir que esos cinco años pasaran mucho más rápido.

I should also mention the grants that supported my research: the fundings for travel, conference, and publication expenses covered by different European and Spanish research projects (references FP6-027685, TSI-2006-26928-E, CENIT-2007-1012, TIN-2008-06566-C04-02, S2009TIC-1542, CCG10-UAM/TIC-5877, and TIN2011-28538-C02-01), the pre-doctoral grants offered by Universidad Autónoma de Madrid ('Ayudas para inicio de estudios en programas de posgrado 2007' and 'Formación de personal investigador FPI 2009', which additionally covered the expenses of my second internship at UCL), and the teaching assistant position ('Profesor Ayudante') offered in 2010 by the same university, which is currently funding this PhD.

También gracias a mis alumnos, con el de este año ya van siete los grupos a los que he dado clase, y aún no termino de acostumbrarme a la sensación de ver cómo alguien confía plenamente en lo que dices para después observar cómo ha aprendido a hacer cosas que un momento antes no imaginaba que podría hacer. Esa sensación la tendré siempre conmigo. Además, agradezco a los profesores de la Escuela Politécnica Superior y del Departamento de Matemáticas todo lo aprendido entonces como alumno, y ahora como profesor, donde desde el otro lado puedo apreciar aún más lo difícil que es ejercer como docente, y cómo algunos os empeñábais en que pareciera muy sencillo: muchísimas gracias. También merecen un agradecimiento especial el personal de Secretaría y de Administración de la Escuela, a los que he traí-

do de cabeza (sobre todo últimamente) con todas las solicitudes, justificantes y demás papeleo necesario en estos últimos 10 años, gracias sobre todo a Marisa Moreno, María José García, Juana Calle y Amelia Martín. Por supuesto, este agradecimiento también se extiende al personal de biblioteca, conserjería, limpieza y cafetería, que hacen que el tiempo que se pasa en la facultad (a veces más del deseable) sea más sencillo, agradable y ameno.

A mis amigos más cercanos, algunos de ellos del colegio, como Emiliano, Raúl, María, Julito, Luis, Gonzalo, Juan, Esther y Claudio; y otros de después, como Roberto, Noha, Alba y Gema; a todos ellos también les agradezco el haber estado ahí todos estos años y no desesperar cuando no podía salir porque estaba siempre ocupado, y por escuchar y mostrar interés cuando se atrevían a preguntar qué hacía en la universidad. Un abrazo muy grande y un gracias transoceánico a mi buen amigo Hugo, que me enseñó que siempre hay que perseguir lo que uno desea, y con el que siempre podré contar, como cuando hablábamos a deshoras durante mi estancia en Londres o trasnochando para acabar algún artículo.

Por último, y por supuesto, no menos importante, a mi familia. Gracias a mis hermanos por obligarme a jugar a la consola para despejarme, y a hablar de fútbol, baloncesto o música aún cuando llevamos varios días sin vernos por nuestros horarios desacoplados, eso hizo más fácil pasar fuera el tiempo necesario para haber podido terminar esta tesis. Gracias a mi madre, de la que he aprendido a trabajar duro para seguir adelante y que el esfuerzo es lo único que cuenta; ella es y siempre ha sido mi inspiración. Y gracias a Susana y a su familia, por acogerme desde el principio como uno más, y por interesarse siempre por mi tesis. Sin duda, Susana ha sido una de las tres personas que más de cerca ha sufrido todo este proceso, sólo me queda agradecerle el haber estado conmigo todo este tiempo y el haber sido capaz de convencerme de que después de cada rechazo habría algo positivo y que, después de todo, yo sería capaz de superarlo. Gracias, no lo habría conseguido sin ti.

Alejandro Bellogín
October 2012

A mi madre, mis hermanos y Susana

Part I

Introduction and context

Antes de nada has de saber
que no soy recomendable.
Ismael Serrano

Chapter 1

Introduction

In this chapter we present a general overview of the thesis. In Sections 1.1 and 1.2 we provide the motivations and research goals of our work. In Section 1.3 we summarise the main contributions of the thesis, and in Section 1.4 we list the publications resulting from our research. Finally, in Section 1.5 we describe the structure of this document.

1.1 Motivation

Information Retrieval (IR) technologies have gained outstanding prevalence in the last two decades with the explosion of massive online information repositories, and much in particular the World Wide Web. IR systems are researched and designed in ways that seek to maximise the degree of satisfaction of certain objective conditions, typically – though not necessarily only – user satisfaction. IR research and development have revolved around the definition of models and algorithms that best achieve this goal, methodologies and metrics that let assess how well the goal is achieved by different systems, and sound theories providing a solid ground and orientation in the development of IR algorithms and their consistent evaluation. Among many new trends stemming from this main stream of research and developments, a new research goal started to be considered by the early 2000's: is it possible to predict how good a result returned by an IR system is going to be, before presenting it to the user, or even, before running the IR system at all (Cronen-Townsend et al., 2002)? This question has given rise to a fertile strand of research on so-called **performance prediction** in IR.

Performance prediction has many potential uses in IR. From the user's perspective it may provide valuable feedback that can be used to direct a search, from the system's perspective it may help to distinguish poorly performing queries, and from the system administrator's perspective it may let identify queries related to a specific subject that are difficult for the search engine. Performance prediction approaches are based on the analysis and characterisation of the evidence used by an IR system to assess the relevance (utility, value, etc.) of retrieval objects (documents, goods, etc.) at execution time (Cronen-Townsend et al., 2002). The most classic and basic retrieval scenario involves a user query and a collection of documents as the basic input to form a ranked list of search results, but other additional elements can be taken into account to select and rank results (Baeza-Yates and Ribeiro-Neto, 2011). Any information the retrieval system takes as input can be taken as input for the performance prediction as well, and often the prediction methods use additional information beyond that. The user context (current tasks, query logs, preferences, etc.), global properties of the document collection, comparisons with respect to other reference elements such as historic data, and the output from other systems, among others, are some examples of the different sources of information that a predictor may draw evidence from.

Predicting the performance of a subsystem, module, function, or input by contrasting the performance estimation for a query for each component, enables an array of dynamic optimisation strategies that select at runtime the option which is predicted to work best or, when larger systems or hybrid approaches are used, allows for adjusting on the fly the participation of each module. The IR field is pervaded with

cases where information relevance, retrieval systems, models, and criteria are based on a fusion or combination of sub-models. Personalised retrieval systems (including techniques such as personalised search, recommender systems, collaborative filtering, and retrieval in context) are clear examples where performance prediction can be applied since such systems combine several sources of evidence for relevance assessment, such as explicit queries, search history, explicit user ratings, social information, user feedback, and context models.

Performance prediction finds additional motivation in personalised recommendation, inasmuch these applications may decide to produce recommendations or hold them back, delivering only the sufficiently reliable ones. Furhermore, current Recommender Systems (RS) are characterised by an increasing diversification of the types and sources of data, content, evidence and methods, available to make decisions and build their output. In such context, predicting the performance of a specific recommendation approach or component becomes an appealing problem, as it lets properly combine the available alternatives, and make the most of them by dynamically adapting the recommendation strategy to the situation at hand. The question gains increasing relevance today, with the proliferation of hybrid recommendation techniques to improve the accuracy of the methods – the Netflix prize was a paradigmatic example of the use of this, where all the top ranked participants used combinations of large sets of recommendation methods. This calls for the research of hybrid approaches with a level of dynamic self-adjustment mechanisms, in order to optimise the resulting effectiveness of the recommendation systems, by opportunistically taking advantage of high-quality data when available, but avoiding sticking to fixed strategies when they can be predicted to yield poor results under certain conditions.

Performance prediction in IR is typically assessed in terms of the correlation between a predictor's scores and a system's performance values on a per-query basis. This requires reliable performance evaluation metrics and methodologies, which have been thoroughly analysed, and are currently well established in the IR field, mostly oriented to ad-hoc search. In contrast, evaluation in the RS field is more open, and the variability in evaluation approaches and experimental configurations is significant. How to measure the performance of a recommender system is a key issue in our research since the system quality measurements may be influenced by statistical properties of the measurement approach and/or the experimental design. Throughout this thesis we shall focus on the accuracy of the system, where we have to avoid that if a metric – i.e., precision – is biased towards some form of noise along with the recommender's quality, then a predictor capturing only that noise would appear as an (equivocal) effective performance predictor. Hence, statistical biases (noises) of the evaluation methodologies should be well understood in order to enable a meaningful assessment of performance predictors.

Drawing from the state of the art on performance prediction in IR as a starting point, the present work restates the problem in the field of Recommender Systems where it has barely been addressed so far. We research meaningful definitions of performance in the context of RS, and the elements to which it can sensibly apply, investigating the statistical biases that may arise when adapting the IR evaluation framework into RS. In doing so, we take as a driving direction the application of performance prediction to achieve improvements in two specific combination problems in the RS field, namely, the dynamic combination of recommendation methods in hybrid recommendation systems, and the dynamic aggregation of neighbours' signals in user-based collaborative filtering.

1.2 Research goals

The main objective of the research presented here is to find predictive methods for the performance of specific components in recommender systems, and to improve the performance of combined recommendation methods, based on the dynamic, automatic analysis and prediction of the expected performance of the constituents of the composite methods, whereupon the relative participation of each constituent is adjusted, in accordance to its predicted effectiveness. To address these problems, this work has the following specific research objectives:

RG1: Analysis and formalisation of how retrieval performance is defined and evaluated in recommender systems. We need to develop an in-depth study on how recommender systems can be reliably evaluated in terms of numeric metric values, since we aim to predict their performance. Moreover, we have to investigate whether there is any bias on the way the systems are evaluated – either by the evaluation methodologies or metrics, since any bias in the evaluation process would lead to inconclusive or misleading results about the predictive power of the performance prediction methods proposed. If these biases do exist, we aim to precisely understand them and develop methodologies to isolate them; then, we shall check the effectiveness of the predictors against well-known baselines and whether it changes when unbiased methodologies are used.

RG2: Adaptation and definition of performance prediction techniques for recommender systems. We aim to study the potential of performance prediction in specific problems and settings in the area of Recommender Systems. We shall investigate the definition of a formal framework where performance predictors can be integrated. As a starting point, we aim to explore the adaptation of specific effective predictors from Information Retrieval such as query clarity (Cronen-Townsend et al., 2002) to recommender systems. Complementarily to the adaptation of known techniques, we aim to research the definition of new predictors based on models from Information Theory and Social Graphs, besides other heuristic, domain-specific ap-

proaches. Once we have defined some recommendation performance predictors, we shall assess the effectiveness of such predictors in terms of their correlation to performance metrics to estimate the predictive power of the performance predictors.

RG3: Application of performance predictors to hybrid and compound recommender systems. We aim to identify and integrate the proposed predictors into combined recommendation methods, in order to achieve an actual improvement in the performance of the combined methods. With this goal in mind, we shall consider problems where an aggregation of recommendation methods is needed, and shall analyse how to apply the performance predictors mentioned above in such problems. Besides, a methodological study for the experimental approach, setup, and metrics should be performed in such a way that appropriate baseline methods and experimental designs are used. Finally, we shall assess the improvements and benefits of the combined methods when the performance predictors are applied.

1.3 Contributions

This thesis is devoted to the problem of estimating the performance of recommender systems for particular users and items. The main contributions of this thesis are related to the evaluation of the performance of a recommender system, and the prediction of such performance, where we have addressed several issues regarding both topics and we have proposed novel models and methods, which have been applied into two applications as we shall see next.

As a first step, this thesis analyses the Cranfield paradigm of Information Retrieval evaluation since recommender systems are usually considered as a particular problem of information filtering, and, thus, of information retrieval at large (Belkin and Croft, 1992). In Chapter 4 **we discuss the differences involved in the experimental design alternatives from the common assumptions made in the Cranfield paradigm**, which result in **substantial statistical biases arising in Recommender Systems**, and **we propose different methods to neutralise these biases**. Additionally, the following related contributions have been addressed:

- We propose a precise and systematic characterisation of design alternatives in the adaptation of the Cranfield paradigm to recommendation tasks. We identify assumptions and conditions underlying the Cranfield paradigm that are not granted in usual recommendation experiments.
- We detect and characterise resulting statistical biases, namely test sparsity and item popularity, which do not arise in common test collections from IR, but do interfere in recommendation experiments.

- We propose two novel experimental designs in order to neutralise these biases. We observe that a percentile-based evaluation considerably reduces the margin for the popularity bias, whereas a uniform-test approach removes any statistical advantage provided by having more positive test ratings. Furthermore, we find that both approaches discriminate well between pure popularity-based recommendation and an efficient personalised recommendation algorithm.

Additionally, in this thesis **we show how query performance prediction techniques developed in Information Retrieval can be adapted to Recommender Systems, and result in effective predictors in this domain**. We present these performance predictors in Chapter 6, where we propose different adaptations of the query clarity predictor based on different interpretations of the underlying language models along with models from Information Theory and Social Graphs. Furthermore, in the same chapter **we assess the effectiveness of such predictors by measuring the correlation with respect to performance metrics**, where we also test the methods proposed in Chapter 4 to neutralise biases on evaluation. Specific contributions regarding performance prediction for recommendation are summarised as follows:

- We define and elaborate several predictive models in the Recommender Systems domain according to different formulations and assumptions, and based on three types of preference data: rating-based, log-based, and social-based.
- Formulations for rating preferences are based on adaptations of query clarity from IR and concepts from Information Theory such as entropy. In this adaptation we propose different probability estimations, where Bayesian derivations and non-parametric estimations are developed.
- We also exploit temporal features when defining log-based predictors. Specifically, we use a time-aware version of the Kullback-Leibler divergence, along with other time series concepts such as a user's autocorrelation.
- We use graph-based metrics from Graph Theory to define predictors leveraging social network structures, and correlations between topological properties of users and the success of recommendations delivered to them.
- We find strong correlations between the outputs of the predictors and the performance metrics, thus finding empirical evidence of the predictive power of the proposed approaches. Furthermore, when unbiased evaluation methodologies are used, the predictors still obtain good correlation values, evidencing that our proposed predictors are not just capturing and benefitting from the analysed biases, especially when we compare them against other trivial predictors.

Finally, Chapters 7 and 8 present two applications of performance predictors on Recommender Systems. In Chapter 7 **we propose several linearly weighted hy-**

brids where the weights are dynamically adjusted based on the predictors' output. We observe that the correlations obtained in Chapter 6 help decide which are the best combinations to experiment with. More importantly, **the correlation between the predictor and the recommender tends to anticipate well when a hybrid will outperform its baseline**. Besides, Chapter 8 **presents a unified framework where the performance predictors are used to select and weight nearest neighbours in a standard user-based collaborative filtering algorithm**. The standard methodology from performance prediction is adapted and translated into this problem, where novel neighbour performance metrics are defined and the predictive power of the predictors is assessed.

The contributions related to the application part of the thesis are, in summary:

- We propose a dynamic hybrid framework to automatically decide when and how dynamic hybridisation should be done, depending on different conditions, namely the correlations between the recommenders and the predictors, and the relative performance level of the combined recommenders.
- In several experiments with the aforementioned performance predictors, our results indicate that a strong correlation with performance tends to correspond with enhancements in dynamic hybrid recommendation when the predictors are used for the adjustment of the combination weights.
- We propose a theoretical framework for neighbour selection and weighting in user-based recommender systems. This framework is based on performance prediction by casting the neighbourhood-based rating prediction task as a case of dynamic output aggregation.
- We compare several state-of-the-art rating-based trust metrics and other proposed neighbour scoring techniques, interpreted as neighbour performance predictors. We also propose several neighbour performance metrics that capture different notions of neighbour quality.

1.4 Publications related to the thesis

In the following international journal and conference papers we presented descriptions, results and conclusions related to this thesis:

Performance prediction and evaluation

1. Bellogín, A., Cantador, I., Díez, F., Castells, P., and Chavarriaga, E. (2012). An empirical comparison of social, collaborative filtering, and hybrid recommenders. *ACM Transactions on Intelligent Systems and Technology*, to appear.

2. Bellogín, A., Castells, P., and Cantador, I. (2011). Predicting the Performance of Recommender Systems: An Information Theoretic Approach. In Amati, G. and Crestani, F., editors, *ICTIR*, volume 6931 of *Lecture Notes in Computer Science*, pages 27–39, Berlin, Heidelberg. Springer Berlin / Heidelberg.

3. Bellogín, A., Castells, P., and Cantador, I. (2011). Self-adjusting hybrid recommenders based on social network analysis. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information*, SIGIR '11, pages 1147–1148, New York, NY, USA. ACM.

4. Bellogín, A., Castells, P., and Cantador, I. (2011). Precision-oriented evaluation of recommender systems: an algorithmic comparison. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 333–336, New York, NY, USA. ACM.

5. Bellogín, A. and Castells, P. (2010). A Performance Prediction Approach to Enhance Collaborative Filtering Performance. In Gurrin, C., He, Y., Kazai, G., Kruschwitz, U., Little, S., Roelleke, T., Rüger, S., and Rijsbergen, editors, *Advances in Information Retrieval*, volume 5993 of *Lecture Notes in Computer Science*, pages 382–393, Berlin, Heidelberg. Springer Berlin / Heidelberg.

6. Bellogín, A. and Castells, P. (2009). Predicting neighbor goodness in collaborative filtering. In And, T. A., Yager and, R. R., And, H. B., And, H. C., and Larsen, H. L., editors, *FQAS*, volume 5822 of *Lecture Notes in Computer Science*, pages 605–616, Berlin, Heidelberg. Springer Berlin / Heidelberg.

Content-based recommendation

7. Cantador, I., Bellogín, A., and Vallet, D. (2010). Content-based recommendation in social tagging systems. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 237–240, New York, NY, USA. ACM.
8. Cantador, I., Bellogín, A., and Castells, P. (2008). News@hand: A Semantic Web Approach to Recommending News. In Nejdl, W., Kay, J., Pu, P., and Herder, E., editors, *Adaptive Hypermedia and Adaptive Web-Based Systems*, volume 5149 of *Lecture Notes in Computer Science*, chapter 34, pages 279–283. Springer Berlin / Heidelberg, Berlin, Heidelberg.
9. Cantador, I., Bellogín, A., and Castells, P. (2009). Ontology-Based Personalised and Context-Aware Recommendations of News Items. In *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT '08. IEEE/WIC/ACM International Conference on*, volume 1, pages 562–565.
10. Cantador, I., Bellogín, A., Fernández-Tobías, I., and López-Hernández, S. (2011a). Semantic Contextualisation of Social Tag-Based Profiles and Item Recommendations. In Huemer, C., Setzer, T., Aalst, W., Mylopoulos, J.,

Sadeh, N. M., Shaw, M. J., Szyperski, C., Aalst, W., Mylopoulos, J., Sadeh, N. M., Shaw, M. J., and Szyperski, C., editors, *Electronic Commerce and Web Technologies*, volume 85 of *Lecture Notes in Business Information Processing*, chapter 9, pages 101–113. Springer Berlin Heidelberg, Berlin, Heidelberg.

11. Fernández-Tobías, I., Cantador, I., and Bellogín, A. (2011). cTag: Semantic contextualisation of social tags. In *Proceedings of the Workshop on Semantic Adaptive Social Web (SASWeb 2011). CEUR Workshop Proceedings, vol. 730*, pages 45–54. RWTH, Aachen (2011).

Collaborative filtering recommendation

12. Bellogín, A., Wang, J., and Castells, P. Bridging Memory-Based Collaborative Filtering and Text Retrieval. Information Retrieval Journal, to appear.

13. Bellogín, A., Cantador, I., and Castells, P. A Comparative Study of Heterogeneous Item Recommendations in Social Systems. *Information Sciences*, to appear.

14. Bellogín, A. and Parapar, J. (2012). Using Graph Partitioning Techniques for Neighbour Selection in User-Based Collaborative Filtering. In *Proceedings of the sixth ACM conference on Recommender systems*, RecSys '12, pages 213–216, New York, NY, USA. ACM. (best short paper award)

15. Bellogín, A., Wang, J., and Castells, P. (2011). Structured collaborative filtering. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 2257–2260, New York, NY, USA. ACM.

16. Bellogín, A., Wang, J., and Castells, P. (2011). Text Retrieval Methods for Item Ranking in Collaborative Filtering. In Clough, P., Foley, C., Gurrin, C., Jones, G., Kraaij, W., Lee, H., and Mudoch, V., editors, *Advances in Information Retrieval*, volume 6611 of *Lecture Notes in Computer Science*, chapter 30, pages 301–306. Springer Berlin / Heidelberg, Berlin, Heidelberg.

17. Bellogín, A., Cantador, I., and Castells, P. (2010). A study of heterogeneity in recommendations for a social music service. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '10, pages 1–8, New York, NY, USA. ACM.

Social filtering recommendation

18. Díez, F., Chavarriaga, J. E., Campos, P. G., and Bellogín, A. (2010). Movie recommendations based in explicit and implicit features extracted from the Filmtipset dataset. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, CAMRa '10, pages 45–52, New York, NY, USA. ACM.

Time-aware recommendation

19. Campos, P. G., Bellogín, A., Díez, F., and Cantador, I. (2012). Time Feature Selection for Identifying Active Household Members. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, New York, NY, USA. ACM (to appear).
20. Campos, P. G., Díez, F., and Bellogín, A. (2011). Temporal rating habits: a valuable tool for rating discrimination. In *Proceedings of the 2nd Challenge on Context-Aware Movie Recommendation*, CAMRa '11, pages 29–35, New York, NY, USA. ACM.
21. Campos, P. G., Bellogín, A., Díez, F., and Chavarriaga, J. E. (2010). Simple time-biased KNN-based recommendations. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, CAMRa '10, pages 20–23, New York, NY, USA. ACM.

Hybrid recommender systems

22. Cantador, I., Castells, P., and Bellogín, A. (2011). An enhanced semantic layer for hybrid recommender systems. *International Journal on Semantic Web and Information Systems*, 7(1):44–78.
23. Cantador, I., Bellogín, A., and Castells, P. (2008). A multilayer ontology-based hybrid recommendation model. *AI Commun.*, 21(2-3):203–210.
24. Cantador, I., Castells, P., and Bellogín, A. (2007). Modelling Ontology-based Multilayered Communities of Interest for Hybrid Recommendations. In *Workshop on Adaptation and Personalisation in Social Systems: Groups, Teams, Communities, at the 11th International Conference on User Modeling*.

Recommender evaluation

25. Bellogín, A., Cantador, I., Castells, P., and Ortigosa, A. (2011). Discerning Relevant Model Features in a Content-based Collaborative Recommender System. In Fürnkranz, J. and Hüllermeier, E., editors, *Preference Learning*, chapter 20, pages 429–455. Springer Berlin Heidelberg, Berlin, Heidelberg.
26. Bellogín, A., Cantador, I., Castells, P., and Ortigosa, A. (2008). Discovering Relevant Preferences in a Personalised Recommender System using Machine Learning Techniques. In *Preference Learning Workshop (PL 2008), at the 8th European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2008)*, pages 82–96.

These publications are related to the contents of this thesis as follows. In [4] we analyse different evaluation methodologies available in the recommendation literature (Chapters 3 and 4). In [2], [5], and [6] we define the formulations for the concept of user clarity based on ratings (Chapter 6), whereas in [1] and [3] we define the social-

based predictors (again, Chapter 6). Besides, in [1] and [3] we also investigate the use of performance predictors for dynamic hybrid recommendation (Chapter 7). Moreover, in [5] and [6] we address the problem of neighbour weighting based on neighbour performance predictors (Chapter 8).

Additionally, during the course of the thesis, the research presented here has motivated a number of publications that address broader topics in the field, such as content-based recommendation [7-11], collaborative filtering [12-17], social filtering techniques [18], time-aware recommendation [19-21], hybrid recommender systems [22-24], and recommendation evaluation [25, 26]. These publications have resulted in the use and construction of datasets, the development of algorithms and the research and use of some evaluation methodologies and metrics that appear in this thesis.

Additional publications

Preliminary work towards the approaches presented in this thesis was published in my Master's Thesis entitled "Performance prediction in recommender Systems: Application to the dynamic optimisation of aggregative methods" (Bellogín, 2009); specifically, the concept of performance prediction for recommendation is proposed in such work. Apart from that, the motivation, potential impact, and initial main results of our research were published as contributions in two international doctoral symposiums:

- Bellogín, A. (2011). Predicting performance in recommender systems. Doctoral Symposium. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 371–374, New York, NY, USA. ACM.
- Bellogín, A. (2011). Performance Prediction in Recommender Systems. Doctoral Symposium. In Konstan, J., Conejo, R., Marzo, J., and Oliver, N., editors, *User Modeling, Adaption and Personalization*, volume 6787 of *Lecture Notes in Computer Science*, pages 401–404, Berlin, Heidelberg. Springer Berlin / Heidelberg.

Furthermore, the following submissions are under revision, some of them closely related to the topics of the thesis:

- Bellogín, A., Castells, P., and Cantador, I. Statistical Biases in IR Metrics for Recommender Systems: A Methodological Framework for the Adaptation of the Cranfield Paradigm. Under review.
- Bellogín, A., Castells, P., and Cantador, I. Neighbour Selection and Weighting in User-Based Recommender Systems: A Performance Prediction Approach. Under review.
- Parapar, J., Bellogín, A., Castells, P., and Barreiro, Á. Relevance-Based Language Modelling for Recommender Systems. Under review.

1.5 Structure of the thesis

The thesis is divided into six parts. The first part introduces and motivates the problem addressed, along with a survey of the Recommender Systems field, where this thesis is framed. The second part describes the different evaluation techniques used in the recommender systems literature and provides an analysis of the design alternatives and statistical biases that may arise. The third part gives background knowledge and a literature survey on performance prediction, proposes translations of this concept into the recommender system space, and evaluates the predictive power of these approaches. The fourth part provides two applications of the proposed recommender performance predictors. The fifth part concludes and summarises the main contributions of this thesis. Additional information and details are provided in the last part.

In more detail, the contents of this thesis are distributed as follows:

Part I. Introduction

- **Chapter 1** presents the motivation, research goals, contributions and publications related to the thesis.
- **Chapter 2** provides an overview of the state of the art in recommender systems, considering a classification of the main types of recommendation approaches. We also describe the weaknesses of the different recommendation techniques and present a broader class of hybrid recommenders that aim to overcome these limitations.

Part II. Evaluating Performance in Recommender Systems

- **Chapter 3** describes the main evaluation metrics and methodologies used in the recommender systems field. The public datasets commonly used in the field are also described.
- **Chapter 4** provides an analysis and formalisation of the different evaluation methodologies reported in the literature. First, we present a systematic characterisation of the experimental design alternatives. Next, we identify and analyse specific statistical biases arising when some methodologies are applied to recommendation, and propose two alternative experimental designs that effectively neutralise such biases to a large extent.

Part III. Predicting Performance in Recommender Systems

- **Chapter 5** presents the problem of performance prediction in Information Retrieval, surveys the main research works in that area, both in the definition of (query) performance predictors and also in the predictor evaluation in order to infer their predictive power.

- **Chapter 6** states the problem of performance prediction in recommender systems. We define several performance predictors based on three recommendation input spaces where we qualitative analyse the predictive power of the predictors.

Part IV. Applications

- **Chapter 7** proposes a framework where recommender performance predictors are used to build dynamic hybrid recommender systems. We evaluate these recommenders in the three input spaces previously considered for the definition of performance predictors and using different experimental design alternatives where some statistical biases are neutralised.
- **Chapter 8** restates the user-based recommendation problem, providing a generalisation as a performance prediction problem. We investigate how to adopt this generalisation to define a unified framework where we conduct an objective analysis of the effectiveness (predictive power) of neighbour scoring functions.

Part V. Conclusions

- **Chapter 9** concludes with a summary of the main contributions of this thesis, and a discussion about future research lines.

Part VI. Appendices

- **Appendix A** provides details about the methods proposed in this thesis: configuration of the recommendation algorithms and parameters of the experimental designs used in the evaluation. Detailed statistics about the datasets used in the experiments are provided, complementary to those given in previous chapters.
- **Appendix B** contains the translation into Spanish of Chapter 1.
- **Appendix C** contains the translation into Spanish of Chapter 9.

Chapter 2

Recommender systems

The aim of recommender systems is to assist users in finding their way through huge databases and catalogues, by filtering and suggesting relevant items taking into account or inferring the users' preferences (i.e., tastes, interests, or priorities).

Three types of recommender systems are commonly recognised according to how recommendations are made, namely content-based filtering (CBF), collaborative filtering (CF), and social filtering (SF) systems. A CBF system suggests a user items similar to those she preferred or liked in the past, a CF system suggests a user items that people with similar preferences liked in the past, and a SF system suggests items according to the preferences of the user's social contacts in a social network. Each of these types of recommendations has its own strengths and weaknesses. In order to address and compensate particular shortcomings, combinations of different recommendation approaches are usually developed, forming the so called hybrid filtering (HF) systems.

In this chapter we provide an overview of terminology, techniques, and limitations related to the above types of recommender systems. In Section 2.1 we formalise the problem of recommendation, and introduce the different types of recommendation approaches. Next, in Section 2.2 we describe content-based recommendation approaches, rating- and log-based recommendation approaches – as special cases of collaborative filtering –, and social-based recommendation approaches. In Section 2.3 we then explain generic hybrid filtering approaches. Finally, in Section 2.4 we present particular limitations of each type of recommender systems.

2.1 Formulation of the recommendation problem

Collaborative filtering can be considered as the first proposed recommendation approach. The term was coined in the mid 90's by Goldberg and colleagues when developing an automatic filtering system for electronic mail (Goldberg et al., 1992), although sometimes the stereotypes defined in (Rich, 1979) have been considered as an earlier reference. Collaborative filtering has been classed as part of the Information Retrieval area by several authors (Belkin and Croft, 1992; Foltz and Dumais, 1992), who have considered recommender systems as a particular case of information filtering. However, only a few recent attempts have been made at bringing recommender systems and information retrieval models together, by establishing equivalences between them (Wang et al., 2008b; Wang et al., 2008a; Bellogín et al., 2011b). Instead, recommender systems have been traditionally investigated from a different perspective, such as preference prediction and Machine Learning (Breese et al., 1998), upon which the main prediction models and evaluation metrics have been developed.

In this context distinct formulations and notations have been proposed. The overview by Adomavicius and Tuzhilin (2005) are among the most cited. In that work the recommendation problem is defined as follows. Let $\mathcal{U}$ be a set of users, and let $\mathcal{I}$ be a set of items. Let $g: \mathcal{U} \times \mathcal{I} \rightarrow \mathrm{R}$, where R is a totally ordered set, be a utility function such that $g(u,i)$ measures the gain of usefulness of item i for user u. Then, for each user $u \in \mathcal{U}$, we aim to choose items $i^{\max,u} \in \mathcal{I}$, unknown to the user, which maximise the utility function g, that is:

$$\forall u \in \mathcal{U}\ ,\ i^{max,u} = \arg\max_{i \in \mathcal{I}} g(u,i)$$

Depending on the exploited source of user preference information, and the way in which the utility function g is estimated for different users, the following two main types of recommender systems are commonly distinguished: 1) content-based recommender systems, in which a user is suggested items similar to those she liked or preferred in the past, and 2) collaborative filtering systems, in which a user is suggested items that people with similar preferences liked in the past. We extend this classification by also considering social recommender systems, i.e., systems in which a user is suggested items that friends – e.g. in an online social network – liked in the past. These systems are related but significantly different from collaborative filtering systems. Moreover, we distinguish two types of collaborative filtering systems, based on the form of their input: systems that exploit explicit user ratings (rating-based systems), and systems that exploit implicit user preference information (log-based systems). The rating assigned to an item by a particular user is typically interpreted as the true utility of that item for the user. There are systems, however, where no explicit ratings are available, but where user interests can be inferred from implicit

feedback information. In order to provide item recommendations in such systems, two plausible approaches do exist: 1) directly exploiting implicit preference data (Wang et al., 2008b; Deshpande and Karypis, 2004; Das et al., 2007; Hu et al., 2008; Linden et al., 2003), and 2) transforming implicit preference data into explicit ratings to be exploited by standard CF strategies (Celma, 2010; Jawaheer et al., 2010; Adams, 2007).

Other types of recommender systems have been considered in the literature, although they will not be described in detail herein; these are knowledge-based, utility-based, and demographic-based recommender systems.They use, respectively, semantic descriptions of the user preferences and item characteristics, an utility function over the items that describes the users' preferences, and demographic information about the users. For further descriptions and examples of these techniques, see (Burke, 2002) and (Ricci et al., 2011).

For any of the above mentioned types of recommender systems, models can be combined to improve their separate performance, or other characteristic of interest, such as the capability of providing more diverse and novel recommendations, and offering better explanations of recommendations. When such a combination is performed, the recommendation approach is considered a hybrid recommender (or hybrid filtering) system (Burke, 2002).

2.2 Recommendation techniques

As mentioned above, the main goal of a recommender system is to provide users with the most useful items according to their preferences. For such purpose, different strategies may be used, which can be categorised based on the type of data exploited, namely content-based, rating- and log-based collaborative filtering, and social recommendation strategies. In this section we formalise these strategies. We shall use the following notation. Letters u and v will be reserved for users ($u, v \in \mathcal{U}$), whereas i and j will denote items ($i, j \in \mathcal{I}$), $\mathcal{U}$ and $\mathcal{I}$ being, respectively, the set of users and items in the system. Besides, $r \in R$ will denote a particular rating value, and R will be the set of possible rating values, either discrete (typically, $R = \{1,2,3,4,5\}$) or continuous (e.g. $R = [0,5]$). Finally, $\tilde{r}$ shall denote a rating prediction (as opposed to observed ratings denoted by r).

2.2.1 Content-based recommenders

Content-based filtering (CBF, or simply content-based) techniques recommend items similar to those previously liked by a user. An extensive survey of this type of techniques can be found in (Lops et al., 2011; Pazzani and Billsus, 2007), and

(Adomavicius and Tuzhilin, 2005). In this section we briefly discuss some of the main approaches proposed in the field.

Content-based recommendation algorithms build a user's profile based on the features of the objects rated by the user, which are assumed to reflect the user's content-based interests (Lops et al., 2011). In general, a CBF technique can be classified according to whether a model is built from underlying data, commonly based on Machine Learning techniques (Lops et al., 2011; Pazzani and Billsus, 1997; de Gemmis et al., 2008), or use a heuristic function to compute item scores, mainly inspired on Information Retrieval methods (Diederich and Iofciu, 2006; Balabanovic and Shoham, 1997; Cantador et al., 2010).

Probabilistic methods in general and the naïve Bayes approach in particular generate a probabilistic model based on previously observed data. The naïve Bayes model estimates the *a posteriori* probability $P(c|d)$ of document d belonging to class c, based on the *a priori* probability $P(c)$ for the class, the probability $P(d)$ of observing the document, and the probability $P(d|c)$ of observing the document given the class (Lops et al., 2011), as follows:

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

In recommendation the naïve Bayes method is used to estimate the probability that a document (an item) is either relevant or irrelevant (class c), based on the information available for each user, that is, documents already rated are used to build the $P(d|c)$ probabilities. This approach has been used by many different authors (Mooney and Roy, 2000; Semeraro et al., 2007; de Gemmis et al., 2008; Lops et al., 2011).

Alternative methods for classifying the items in a system as relevant or irrelevant for each user include decision trees and neural networks (Pazzani and Billsus, 1997). These techniques, similarly to the naïve Bayes method, estimate in which class each (unrated or unobserved) item fits best with the user's profile.

Techniques based on Information Retrieval methods are specified by the way users and items are represented and the similarity function used between them. Typically they use a vector space model where each feature is weighted in a particular way. For instance, instead of using the frequency of each feature in a user/item profile, more complex functions from the Information Retrieval field may be used, such as TF-IDF and BM25 (Cantador et al., 2010). Furthermore, many different feature spaces have been considered in the literature: keywords (Lieberman, 1995; Pazzani et al., 1996), tags (Diederich and Iofciu, 2006; Michlmayr and Cazer, 2007), and semantic concepts enriched by different techniques (Magnini and Strapparava, 2001; Eirinaki et al., 2003; Cantador, 2008).

Regarding the feature vector similarity, the most common measure is the cosine similarity, even though the standard dot product between two vectors has also been used (Cantador et al., 2010):

$$sim_{dot}(d_i, d_j) = \sum_k w_{ki} w_{kj} \tag{2.1}$$

$$sim_{cos}(d_i, d_j) = \frac{sim_{dot}(d_i, d_j)}{\sqrt{\sum_k w_{ki}^2} \sqrt{\sum_k w_{kj}^2}} \tag{2.2}$$

where w_{ki} is the weight assigned (by any of the techniques mentioned before) to the feature k in document i.

In recommender systems items are suggested by decreasing order of similarity with the user, whose profile is represented in the same form of the documents (that is, in the space of features under consideration). The similarities are computed as the feature vector similarity between each (unrated or unobserved) document in the collection and the user's vector.

2.2.2 Rating-based recommenders

Collaborative filtering (CF) techniques match people with similar preferences, or items with similar choice patterns from users, in order to make recommendations. Unlike CBF, CF methods aim to predict the utility of items for a particular user according to the items previously evaluated by other like minded users. These methods have the interesting property that no item descriptions are needed to provide recommendations, since the methods merely exploit information about past ratings. Compared to CBF approaches, CF also has the salient advantage that a user may benefit from other people's experience, thereby being exposed to potentially novel recommendations beyond her own experience (Adomavicius and Tuzhilin, 2005).

In this section we focus on those CF techniques based on explicit numeric ratings, which are the most common in the literature. For additional references, see (Desrosiers and Karypis, 2011; Koren and Bell, 2011) and (Adomavicius and Tuzhilin, 2005). Most of our discussion nonetheless applies to log-based recommenders alike. In fact, as we shall show in the next section, most of the rating-based techniques can be used when no ratings are available (although the equivalence introduces additional assumptions).

In general, CF approaches are commonly classified into two main categories: **model-based** and **memory-based**. Model-based approaches build statistical models of user/item rating patterns to provide automatic rating predictions. Some approaches learn such models by performing some form of dimensionality reduction in order to uncover latent factors between users and items, e.g. by such techniques as

Singular Value Decomposition (SVD) for matrix factorisation (Billsus and Pazzani, 1998; Koren et al., 2009), probabilistic Latent Semantic Analysis (pLSA), or Latent Dirichlet Allocation (LDA) (Hofmann, 2003; Blei et al., 2003). Other approaches use probabilistic models where the recommendation task is modelled by user and item probability distributions (Wang et al., 2006b; Wang et al., 2008a), e.g. by learning a probabilistic model with a maximum entropy estimation (Pavlov et al., 2004; Zitnick and Kanade, 2004), Bayesian networks (Breese et al., 1998), and Boltzmann machines (Salakhutdinov et al., 2007). A graph-based model that exploits positive and negative preference data is proposed in (Clements et al., 2009). Besides, other Machine Learning techniques have also been proposed, such as artificial neural networks (Billsus and Pazzani, 1998) and clustering strategies (Kohrs and Merialdo, 1999; Cantador and Castells, 2006).

Memory-based approaches, on the other hand, make rating predictions based on the entire rating collection (Adomavicius and Tuzhilin, 2005; Desrosiers and Karypis, 2011). These approaches can be user- and item-based strategies. **User-based** strategies are built on the principle that a particular user's rating records are not equally useful to all other users as input for providing personal item suggestions (Herlocker et al., 2002). Central aspects to these algorithms are thus a) how to identify which neighbours form the best basis to generate item recommendations for the target user, and b) how to properly make use of the information provided by them. Typically, neighbourhood identification is based on selecting those users who are more similar to the target user according to a similarity metric (Desrosiers and Karypis, 2011). The similarity between two users is generally computed by a) finding a set of items that both users have interacted with, and b) examining to what degree the users displayed similar behaviors (e.g. rating, browsing and purchasing patterns) on these items. This basic approach can be complemented with alternative comparisons of virtually any user feature a system has access to, such as personal demographic and social network data. It is also common practice to set a maximum number of neighbours (or a minimum similarity threshold) to restrict the neighbourhood size either for computational efficiency, or in order to avoid noisy users who are not similar enough. Once the target user's neighbours are selected, the more similar a neighbour is to the user, the more her preferences are taken into account as input to produce recommendations. For instance, a common user-based approach consists of predicting the relevance of an item for the target user by a linear combination of her neighbours' ratings, weighted by the similarity between the target user and such neighbours.

In the following equations we present two versions of a user-based CF technique; in the first one rating deviations from the user's and neighbour's rating means are considered (Resnick et al., 1994), whereas in the second one the raw scores given by each neighbour are used (Aggarwal et al., 1999; Shardanand and Maes, 1995):

$$\tilde{r}(u,i) = \bar{r}(u) + C \sum_{v \in N_k(u,i)} sim(u,v)\big(r(v,i) - \bar{r}(v)\big) \tag{2.3}$$

$$\tilde{r}(u,i) = C \sum_{v \in N_k(u,i)} sim(u,v) r(v,i) \tag{2.4}$$

where C is a normalisation factor (different in each formulation) and $N_k(u,i)$ is a neighbourhood of size k, which may use information of the target item i. As stated in (Adomavicius and Tuzhilin, 2005), these techniques simply use a different function to aggregate the ratings from the neighbourhood. Note that in this case the utility function $g(u,i)$ is assumed to be equivalent to the predicted rating $\tilde{r}(u,i)$, although alternative transformations could be applied if required. Additionally, the similarity between two users is generally computed by means of the Pearson's correlation coefficient or the cosine similarity between the vectors representing each user's preferences (Adomavicius and Tuzhilin, 2005):

$$sim_{Pearson}(u,v) = \frac{\sum_i \big(r(u,i) - \bar{r}(u)\big)\big(r(v,i) - \bar{r}(v)\big)}{\sqrt{\sum_i \big(r(u,i) - \bar{r}(u)\big)^2} \sqrt{\sum_i \big(r(v,i) - \bar{r}(v)\big)^2}} \tag{2.5}$$

$$sim_{Cosine}(u,v) = \frac{\sum_i r(u,i) r(v,i)}{\sqrt{\sum_i r(u,i)^2} \sqrt{\sum_i r(v,i)^2}} \tag{2.6}$$

Note that these similarities are equivalent when the data is centered on the mean. Nonetheless, some authors have reported that the performance of recommenders based on Pearson's similarity is superior to that of cosine's (Breese et al., 1998; Herlocker et al., 1999). Moreover, other similarity measures and modifications on how the neighbours are selected and weighted have been proposed, either by modifying the similarity measure (McLaughlin and Herlocker, 2004; Ma et al., 2007), by using clustering methods to compute a user's neighbourhood (O'Connor and Herlocker, 1999; Xue et al., 2005), or by learning the best interpolation weights for rating prediction (Bell and Koren, 2007; Koren, 2008). Additionally, in the context of trust-based recommendation, the neighbours are weighted (and selected) according to their importance from the target user's point of view (O'Donovan and Smyth, 2005; Weng et al., 2006; Kwon et al., 2009; Hwang and Chen, 2007).

Item-based strategies, on the other hand, recognise patterns of similarity between the items themselves, instead of between user choices like user-based approaches do. In general item-based recommenders look at each item on the target user's list of chosen/rated items, and find other items that seem to be "similar" to that item (Shardanand and Maes, 1995; Sarwar et al., 2001). The item similarity is usually defined in terms of rating correlations between users, although cosine-based or probability-based similarities have also been proposed (Deshpande and Karypis,

2004). As stated in (Sarwar et al., 2001), adjusted cosine similarity has been proved to obtain better performance than other item similarities. This similarity subtracts the user's average rating from each co-rated pair in the standard cosine formulation:

$$sim(i,j) = \frac{\sum_u \big(r(u,i) - \bar{r}(u)\big)\big(r(u,j) - \bar{r}(u)\big)}{\sqrt{\sum_u \big(r(u,i) - \bar{r}(u)\big)^2}\sqrt{\sum_u \big(r(u,j) - \bar{r}(u)\big)^2}} \tag{2.7}$$

The rating prediction computed by item-based strategies is generally estimated as follows (Sarwar et al., 2001):

$$\tilde{r}(u,i) = C \sum_{j \in S_i} sim(i,j) r(u,j) \tag{2.8}$$

We have to note that the set of more similar items S_i is generally replaced by $\mathcal{I}_u$ – the set of items rated by user u – since for any other item, the rating provided by the user is assumed to be zero, and thus, it does not contribute to the summation.

2.2.3 Log-based recommenders

Different methods have been proposed to use implicit evidence of user preferences. The work of Oard and Kim (1998) represents one of the first attempts to exploit implicit user feedback to estimate future ratings in a recommender system. In general most of recent approaches have used formal models (generally probabilistic) in order to introduce implicit data for recommendation, although some approaches using ad-hoc techniques can be found. For example, Linden et al. (2003) use a simple vector representation, where each component represents purchased items, and recommendations are obtained by ranking each item according to how many similar users purchased it. Bernhardsson (2009) proposes a graph item-based algorithm that finds the closest tracks for a given track using probabilistic LSA (pLSA), and then derives the recommendations using heuristic and model-based probabilities, by brute force.

Additionally, several formal algorithms have been proposed to use implicit user feedback from log data: namely matrix factorisation, such as SVD (Hu et al., 2008) and pLSA (Das et al., 2007), and language models and other probabilistic approaches (Wang et al., 2006a; Wang, 2009; Wang et al., 2008b; Deshpande and Karypis, 2004). These algorithms aim to capture the user's preferences by considering the consumed (purchased, listened, browsed, etc.) items as evidence of positive relevance for the user. This fact often leads to binary models in which the number of times the user has consumed each item is not taken into consideration. Nevertheless, a benefit of using binary data is that it allows to better account for the fact that ratings are not missing at random – or equivalently, that users choose deliberately which items to rate (Marlin et al., 2007). Besides this a general concern about negative preferences has arisen. For instance, in (Lee and Brusilovsky, 2009), (Wang et al., 2008c), and

(Xin and Steck, 2011) the authors attempt to incorporate negative preferences inferred from implicit data.

Other authors have proposed different transformations in order to obtain explicit ratings from implicit feedback. The most naive approach is to make a correspondence between the existence of an item in a log record and a (frequency-independent) rating. For example, the algorithm proposed in (Ali and van Stam, 2004) cannot distinguish an explicit +1 rating from the rating inferred from implicit data. This is the same procedure that can be found in (Lee et al., 2008), but with other transformations based on the time in which an item was entered in the system and consumed.

In (Baltrunas and Amatriain, 2009), based on (Celma, 2010) and (Celma, 2008), the authors use a more elaborate mapping where the number of times a user listened to an artist (or track) is taken into account, in such a way that the artists (tracks) located in the 80-100% interquintile range of the user's playcount distribution receive a rating of value 5 (in a five point scale), the next interquintile range is mapped to a rating of value 4, and so on. This technique has also been used in other works, such as (Vargas and Castells, 2011). A similar technique is presented in (Jawaheer et al., 2010), where three methods are proposed in order to calculate the preference of a user for an artist: i) absolute, where the raw count of the number of times that artist has been played is used; ii) normalised, where the preference is inferred by the ratio between the counts for an artist and the total number of artists played by the user; and iii) logarithmic, similar to the previous one but smoothing the preference values by applying a logarithmic transformation.

Finally, Adams (2007) proposes a complete ad-hoc formula that takes several parameters into consideration, such as the number of times the current track has been played and skipped, the number of seconds when it was skipped, and the number of days since it was last played.

In conclusion, there is no definitive unique method for transforming implicit into explicit data. Moreover, it is unclear to what extent the mapping is reliable (Hu et al., 2008), since it inherently represents different information gathered from the user – for instance, negative preferences can only be fetched using explicit data. However, a recent study reported a strong relation between the amount of times users listen to an album, and the rating they provide to the album (Parra and Amatriain, 2011).

2.2.4 Social-based recommenders

Recommender systems that exploit social information, such as contacts and interactions between users, have started to be developed in recent years. We shall henceforth refer to this type of recommendation approaches as Social Filtering (SF) systems. Recommendations by SF approaches have the interesting property that they

are generally easier to explain than user-based CF approaches. Recommendations through friends are indeed easy to interpret by end-users. They also help dealing with the cold start problem, where new users are more difficult to provide recommendations for as long as it is not possible to reliably compute their similarity with other users for lack of data (Golbeck, 2006; Arazy et al., 2009).

Shepitsen et al. (2008) propose a personalisation approach for recommendation in folksonomies that relies on hierarchical tag clusters. The approach suggests the most similar items to the user's closest cluster by means of the cosine similarity measure. Other approaches focus on graph based techniques for finding the most relevant items for a particular user through hybrid networks involving people, items, and tags (Konstas et al., 2009; Clements et al., 2010). In this context alternative methods have been proposed to deal with data sparsity. Besides, prediction accuracy is improved by means of factor analysis based on probabilistic matrix factorisation, employing both the users' social network information and rating records (Ma et al., 2008). Ma et al. (2009) combine the recommendations made by trusted friends with those generated by a matrix factorisation algorithm. In a similar way, Jamali and Ester (2009) propose to perform a random walk on the trust network, considering the similarity of users in the termination condition; then, the top rated items are recommended. Both approaches are competitive in cold start situations.

Complementarily, simpler algorithms (referred to as "pure" social recommenders henceforth) have also been proposed in (Liu and Lee, 2010) and (Bellogín et al., 2012). In (Liu and Lee, 2010) an adaptation of the user-based CF technique is proposed, where the set of nearest neighbours is replaced by the target user's (explicit) friends. That is:

$$N_k(u,i) = \{v \in \mathcal{U}: v \text{ is friend of } u\} \tag{2.9}$$

This lets easily incorporate social information into the CF prediction equation, building a straightforward technique that enables a direct interpretation of the suggestions, namely those items recommended by friends. Similarly, based on a recommender proposed in (Barman and Dabeer, 2010), where the items suggested to a user are the most popular among her set of similar users, in (Bellogín et al., 2012) we proposed a friends' popularity recommender that suggests the target user those items most popular for her set of friends. A score is generated by transforming the item position with the following equation, once a ranking has been generated using the score $f(u,i)$:

$$\begin{aligned} f(u,i) &= |\{v \in \mathrm{U}: v \text{ is friend of } u \text{ and } rat(v,i) \neq \phi\}| \\ g(u,i;N) &= 1 - \frac{pos(u,i)}{N} \end{aligned} \tag{2.10}$$

where $pos(u,i)$ represents the position of item i in the top-N recommended list for user u. We may trim the returned list at some level N, or assume N to be exactly the

length of the generated recommendation list. Obviously, the computed scores cannot be interpreted as ratings, but as a utility or ranking score. In (Bourke et al., 2011) Bourke and colleagues also make use of the social graph of a user to build the neighbourhood, analysing the perceived trust, which is found to be higher when the users are given the opportunity to manually select the neighbourhood to be used for computing recommendations.

Ben-Shimon et al. (2007) propose a recommendation approach based on the distances between users in the social graph. The approach uses Breadth-First Search to build a social tree for each user u denoted as $X(u, L)$, where L is the maximum number of levels taken into consideration in the algorithm, and K is an attenuation coefficient of the social network that determines the extent of the effect of $d(u, v)$, that is, the impact of the distance between two users in the social graph (e.g. by using an algorithm that computes the distance between two nodes in a graph, such as the Dijkstra's algorithm (Dijkstra, 1959). Hence, when $K = 1$ the impact is constant, and the resulting ranking is sorted by the popularity of the items. Furthermore, for that value of K, no expansion is applied and only directly connected users are involved in the score computation. Once the value of K is chosen, a rating score is generated according to the following equation:

$$g(u, i) = \sum_{v \in X(u,L)} K^{-d(u,v)} r(v, i) \tag{2.11}$$

An alternative way of introducing social information into a recommender system is by the so called trust-based recommendation approaches, even though social relationships and trust relationships do not model exacly the same concept (Ma et al., 2011). Trust-aware recommenders, in contrast with those defined in Section 2.2.2, make use of trust networks, where users express a level of trust on other users (Massa and Avesani, 2007a). These recommenders need a trust network and a trust metric, so that trustworthiness of every user can be computed. Depending on the available data, we would have to infer a plausible trust network, from the information we already know about users, such as social interactions among users or explicit trust relations. Typically, uniform trust values from each user are assumed, since no distinction can be made among a user's contacts. For example, a user with 4 friends would have a trust level of 0.25 for each friend, whereas a user with 2 friends would have such trust level of 0.5.

Once the trust network is defined, either explicitly or implicitly, we can set different definitions for the trust metrics depending on whether they are global (a global reputation value is calculated for each user) or local (a trust score is computed between a source user on a target user). Social-based trust metrics make use of explicit trust networks of users, built upon friendship relationships (Massa and Bhattacharjee, 2004) and explicit trust scores between individuals in a system (Ma et al., 2009; Wal-

ter et al., 2009). These metrics and, to some extent, their inherent meanings, are different with respect to rating-based metrics. Nonetheless, Ziegler and Lausen (2004) conduct a thorough analysis that shows empirical correlations between trust and user similarity, suggesting that users tend to create social connections with people who have similar preferences. Once such a correlation is proved, techniques based on social-based trust are applicable.

Golbeck and Hendler (2006) propose a metric called TidalTrust to infer trust relationships by recursive search. Inferred trust values are used for every user who has rated a particular item in order to select only those users with high trust values. Then, a weighted average between ratings and trust provides the predicted ratings. A similar algorithm is used in (Walter et al., 2009), where the prediction is based on the ratings of the trusted neighbours. Different integrations of the trust metric into the recommendation process are proposed in (Massa and Avesani, 2007a), along with two metrics: PageRank and MoleTrust. The former is considered as a global metric based on the well-known PageRank algorithm (Brin and Page, 1998); the latter is a local metric based on a Depth-First graph traversal algorithm with an adjustable trust propagation horizon (Massa and Avesani, 2007a).

Finally, as proposed in (Massa and Avesani, 2007a), two ways to incorporate these trust metrics into the recommendation models can be considered. The first one makes use of the trust metric instead of the similarity metric in the standard user-based CF formula. The second one, on the other hand, computes the average between Pearson's similarity and the trust metric when both values are available; otherwise it uses the only available value, thus overcoming the natural data sparsity. Recently, Guo et al. (2012) propose to merge the ratings from the trusted neighbours in order to decrease sparsity prior to the computation of the predicted rating.

2.3 Combining recommender systems

The proliferation of new recommendation strategies is giving rise to an increasing variety of available options for the development of recommender systems. Research in Machine Learning has long shown that the combination of methods usually achieves better results than each method separately, which is also true in Recommender Systems – the Netflix prize has been a paradigmatic example of this, where all the top classified teams used large recommender ensembles, which can be considered as a case of hybrid filtering approaches.

In such a hybrid approach the most important decision is how to combine the information. First, however, it has to be decided what kind of information is going to be used in the ensemble. The standard approach in the literature is to combine CBF and CF recommenders, overcoming the sparsity and restricted feature problems of individual recommenders, as we shall see in the next section. However, other types

and sources of information, such as social contacts and timestamps, have been recently integrated into the classical formulation of standard recommendation techniques.

In (Burke, 2002) a detailed taxonomy of hybrid recommender systems is presented, classifying existing approaches into the following types:

- **Cascade**: the recommendation is performed as a sequential process in such a way that one recommender refines the recommendations given by the other.
- **Feature augmentation**: the output from one recommender is used as an additional input feature for other recommender.
- **Feature combination**: the features used by different recommenders are integrated and combined into a single data source, which is exploited by a single recommender.
- **Meta-level**: the model generated by one of the recommenders is used as the input for other recommender. As stated in (Burke, 2002): "this differs from feature augmentation: in an augmentation hybrid, we use a learned model to generate features for input to a second algorithm; in a meta-level hybrid, the entire model becomes the input."
- **Mixed**: recommendations from several recommenders are available, and are presented together at the same time by means of certain ranking or combination strategy.
- **Weighted**: the scores provided by the recommenders are aggregated using a linear combination or a voting scheme.
- **Switching**: a special case of the previous type considering binary weights, in such a way that one recommender is turned on and the others are turned off.

The use of a specific type of hybrid recommendation method depends on the final application, but, more importantly, on the type of recommenders being combined. Indeed, Burke (2002) presents an analysis of the possible hybrids, their limits and incompatibilities, based on a representative subset of the recommendation techniques available nowadays. Moreover, the author notes that some combinations turn out to be redundant because of the symmetry in the hybridisation process for some of the techniques listed above: weighted, mixed, switching, and feature combination. Incompatible combinations arise for the feature combination and meta-level techniques, where in some situations one of the recommenders is not able to use the model or the features generated by the other recommender.

Burke (2002) focuses on hybrid techniques where the information being combined consists of ratings (to be used by CF recommenders), content features (to be used by CBF, knowledge-based, and utility-based recommenders), and demographic

information. In the following, we survey hybrid recommenders where inputs in the form of social information, collaborative (either ratings or logs), and content features have been used. In the next section we analyse the limitations of these types of techniques, together with the benefits that hybridisation may bring.

Among these possibilities, the most popular combination (probably due to its inherent interest) consists of blending content-based and collaborative filtering recommenders. In fact, one of the first proposed hybrid techniques (Balabanovic and Shoham, 1997) makes use of these two recommendation approaches by suggesting items similar to the user's profile (using content-based profiles) and those items highly rated by a user with a similar profile, by means of a collaborative formulation where neighbours are determined using a content-based similarity. In a similar way, Pazzani (1999) combines content-based, demographic, and collaborative information using two techniques: by plugging content-based similarity functions into collaborative methods and by combining the final rankings produced by each recommender seeking a consensus, that is, how many systems recommend each item, and in what ranking position are both considered to build the final ranking.

In (Rojsattarat and Soonthornphisaj, 2003) a technique to derive a less sparse pseudo rating matrix is proposed. More specifically, a pseudo user-ratings vector for every user is built with the item ratings provided by user u when available, or the ratings predicted by a content-based recommender otherwise. Gunawardana and Meek (2009) propose to combine content and collaborative information in a coherent manner by using a specific type of probabilistic models, Boltzmann machines. These models let encoding the above sources of information as features, and then, weights are learned to reflect how each feature helps predict the user ratings. Other probabilistic models for combining these sources of information have been proposed in (Yu et al., 2003), where a hierarchical Bayesian model learns a prior distribution by using probabilistic Support Vector Machines (SVMs).

Also from a machine learning perspective, an ensemble technique known as stacking is used in (Bao et al., 2009), which learns multiple classifiers for different prediction levels: at the first level, the recommendation techniques (a user-based CF, an item-based CF, and a CBF algorithm) output a rating prediction, which may be combined at the second level by a meta-learning algorithm that uses the predictions as meta-features.

Alternatively, the same model can also be combined with itself using different parameter values. For instance, in (Gantner et al., 2010) different factor models are combined, where each model may have different regularisation parameters, stop conditions and dimensionality values. Jahrer et al. (2010) combine a set of diverse CF recommenders by using different machine learning techniques such as linear regression, neural networks, and a combination of bagging and gradient boosting trained with decision trees.

Furthermore, hybrid models have been proposed combining social and content or collaborative information. In (Konstas et al., 2009) a Random Walk algorithm is applied to a graph comprising of tags, social information, and implicit feedback from users. In this way, more elaborate patterns and rules than the standard correlation measure between users are provided. A similar approach can be found in (Liu et al., 2010) for tag recommendation. The approach defined in (Clements et al., 2010) improves search and recommendation by combining tags and ratings, and integrating them into the user's social network also using a Random Walk algorithm. Hotho et al. (2006) exploit social information along with tag content by converting a folksonomy into a graph and then applying a weight-spreading algorithm for folksonomies called FolkRank (similar to the well-known PageRank algorithm (Brin and Page, 1998). Finally, Jamali and Ester (2009) combine information from the social network (in terms of trust between users) and ratings (collaborative) in order to alleviate the cold-start problem. In that work, the authors make use of the collaborative information as a termination condition of a random walk performed over the trust network by considering the similarity of users; additionally, the authors also combine those two sources for computing two sets of neighbours and, then, merging the items produced from those similar users.

2.4 General limitations of recommender systems

Each type of recommendation technique has strengths and weaknesses, well known in the field. We have already noted the main characteristics of each technique, which are largely dependent on the source of information being used. In this section we analyse the main limitations of each technique. Furthermore, although ideally hybrid recommendation techniques would overcome the problems of the combined techniques, there are certain limitations that are inherent to the recommendation problem, and thus, have to be addressed indepedently. Besides, by combining different methods, additional problems, along with more limitations, arise.

2.4.1 Limitations of single recommendation algorithms

In this section we describe the different limitations identified in the literature for the main types of recommenders described in the previous sections.

The main limitations of CBF approaches are the following (Adomavicius and Tuzhilin, 2005; Pazzani and Billsus, 2007; Cantador, 2008):

- **Restricted content analysis**. Content-based recommendations depend on the available features explicitly associated with the items. These features should be in a form that can be automatically parsed by a computer, or manually ex-

tracted somehow, which, depending on the domain, could be unfeasible or very difficult to maintain.

- **New user**. A user has to show some preference (ratings) for a sufficient number of items before a recommender can build a reliable content-based user profile.
- **Overspecialisation**. Since content-based recommenders only retrieve items similar to what the user has already rated, recommendations are very similar and, probably, well known to the user, providing little (or none) novelty from the user perspective.
- **Portfolio effect**. Related to the previous limitation, sometimes the recommended items are very similar among them, leading to a set of insufficiently diverse or too redundant item suggestions.

CF approaches have the following general weaknesses:

- **Rating data sparsity**. The number of observed user-item interactions (e.g. ratings) is generally very small compared to the number of all user-item pairs. This fact may cause CF algorithms to produce unreliable recommendations, since they have been inferred from insufficient data.
- **Grey sheep**. Since collaborative recommendations rely on the tastes of similar people to suggest new items, when a user has very specific or unusual preferences, it will be more difficult for the system to find good neighbours, and thus, to recommend interesting items.
- **New item**. Until a new item has been rated by a substantial number of users, a recommender system may not be able to recommend it; hence, popular items tend to have advantage in this kind of systems.
- **New user**. Like in the content-based approaches, until a user has not provided with enough ratings, the system is unable to recommend her interesting unknown items.

In addition to these weaknesses, log-based CF techniques have other limitations. Specifically, they are not able to capture negative preferences from the user since unobserved items cannot be inferred as unliked items (they may represent items unknown for the user). In contrast, it is easier to capture this type of information because it is less expensive for the user than providing a rating. Furthermore, although the problem of ratings missing not at random is ubiquitous and inherent to any recommender system – since users typically rate only a small fraction of the available items – log-based recommenders, and more specifically, the binary data inferred from these implicit interactions, have the theoretical advantage that they are able to exploit implicit preferences since the items observed by the users are deliberately

Problem	Description	CBF	CF	SF
Restricted content analysis	Items to be recommended must have available data related to their features. This data is often unavailable or incomplete.	Yes	No	No
Overspecialisation	CBF recommenders are trained with the content features of the items. All the recommended items are similar to those already rated.	Yes	No	No
Portfolio effect	CBF recommenders suggest items based on the item features. An item is recommended even if it is too similar to a previously rated item.	Yes	No	No
New user	A user has to rate enough items in order to infer their preferences. When a new user enters into the system she has no ratings.	Yes	Yes	No
New item	Items have to be rated by a substantial number of users for being recommended. Recently incorporated items have none or insufficient ratings.	No	Yes	No
Grey sheep	A user has to be similar to others in the community to receive recommendations. Users whose tastes are unusual may not receive useful suggestions.	No	Yes	No
Rating data sparsity	Ratings are used to train user and item models. The number of available ratings is usually small.	No	Yes	No
Social sparsity	Social connections are used to build social models. The number of connections per user may be small.	No	No	Yes
New social connection	A user has to be connected with someone else to receive recommendations. When the user is new, she may not have any social connections.	No	No	Yes
Social similarity	Similarity based on social connections is used in SF recommenders. Two users socially connected may or may not have interests in common.	No	No	Yes

Table 2.1. List of common problems in CBF, CF, and SF systems.

selected by them. Thus, potentially more useful information about the user can be gathered (Koren and Bell, 2011).

Regarding CF in general, memory-based approaches achieve lower performance than model-based approaches. However, as stated in (Desrosiers and Karypis, 2011) and (Koren and Bell, 2011), good prediction accuracy does not guarantee an effective and satisfying user experience. Hence, the main advantages of memory-based recommenders are simplicity, justifiability, efficiency, and stability.

Finally, SF approaches have other limitations, as we describe next:

- **Social sparsity**. Social filtering methods need that every user has to be connected through at least one contact in the social network to be able to produce recommendations, which is not a typical situation for most of the users in a system.

- **New social connection**. Recommendations may get biased if a user has a very small social network, up to the point that if she has only one connection, every social recommendation would be generated based on the activity of just one user.
- **Social similarity**. The fact that two users share some kind of connection in a social network does not necessarily mean that these users have similar interests. Although some studies have shown some correlation between both (Ziegler and Lausen, 2004), the misuse of this similarity may lead to bad recommendations, even though the user's experience may be improved in terms of diversity and serendipity.

As a summary, Table 2.1 shows a comparison of the main limitations for the three types of recommendation algorithms described.

2.4.2 Limitations of recommender ensembles

As we have explained in the previous section, each type of recommendation – CBF, CF, and SF – has its own limitations. Hybrid filtering systems are normally out of this analysis since they compensate the shortcomings of one approach by the strengths of the other, unless both suffer from the same problem, as in the case of a new user when we combine CBF and CF approaches.

In general, hybrid recommenders are useful for alleviating the individual limitations of the combined recommenders. However, recommender ensembles do not always outperform individual recommenders. Van Setten (2005) describes the situation where all recommenders produce predictions that are "on the same side of the rating the user would give, all too low or all too high." In this situation the ensemble would be less accurate than the best individual recommender. Additionally, when a particular recommender always obtains superior/inferior performance than the rest of recommenders in the ensemble, the corresponding recommender ensemble may not be useful. In that case the underperforming recommenders are useless from the beginning, whereas the over performing one should be used alone, and there is no point in combining them.

The above issues assume that a particular metric is aimed to be optimised. Needless to say that the use of multiple recommenders may provide better results with respect to other evaluation properties, such as diversity, novelty, and serendipity, probably at the expense of a lower quality or accuracy of the recommendations (Shani and Gunawardana, 2011).

Additionally, the recommender ensemble problem is similar to that of combining classifiers in the Machine Learning field, a well studied research problem in that community (Kuncheva, 2004). In such context, the diversity in the classifier outputs is known to be a requirement for the combination to be effective. Thus, whenever

some classifiers in an ensemble fail, these errors should be made on different objects, in order to let a final performance improvement with the ensemble. In (Kuncheva, 2004) and (Kuncheva and Whitaker, 2003) Kuncheva and Whitaker present a number of diversity metrics, and analyse the relation of such metrics with respect to the accuracy of a recommender ensemble, although they do not provide a systematically formulation of such relation. As stated by the authors, the problem of classifier combination and its relation with diversity may rise from the underlying meaning of diversity: whether it is a characteristic of the set of classifiers, or it is more complex and a mixture of the characteristics of the set of classifiers, the combiner, and the errors.

Finally, although many different hybrid filtering approaches have been proposed for recommender systems, there is a lack of a similar analysis to the one performed in Machine Learning, where the different characteristics of the datasets and individual recommenders have been investigated and assessed. A preliminary analysis was performed in (Bellogín et al., 2010), but an in-depth and larger-scale study would benefit the community, considering different evaluation perspectives and, probably, borrowing from the Machine Learning research on this topic.

2.5 Summary

Along over two decades of research and commercial development, recommender systems have proved to be a successful technology to overcome the information overload that burdens users in modern online media. The inherent possibility of dealing with diverse sources of information, such as the content of the items and the collaborative and social interactions among users and between users and a system, has enabled the development of rich strategies based on each of these evidences, deriving content-based, collaborative, and social filtering recommendation approaches. Furthermore, as each particular type of recommendation technique has its own limitations and weaknesses, hybrid strategies have been proposed that combine the suggestions generated by different techniques in different ways. The success of ensemble approaches has been recently evidenced in the Netflix prize, where the top classified teams used different forms of recommender ensembles.

There are, however, general limitations remain unsolved, and are still considered as open research problems in the field. We have mentioned the sparsity of the information (either in the forms of content-based attributes, collaborative ratings, and social connections), and the new user problem, but other problems, not related to an specific recommendation technique, have been identified in the literature, and deserve special attention by themselves, such as the need of contextualisation, the explanation of the recommendations, and the efficiency in computing recommendations (Adomavicius and Tuzhilin, 2005).

Part II

Evaluating performance in recommender systems

If you cannot measure it, you cannot improve it.
William Thomson (Lord Kelvin)

Chapter 3

Evaluation of recommender systems

The evaluation of recommender systems has been, and still is, the object of active research in the field. Since the advent of the first recommender systems, recommendation performance has been usually equated to the accuracy of rating prediction, that is, estimated ratings are compared against actual ratings, and differences between them are computed by means of the *mean absolute error* and *root mean squared error* metrics. In terms of the effective utility of recommendations for users, there is however an increasing realisation that the quality (precision) of a ranking of recommended items can be more important than the accuracy in predicting specific rating values. As a result, precision-oriented metrics are being increasingly considered in the field, and a large amount of recent work has focused on evaluating top-N ranked recommendation lists with the above type of metrics.

In this chapter we provide a survey of different evaluation metrics, protocols, and methodologies in the recommender systems field. In Section 3.1 we provide a preliminary overview of how recommender systems are evaluated, presenting the main (online and offline) evaluation protocols and dataset partitioning methods. Next, in Section 3.2 we present the most common evaluation metrics, classified into error-based and precision-based metrics, and in Section 3.3 we describe different dataset partition strategies used in the experimental configurations. Finally, in Section 3.4 we present some evaluation datasets which are commonly used by the research community, and that were used in the experimental work of this thesis.

3.1 Introduction

The evaluation of recommender systems has been a major object of study in the field since its earliest days, and is still a topic of ongoing research, where open questions remain (Herlocker et al., 2004; Shani and Gunawardana, 2011). Two main evaluation protocols are usually considered (Gunawardana and Shani, 2009): *online* and *offline*. In this thesis we focus on offline evaluation, which lets compare a wide range of candidate algorithms at a low cost (Shani and Gunawardana, 2011). For a review of the different tasks and protocols for online recommendation evaluation, see (Shani and Gunawardana, 2011), (Pu et al., 2012), and (Kohavi et al., 2009).

Drawing from methodological approaches common to the evaluation of classification, machine learning and information retrieval algorithms, offline recommender system evaluation is based on holding out from the system a part of the available knowledge of user likes (test data), leaving the rest (training data) as input to the algorithm, and requiring the system to predict such preferences, so that the goodness of recommendations is assessed in terms of how the system's predictions compare to the withheld known preferences. In the dominant practice, this comparison has been oriented to measure the accuracy of rating prediction, computing error-based metrics. However, in terms of the effective utility of recommendations for users, there is an increasing realisation that the quality (precision) of the ranking of recommended items can be more important than the accuracy (error) in predicting specific rating values. As stated in (Herlocker et al., 2004), the research community has moved from the *annotation in context* task (i.e., predicting ratings) to the *find good items* task (i.e., providing users with a ranked list of recommended items), which better corresponds to realistic settings in working applications where recommender systems are deployed. As a result, precision-oriented metrics are being increasingly considered in the field. Yet there is considerable divergence in the way such metrics are applied by different authors, as a consequence of which the results reported in different studies are difficult to put in context and be compared.

In the classical formulation of the recommendation problem, user preferences for items are represented as numeric ratings, and the goal of a recommendation algorithm consists of predicting unknown ratings based on known ratings and, in some cases, additional information about users, items, and the context. In this scenario, the accuracy of recommendations has been commonly evaluated by measuring the error between predicted and known ratings, using metrics such as the Mean Absolute Error (MAE), and the Root Mean Squared Error (RMSE). Although dominant in the literature, some authors have argued this evaluation methodology is detrimental to the field since the recommendations obtained in this way are not the most useful for users (McNee et al., 2006). Acknowledging this, recent work has evaluated top-N ranked recommendation lists with precision-based metrics (Cremonesi et al., 2010;

McLaughlin and Herlocker, 2004; Jambor and Wang, 2010b; Bellogín et al., 2011b), drawing from evaluation well studied methodologies in the Information Retrieval field.

Precision-oriented metrics measure the amount of relevant and non-relevant retrieved (recommended) items. A solid body of metrics, methodologies, and datasets has been developed over the years in the Information Retrieval field. Recommendation can be naturally stated as an information retrieval task: users have an implicit need with regards to a space of items which may serve the user's purpose, and the task of the recommender system is to select, rank and present the user a set of items that may best satisfy her need. The need of the user and the qualities or reasons why an item satisfies it cannot be observed in full, or described in an exact and complete way, which is the defining characteristic of an information retrieval problem, as opposed to data retrieval tasks or logical proof. It is thus natural to adapt relevance-based Information Retrieval evaluation methodologies here, which mainly consist of obtaining manual relevance labels of recommended items with respect to the user's need, and assessing, in different ways, the amount of relevant recommended items.

Recommendation tasks and the available data for their evaluation, nonetheless, have specific characteristics, which introduce particularities with respect to mainstream experience in the Information Retrieval field. In common information retrieval experimental practice, driven to a significant extent by the TREC campaigns (Voorhees and Harman, 2005), relevance knowledge is typically assumed to be (not far from) complete – mainly because in the presence of a search query, relevance is simplified to be a user-independent property. However, in recommender systems it is impractical to gather complete preference information for *each* user in a system. In datasets containing thousands of users and items, only a fraction of the items that users like is generally known. The unknown rest are, for evaluation purposes, assumed to be non-relevant. This is a source of – potentially strong – bias in the measurements depending on how unknown relevance is handled. In the next chapter we cover in detail these problems, along with an analysis of the different experimental design alternatives available in the literature.

In the reminder of this chapter we present some of the most common evaluation metrics. We classify them into error-based and precision-based metrics, accounting for the two tasks previously described – rating prediction and item ranking, respectively. After that, we describe the main methodologies used in the area to partition datasets and to select the candidate items in the latter task. Finally, we introduce the datasets used in this thesis to evaluate different recommendation algorithms.

3.2 Evaluation metrics

The evaluation of recommender systems should take into account the goal of the system itself (Herlocker et al., 2004). For example, in (Herlocker et al., 2004) the authors identify two main user tasks: *annotation in context* and *find good items*. In these tasks the users only care about errors in the item rank order provided by the system, not the predicted rating value itself. Based on this consideration, researchers have started to use precision-based metrics to evaluate recommendations, although most works also still report error-based metrics for comparison with state of the art approaches. Moreover, other authors, such as Herlocker and colleagues (Herlocker et al., 2004), encourage considering alternative performance criteria, like the novelty of the suggested items and the item coverage of a recommendation method. We describe the above types of evaluation metrics in the subsequent sections.

3.2.1 Error-based metrics

A classic assumption in the recommender systems literature is that a system that provides more accurate predictions will be preferred by the user (Shani and Gunawardana, 2011). Although this has been further studied and refuted by several authors (McNee et al., 2006; Cremonesi et al., 2011; Bollen et al., 2010), the issue is still worth being analysed.

Traditionally, the most popular metrics to measure the accuracy of a recommender system have been the **Mean Absolute Error** (MAE), and the **Root Mean Squared Error** (RMSE):

$$\mathrm{MAE} = \frac{1}{|\mathrm{Te}|} \sum_{(u,i)\in \mathrm{Te}} |\tilde{r}(u,i) - r(u,i)| \tag{3.1}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{|\mathrm{Te}|} \sum_{(u,i)\in \mathrm{Te}} \big(\tilde{r}(u,i) - r(u,i)\big)^2} \tag{3.2}$$

where $\tilde{r}$ and r denote the predicted and real rating, respectively, and Te corresponds to the test set. The RMSE metric is usually preferred to MAE because it penalises larger errors.

Different variations of these metrics have been proposed in the literature. Some authors **normalise MAE** and **RMSE** with respect to the maximum range of the ratings (Goldberg et al., 2001; Shani and Gunawardana, 2011) or with respect to the expected value if ratings are distributed uniformly (Marlin, 2003; Rennie and Srebro, 2005). Alternatively, **per-user** and **per-item average errors** have also been proposed in order to avoid biases from the error (or accuracy) on a few very frequent users or items (Massa and Avesani, 2007a; Shani and Gunawardana, 2011). For instance, the user-average MAE is computed as follows:

$$\text{uMAE} = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\frac{1}{|\text{Te}_u|}\sum_{i\in\text{Te}_u}|\tilde{r}(u,i) - r(u,i)| \tag{3.3}$$

A critical limitation of these metrics is that they do not make any distinction between the errors made on the top items predicted by a system, and the errors made for the rest of the items. Furthermore, they can only be applied when the recommender predicts a score in the allowed range of rating values. Because of that, log-based, and some content-based and probabilistic recommenders cannot be evaluated in this way, since $\tilde{r}(u,i)$ would represent a probability or, in general, a preference score. Hence, these methods can only be evaluated by measuring the performance of the generated ranking using precision-based metrics.

3.2.2 Precision-based metrics

These metrics can be classified into three groups: metrics that only use one ranking, metrics that compare two rankings (typically, one of them is a reference or ideal ranking), and metrics from the Machine Learning field.

Metrics based on one ranking

Examples of these metrics are precision, recall, normalised discounted cumulative gain, mean average precision, and mean reciprocal rank. Each of these metrics captures the quality of a ranking from a slightly different angle. More specifically, **precision** accounts for the fraction of recommended items that are relevant, whereas **recall** is the fraction of the relevant items that has been recommended. Both metrics are inversely related, since an improvement in recall typically produces a decrease in precision. They are typically computed up to a ranking position or cutoff k, being denoted as $P@k$ and $R@k$, and defined as follows (Baeza-Yates and Ribeiro-Neto, 2011):

$$P@k = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\frac{|\text{Rel}_u@k|}{k} \tag{3.4}$$

$$R@k = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\frac{|\text{Rel}_u@k|}{|\text{Rel}_u|} \tag{3.5}$$

where Rel_u represents the set of relevant items for user u, and $\text{Rel}_u@k$ is the number of relevant recommended items up to position k.

Recall has also been referred to as **hit-rate** in (Deshpande and Karypis, 2004). Hit-rate has also been defined as the percentage of users with at least one correct recommendation (Bellogín et al., 2012), corresponding to the **success** metric (or **first relevant score**), as defined by TREC (Tomlinson, 2005).

Furthermore, the **mean average precision** (MAP) metric provides a single summary of the user's ranking by averaging the precision figures obtained after each new relevant item is obtained, as follows (Baeza-Yates and Ribeiro-Neto, 2011):

$$\text{MAP} = \frac{1}{|\mathcal{U}|}\sum_{u}\frac{1}{|\text{Rel}_u|}\sum_{i\in\text{Rel}_u}\text{P@rank}(u,i) \tag{3.6}$$

where $\text{rank}(u,i)$ outputs the ranking position of item i in the user's u list; hence, precision is computed at the position where each relevant item has been recommended.

Normalised discounted cumulative gain (nDCG) uses graded relevance that is accumulated starting at the top of the ranking and may be reduced, or discounted, at lower ranks (Järvelin and Kekäläinen, 2002):

$$\text{nDCG} = \frac{1}{|\mathcal{U}|}\sum_{u}\frac{1}{\text{IDCG}_u^{p_u}}\sum_{p=1}^{p_u}\text{f}_{\text{dis}}\big(\text{rel}\big(u,i_p\big),p\big) \tag{3.7}$$

where the discount function $f_{\text{dis}}\big(rel\big(u,i_p\big),p\big)$ is usually defined as $f_{\text{dis}}(x,y) = (2^x-1)/\log(1+y)$ or simply $f_{\text{dis}}(x,y) = x/\log y$ if $y>1$, $f_{\text{dis}}(x,y)=x$ otherwise, depending on the emphasis required on retrieving highly relevant items (Croft et al., 2009). IDCG_u^k denotes the score obtained by an ideal or perfect ranking for user u up to position k, which acts as a normalisation factor in order to compare different users and datasets. Besides, p_u denotes the maximum number of items evaluated for each user; which is typically assumed to be a cutoff k, the same for all the users. In that situation, this metric is denoted as $\text{nDCG@}k$.

Using a different discount function, the **rank score** or **half-life utility** metric (Breese et al., 1998; Herlocker et al., 2004; Huang et al., 2006) can be obtained as follows:

$$\text{HL} = 100\left(\sum_{u}\text{HL}_u^{\max}\right)^{-1}\sum_{u}\text{HL}_u\,;\quad \text{HL}_u = \sum_{p=1}^{p_u}\frac{\max\big(\tilde{r}\big(u,i_p\big)-d,0\big)}{2^{(p-1)/(\alpha-1)}} \tag{3.8}$$

where d is the default ranking, and α is the half-life utility that represents the rank of the item on the list such that there is a 50% chance that the user will view that item. In (Breese et al., 1998) the authors use a value of 5 in their experiments, and note that they did not obtain different results with a half-life of 10.

Mean reciprocal rank (MRR) favours rankings whose first correct result occurs near the top ranking results (Baeza-Yates and Ribeiro-Neto, 2011). It is defined as follows:

$$\text{MRR} = \sum_{u}\frac{1}{s_r(u)} \tag{3.9}$$

where $s_r(u)$ is a function that returns the position of the first relevant item obtained for user u. This metric is similar to the **average rank of correct recommendation** (ARC) proposed in (Burke, 2004) and to the **average reciprocal hit-rank** (ARHR) defined in (Deshpande and Karypis, 2004).

It is important to note that since its early days, there has been a concern in the Information Retrieval field for the value and validity of the standard precision and recall metrics in interactive contexts (Su, 1992; Belkin and Croft, 1992). Nonetheless, precision-based metrics such as precision and recall, and more in general, metrics that measure the quality of the item ranking returned by a recommender have been frequently used in the field, despite they often lead to uncomparable results (Bellogín et al., 2011a).

Metrics based on two rankings

Additionally, specific metrics have been defined in the context of recommender evaluation that take as inputs two rankings (ideal vs estimated) instead of just one. A first example is the **normalised distance-based performance measure** (NDPM), used in (Balabanovic and Shoham, 1997), and proposed in (Yao, 1995). This metric compares two different weakly ordered rankings, and is formulated as follows (Herlocker et al., 2004; Shani and Gunawardana, 2011):

$$\text{NDPM} = \frac{1}{|\mathcal{U}|}\sum_{u}\frac{2C_u^{\text{con}} + C_u^{\text{tie}}}{2C_u} \tag{3.10}$$

where C_u is the number of pairs of items for which the real ranking (reference ranking using the ground truth) asserts an ordering, i.e., the items are not tied. Besides, C_u^{con} denotes the number of discordant item pairs between the method's ranking and the reference ranking, and C_u^{tie} represents the number of pairs where the reference ranking does not tie, but where the method's ranking does. This metric is comparable across datasets since it is normalised with respect to the worst possible scenario (denominator). Furthermore, it provides a perfect score of 0 to systems that correctly predict every preference relation asserted by the reference, and a worst score of 1 to methods that contradict every reference preference relation. Besides, a penalisation of 0.5 is applied when a reference preference relation is not predicted, whereas predicting unknown preferences (i.e., they are not ordered in the reference ranking) receives no penalisation.

As the previous metric, rank correlation metrics such as **Spearman**'s ρ and **Kendall**'s τ have also been proposed to directly compare the system ranking to a preference order given by the user. These correlation coefficients are later defined and analysed (Chapter 5). Here we only indicate that they provide scores in the range

of -1 to 1, where 1 denotes a perfect correlation between the two above rankings, and -1 represents an inverse correlation.

These two metrics, along with NDPM, suffer from the interchange weakness (Herlocker et al., 2004), that is, interchanges at the top of the ranking have the same weight that interchanges at the bottom of the ranking.

Metrics from Machine Learning

Finally, some other metrics from the Machine Learning literature have also been used, although they are not very popular. For instance, the **receiving operating characteristic** (ROC) curve and the **area under the curve** (AUC) have been used in (Herlocker et al., 1999), (Schein et al., 2001), (Schein et al., 2002), and (Rojsattarat and Soonthornphisaj, 2003), among others. Metrics based on the ROC curve provide a theoretically grounded alternative to precision and recall (Herlocker et al., 2004). The ROC model attempts to measure the extent to which an information filtering system can successfully distinguish between signal (relevant items) and noise. Starting from the origin of coordinates at (0,0), the ROC curve is built by considering, at each rank position, whether the item is relevant or not for the user; in the first case, the curve goes one step up, and in the second, one step right.

A random recommender is expected to produce a straight line from the origin to the upper right corner; on the other hand, the more leftwards the curve leans, the better is the performance of the system. These facts are related to the area under the ROC curve, a summary metric that is expected to be higher when the recommender performs better, where the expected value of a random recommender is 0.5, corresponding to a diagonal curve in the unit square.

In (Schein et al., 2001) the authors discriminate between the Global ROC (GROC) curve and the Customer ROC (CROC) curve, where the former assumes that only the most certain recommendations are made where some users may receive no recommendation at all; thus, the number of recommendations could be different for each user. The CROC curve is more realistic in the sense that every user receives the same amount of recommended items. However, for this curve a perfect recommender would not necessarily obtain an AUC of 1, and thus, it is required to compute the associated value of a perfect ROC curve in order to provide a fair comparison and normalise accordingly.

3.2.3 Other metrics

As different applications have different needs, additional characteristics of recommendations could be taken into consideration, and thus alternative metrics beyond accuracy and precision may be measured. In this context, it is important to understand and evaluate the possible trade-offs between these additional characteristics

and their effect on the overall recommendation performance (Shani and Gunawardana, 2011). For instance, some algorithms may provide recommendations with high quality or accuracy, but only for a small proportion of users or items, probably due to data sparsity. This effect can be quantified by measuring the **coverage** of the recommender system. Two types of coverage can be defined: *user coverage* (proportion of users to whom the system can recommend items) and *item or catalog coverage* (proportion of items the system can recommend). In (Shani and Gunawardana, 2011) two metrics are proposed for measuring item coverage: one based on the Gini's index, and another based on Shannon's entropy. In (Ge et al., 2010) the authors propose simple ratio quantities to measure such metrics, and to discriminate between the percentage of the items for which the system is able to generate a recommendation (*prediction coverage*), and the percentage of the available items that are effectively ever recommended (*catalog coverage*). A similar distinction is considered in (Herlocker et al., 2004) and (Salter and Antonopoulos, 2006). In (Herlocker et al., 2004) it is acknowledged that item coverage is particularly important for the tasks of *find all good items* and *annotation in context*. Besides, a system with low coverage is expected to be less valuable to users and the authors propose to combine coverage with accuracy measures to yield an overall "practical accuracy" measure for the system, in such a way that coverage is raised only because recommenders produce bogus predictions.

Beyond coverage, two recommendation characteristics have become very popular recently: **novelty** and **diversity**. Already a large amount of work has focused on defining metrics for measuring such characteristics (Lathia et al., 2010; Shani and Gunawardana, 2011; Vargas and Castells, 2011; Zhang and Hurley, 2009), and designing algorithms to provide novel and/or diverse recommendations (Jambor and Wang, 2010b; Onuma et al., 2009; Weng et al., 2007; Zhou et al., 2010).

Novel recommendations are those that suggest the user items she did not know about prior to the recommendation (Shani and Gunawardana, 2011), referred to as non-obvious items in (Herlocker et al., 2004; Zhang et al., 2002). Novelty can be directly measured in online experiments by directly asking users whether they are familiar with the recommended item (Celma and Herrera, 2008). However, it is also interesting to measure novelty in an offline experiment, so as not to restrict its evaluation to costly and hardly reproducible online experiments.

Novelty can be introduced into recommendations by using a topic taxonomy (Weng et al., 2007), where items containing novel topics are appreciated. Typically, novel topics are obtained by clustering the previously observed topics for each user. In (Lathia et al., 2010), novelty measures the amount of new items appearing in the recommended lists over time. In (Onuma et al., 2009) a technique based on graphs is introduced to suggest nodes (items) well connected to older choices, but at the same time well connected to unrelated choices.

Metrics based on Information Theoretic properties of the items being recommended have also been proposed by several authors. In (Bellogín et al., 2010) the entropy function is used to capture the novelty of a recommendation list, in (Zhou et al., 2010) the authors use the self-information of the user's top recommended items, and in (Filippone and Sanguinetti, 2010) the Kullback-Leibler divergence is used.

In Information Retrieval, diversity is seen as an issue of avoiding redundancy and finding results that cover different aspects of an information need (Radlinski et al., 2009). In that context, most of the proposed methods and metrics make use of (explicit or inferred) query aspects (topics or interpretations) to rank higher the most likely results (Demidova et al., 2010), or diversify a prior result set (Clarke et al., 2008; Agrawal et al., 2009; Chandar and Carterette, 2010; Radlinski et al., 2008; Rafiei et al., 2010).

In recommender systems diversity has been typically defined in an ad-hoc way, often mixing concepts such as diversity, novelty and coverage. For example, in (Salter and Antonopoulos, 2006) the authors make use of the catalog coverage defined above as a measure of recommendation diversity. A similar assumption is done in (Kwon, 2008). In (Zhou et al., 2010) the authors show that by tuning appropriately a hybrid recommender it is possible to obtain simultaneous gains in both accuracy and diversity, which is measured as the inter-list distance between every pair of users in the collection. Zhang and Hurley (2008) measure the novelty of an item by the amount of diversity it brings to the recommendation list, which is computed using a distance or dissimilarity function.

More formal definitions for diversity have also been proposed. In (Lathia et al., 2010) the authors propose to analyse diversity of top-N lists over time by comparing the intersection of sequential top-N lists. A statistical measure of diversity is proposed in (Zhang and Hurley, 2009), where the authors consider a recommendation algorithm to be fully diverse if it is equally likely to recommend any item that the user likes. In (Jambor and Wang, 2010b), the introduction of the covariance matrix into the optimisation problem leads to promote items in the long tail. A similar result is obtained in (Celma and Herrera, 2008), where the items with fewer interactions within the community of users (long tail) are assumed to be more likely to be unknown. Based on item similarities and focused on content-based algorithms, the authors in (Bradley and Smyth, 2001) propose a quality metric which considers both the diversity and similarity obtained in the recommendation list. A definition based on the entropy of the probability distributions of each recommender with respect to the items is proposed in (Bellogín et al., 2010), and the Gini's index is used in (Fleder and Hosanagar, 2009).

Finally, in (Vargas and Castells, 2011) a formal framework for the definition of novelty and diversity metrics is presented, where several previous metrics are unified

by identifying three ground concepts at the roots of novelty and diversity: choice, discovery, and relevance.

Other metrics such as serendipity, privacy, adaptivity, confidence, and scalability have been less explored in the literature, but their importance and application to recommender systems have already been discussed, making clear their relation with the user's experience and satisfaction, which is the ultimate goal of a "good" recommender system (Herlocker et al., 2004; McNee et al., 2006; Shani and Gunawardana, 2011).

3.3 Experimental setup

An important decision in the experimental configuration of a recommender evaluation is the dataset partition strategy. How the datasets are partitioned into training and test sets may have a considerable impact on the final performance results, and may cause some recommenders to obtain better or worse results depending on how this partition is configured. Although an exhaustive analysis of the different possibilities to choose the ratings/items to be hidden is out of the scope of this thesis, we briefly discuss now some of the most well-known methods used.

First, we have to choose whether or not to take time into account (Gunawardana and Shani, 2009). Time-based approaches naturally require the availability of user interaction data timestamps. A simple approach is to select a time point in the available interaction data timeline to separate training data (all interaction records prior to that point) and test data (dated after the split time point). The split point can be set so as to, for instance, have a desired training/test ratio in the experiment. The ratio can be global, with a single common split point for all users, or user-specific, to ensure the same ratio per user. Time-based approaches have the advantage of more realistically matching working application scenarios, where "future" user likes (which would translate to positive response to recommendations by the system) are to be predicted based on past evidence. As an example, the well-known Netflix prize provided a dataset where the test set for each user consisted on her most recent ratings (Bennett and Lanning, 2007).

If we ignore time, there are at least the following three strategies to select the items to hide from each user: a) sample a fixed number (different) for each user; b) sample a fixed (but the same for all) number for each user, also known as *given n* or *all but n* protocols; c) sample a percentage of all the interactions using *cross-validation*. The most usual protocol is the last one (Goldberg et al., 2001; Sarwar et al., 2001), although several authors have also used the *all but n* protocol (Breese et al., 1998; Wang et al., 2008a). The MovieLens datasets provide random splits following a five-fold cross validation strategy, as we shall see in the next section.

Nonetheless, independently from the dataset partition, it is recognised that the goals for which an evaluation is performed may be different in each situation, and thus, a different setting (and partition protocol) should be developed (Herlocker et al., 2004; Gunawardana and Shani, 2009). If that is not the case, the results obtained in a particular setting would be biased and difficult to use in further experiments, for instance, in an online experimentation.

Furthermore, as mentioned earlier, in order to evaluate ranked recommendations for a target user u, it is required to select two sets of items, namely relevant and not relevant. In the next chapter, we describe different possibilities explored in the literature, along with a detailed analysis of these alternatives and the possible biases that may appear.

3.4 Evaluation datasets

In this section we present three datasets that were used in the experimental parts of this thesis. The datasets correspond to different domains: movie recommendation, in which user preferences are provided in the form of ratings, and music recommendation, where user preferences are derived from implicit (log-based) evidence. Furthermore, one of the datasets includes social information that can be exploited by social filtering algorithms.

3.4.1 MovieLens dataset

The GroupLens research lab[1] has released different datasets obtained from user interaction in the MovieLens recommender system. At the time of writing, there are three publicly available MovieLens datasets of different sizes:

- The 100K dataset, containing 100,000 ratings for 1,682 movies by 963 users.
- The 1M dataset, with one million ratings has 6,040 users and 3,900 movies.
- The 10M dataset, with 10 million ratings consists of almost 71,600 users and 10,700 movies, and 100,000 tag assignments.

Although there are larger public datasets (such as the one provided for the well-known competition organised by Netflix[2] between 2006 and 2009), the first two MovieLens datasets are currently, by far, the most used in the field.

The ratings range on a 5-star scale in all three datasets; the 100K and 1M versions only use "integer" stars, and 10M uses "half star" precision (ten discrete rating values). Every user has at least 20 ratings in any of the datasets.

[1] GroupLens research lab, http://www.grouplens.org

[2] Netflix site, http://www.netflix, and Netflix Prize webpage, http://www.netflixprize.com

3.4.2 Last.fm dataset

Last.fm is a social music website. At the moment, the site has more than 40 million users (claimed 30 million active in 2009[3]) in more than 190 countries. Several authors have analysed and used this system for research purposes; special mention deserves those who have made their datasets public, such as (Konstas et al., 2009), (Celma, 2010), and (Cantador et al., 2011).

In 2010, Òscar Celma released two datasets collected using the Last.fm API. The first one (usually referred to as 360K) contains the number of plays (called *scrobblings* in the platform) of almost 360,000 users, counted on music artists, amounting to more than 17 million of (user, artist, playcounts) tuples. The second dataset (named 1K) contains fewer users (nearly 1,000) but, in contrast to the previous one, the whole listening history of each user is collected as tuples (user, timestamp, artist, music track) for up to 19 million tuples.

3.4.3 CAMRa dataset

In 2010 the 1st Challenge on Context-aware Movie Recommendation (CAMRa 2010[4]) was held at the 4th ACM conference on Recommender Systems (RecSys 2010). The challenge organisers released four datasets that were used in three different challenge tracks (Adomavicius et al., 2010). These tracks were focused on temporal recommendation (*weekly recommendation*), recommendation based on mood (*Moviepilot track*), and social recommendation (*Filmtipset track*). Two different datasets were provided for the first track, whereas the second and third tracks were assigned a different dataset each (Said et al., 2010).

These datasets were gathered from the Filmtipset[5] and Moviepilot[6] communities, and, depending on the track, contained social links between users, movie ratings, movie reviews, review ratings, comments about actors and movies, movie directors and writers, lists of favourite movies, moods, and links between similar movies. Filmtipset is the largest online social community in the movie domain in Sweden, with more than 90,000 registered users and 20 million ratings in its database. Moviepilot, on the other hand, is the leading online movie and TV recommendation community in Germany; it has over 100,000 registered users and a database of over 40,000 movies with roughly 7.5 million ratings (Said et al., 2010).

Further editions of this challenge have also released datasets related to recommendation tasks (focused on group recommendation in 2011 (Said et al., 2011) and

[3] Announcement, http://blog.last.fm/2009/03/24/lastfm-radio-announcement

[4] CAMRa site, http://2010.camrachallenge.com/

[5] Filmtipset site, http://www.filmtipset.se

[6] Moviepilot site, http://www.moviepilot.de

on additional context information in 2012). However, they were not used in this thesis, and thus, they are not described in detail here.

3.5 Summary

In the Recommender Systems literature several evaluation metrics, protocols, and methodologies have been defined. It remains unclear the equivalence between them and the extent to which they would provide comparable results.

The problem of evaluating recommender systems has been a major object of study and methodological research in the field since its earliest days. Error-based metrics have widely dominated the field, and precision-based metrics have started to be adopted more recently. Other metrics from the Machine Learning field have been proposed but they are not widely used in the community yet. Moreover, metrics for additional dimensions such as novelty or diversity have also started to be researched in the last few years.

There are, still, important characteristics of the evaluation methodologies and metrics that remain unexplored. In contrast to the Information Retrieval community, where statistical analysis and eventual biases in the evaluation as a whole have been studied (Buckley et al., 2006; Aslam et al., 2006; Soboroff, 2004), there is a lack of such an analysis for recommender systems. This raises a key issue for our research which shall be analysed in depth in the next chapter, where we propose some alternative methodologies to overcome some of the possible biases that may arise.

Chapter 4

Ranking-based evaluation of recommender systems: experimental designs and biases

There is an increasing consensus in the Recommender Systems community that the dominant error-based evaluation metrics are insufficient, and to some extent inadequate, to properly assess the practical effectiveness of recommendations. Seeking to evaluate recommendation rankings – which largely determine the effective accuracy in matching user needs – rather than predicted rating values, Information Retrieval metrics have started to be applied to evaluate recommender systems.

In this chapter we analyse the main issues and potential divergences in the application of Information Retrieval methodologies on recommender system evaluation, and provide a systematic characterisation of experimental design alternatives for this adaptation. We lay out an experimental configuration framework upon which we identify and analyse specific statistical biases arising in the adaptation of Information Retrieval metrics to recommendation tasks, which considerably distort the empirical measurements, hindering the interpretation and comparison of results across experiments. We propose two experimental design approaches that effectively neutralise such biases to a large extent. We support our findings and proposals through both analytical and empirical evidence.

We start the chapter by introducing the problem of (un)biased evaluation in recommender systems. The reminder of the chapter follows by revisiting the principles and assumptions underlying the Information Retrieval evaluation methodology: the Cranfield paradigm (Section 4.2). After that, in Section 4.3 we elaborate a formal synthesis of the main approaches to the application of Information Retrieval metrics to recommendation. In Sections 4.4 and 4.5 we analyse, respectively, the sparsity and popularity biases of Information Retrieval metrics on recommendation tasks. We present and evaluate two approaches to avoid these biases in Section 4.6,and end with some conclusions in Section 4.7.

4.1 Introduction

There seems to be a raising awareness in the Recommender Systems (RS) community that important – or even central – open questions remain to be addressed concerning the evaluation of recommender systems. As we mentioned in the previous chapter, the error in predicting held-out user ratings has been by far the dominant offline evaluation methodology in the RS literature (Breese et al., 1998; Herlocker et al., 2004). The limitations of this approach are increasingly evident, and have been extensively pointed out (Cremonesi et al., 2010). The prediction error has been found to be far from enough or even adequate to assess the practical effectiveness of a recommender system in matching user needs. The end users of recommendations receive lists of items rather than rating values, whereby recommendation accuracy metrics – as surrogates of the evaluated task – should target the quality of the item selection and ranking, rather than the numeric system scores that determine this selection.

For this reason, researchers are turning towards metrics and methodologies from the Information Retrieval (IR) field (Barbieri et al., 2011; Cremonesi et al., 2010; Herlocker et al., 2004), where ranking evaluation has been studied and standardised for decades. Yet, gaps remain between the methodological formalisation of tasks in both fields, which result in divergences in the adoption of IR methodologies, hindering the interpretation and comparability of empirical observations by different authors. The use of IR evaluation techniques involves the adoption of the Cranfield paradigm (Voorhees and Harman, 2005), and common metrics such as precision, mean average precision (MAP), and normalised Discounted Cumulative Gain (nDCG) (Baeza-Yates and Ribeiro-Neto, 2011). Given the natural fit of top-n recommendation in an IR task scheme, the adoption of IR methodologies would seem straightforward. However, recommendation tasks, settings, and available datasets for offline evaluation involve subtle differences with respect to the common IR settings and experimental assumptions, which result in substantial biases to the effectiveness measurements that may distort the empiric observations and hinder comparison across systems and experiments.

Furthermore, how to measure the performance of a recommender system is a key issue in our research. The variability in the experimental configurations, and the observed statistical biases of the evaluation methodologies should be well understood, since we aim to predict the performance of a system. We should avoid the situation where a metric shows some source of noise together with the recommender's quality, since then a predictor capturing only that noise would appear as an (equivocal) effective performance predictor.

Taking up from prior studies on the matter (Cremonesi et al., 2010; Herlocker et al., 2004; Shani and Gunawardana, 2011; Steck, 2011), we revisit the methodological assumptions underlying IR metrics, and analyse the differences between Recom-

mender Systems and Information Retrieval evaluation settings and their implications. Upon this, we identify two sources of bias in IR metrics on recommender systems: data sparsity and item popularity. We characterise and study the effect of these two factors both analytically and empirically. We show that the value range of common IR metrics is determined by the density of the available user preference information, to such an extent that the measured values per se are not meaningful, except for the purpose of comparison within a specific experiment. Furthermore, we show that the distribution of ratings among items has a drastic effect on how different algorithms compare to each other. Finally, we propose and analyse two approaches to mitigate popularity biases on the measured ranking quality, providing theoretical and empirical evidence of their effectiveness.

4.2 Cranfield paradigm for recommendation

Information Retrieval evaluation methodologies have been designed, studied, and refined over the years under the so-called *Cranfield paradigm* (van Rijsbergen, 1989; Voorhees, 2002b). In the Cranfield paradigm, as e.g. typically applied in the TREC campaigns (Voorhees and Harman, 2005), information retrieval systems are evaluated on a dataset comprising a set of documents, a set of queries – referred to as *topics* and consisting of a description or representation of user information needs –, and a set of relevance judgments by human assessors – referred to as *ground truth*. The assessors manually inspect queries and documents, and decide whether each document is relevant or not for a query. Theoretically, each query-document pair should be assessed for relevance, which, for thousands or millions of documents, is obviously unfeasible. Therefore, a so-called *pooling* approximation is applied, in which the assessors actually inspect and judge just a subset of the document collection, consisting of the union of the top-n documents returned by a set of systems for each query. These systems for pooling are commonly the ones to be evaluated and compared, and n is called the *pooling depth*, typically ranging around 100 documents. While this procedure obviously misses some relevant documents, it has been observed that the degree of incompleteness is reasonably small, and the missing relevance does not alter the empiric observations significantly, at least up to some ratio between the pooling depth and the collection size (Buckley et al., 2007).

Whereas in a search system users may enter multiple queries, the recommendation task – in its classic formulation – typically considers a single "user need" per user, that is, a user has a set of cohesive preferences which defines her main interests. In this view a natural fit of recommendation in the Cranfield paradigm would take users – as an abstract construct – as the equivalent of queries in ad-hoc retrieval (the user need to be satisfied), and items as equivalent to documents (the objects to be retrieved and ranked), summarised in Table 4.1. A first obvious difference is that

Task element	TREC ad-hoc retrieval task	Recommendation task
Information need expression	Topic (query and description)	User profile
Candidate answers	All documents in the collection	Target item set
	Same for all queries	One or more per user, commonly different
Document data available as system input	Document content	Training ratings, item features
Relevance	Topical, objective	Personalised, subjective
Ground truth	Relevance judgments	Test ratings
Relevance assessment	Editorial assessors	End users
Relevance knowledge coverage	Reasonably complete (pooling)	Highly incomplete (inherently to task)

Table 4.1. Fitting the recommendation task in the Cranfield IR evaluation paradigm

queries are explicit representations of specific information needs, whereas in the recommendation setting, user profile records are a global and implicit representation of what the user may need or like. Still, the query-user mapping is valid, inasmuch as user profiles may rightfully fit in the IR scheme as "vague queries."

The definition of ground truth is less straightforward. User ratings for items, as available in common recommendation datasets, are indeed relevance judgments of items for user needs. However, many recommendation algorithms (chiefly, collaborative filtering methods) require these "relevance judgments" as input to compute recommendations. The rating data withholding evaluation approach, pervasive in RS research, naturally fits here: some test ratings can be held out as ground truth and the rest be left as training input for the systems. Differently from TREC, here the "queries" and the relevance assessments are both entered by the same people: the end-users. Furthermore, how much data are taken for training and for ground truth is left open to the experiment designers, thus adding a free variable to be watched over as it significantly impacts the measurements.

On the other hand, whereas in the IR setting all the documents in the collection are candidate answers for all queries, the set of target items on which recommender systems are tested for each user need not be necessarily the same. As already described in the previous chapter, in general, the items with a test rating are included in the candidate set for the raters, though not necessarily in a single run (Cremonesi et al., 2010). Moreover, it is common to select further non-rated target items, but not necessarily all the items (Bellogín et al., 2011a). Furthermore, the items rated by a user in the training set are generally excluded from the recommendation to this user. The way these options are configured has a drastic effect on the resulting measurements, with variations in orders of magnitude (Bellogín et al., 2011a).

In addition to this, the coverage of user ratings is inherently much smaller in recommender systems' datasets compared to TREC collections. The amount of un-

known relevance – which in TREC is assumed to be negligible – is pervasive in recommendation settings (it is in fact intrinsic for the task to make sense), to a point where some assumptions of the IR methodology may not hold, and the gap between measured and real metric values becomes so significant that a metric's absolute magnitude may just lose any meaning. Still, such measurements may support comparative assessments between systems, as far as the bias is system-independent.

Finally, the distribution of relevance in the retrieval space displays popularity patterns that are absent in IR datasets. The number of users who like each item is very variable (typically long-tailed) in recommendation datasets, whereas in TREC collections very few documents are relevant for more than one query. We shall show that this phenomenon has a very strong effect not only on metric values, but more importantly on how systems compare to each other.

In order to provide a formal basis for our study we start by elaborating a systematic characterisation of design alternatives for the adaptation of IR metrics to recommender systems, taking into account prior approaches described in the literature, such as those presented in the previous chapter. This formal framework will help us to analyse and describe the measurement biases in the application of IR metrics to recommender systems, and study new approaches to mitigate them.

4.3 Experimental design alternatives

The application of Information Retrieval metrics to recommender systems evaluation has been studied by several authors in the field (Barbieri et al., 2011; Breese et al., 1998; Cremonesi et al., 2010; Herlocker et al., 2004; Shani and Gunawardana, 2011). We elaborate here an experimental design framework that aims to synthesise commonalities and differences between studies, encompassing prior approaches and supporting new variants upon a common methodological grounding. We formalise the different methodologies presented in the previous chapter, and provide an equivalence between both formulations.

In the following, given a rating set split into training and test rating sets, we say an item $i \in \mathcal{I}$ is relevant for a user $u \in \mathcal{U}$ if u rated i positively, and its corresponding rating falls in the test set. By positive rating we mean a value above some design-dependent threshold. All other items (non-positively rated or non-rated) are considered as non-relevant. Like in the previous chapter, recommender systems are requested to rank a set of target items T_u for each user. Such sets do not need to be the same for each user, and can be formed in different ways. In all configurations T_u contains a combination of relevant and non-relevant items, and the different approaches are characterised by how these are selected, as we describe next.

Design settings		Alternatives
Base candidate items		**AI** $\mathcal{C} = \mathcal{I}$
		TI $\mathcal{C} = \bigcup_{u \in \mathcal{U}} \mathrm{R}_{\mathrm{test}}(u)$
Item selection	Relevant	**AR** $\mathrm{T}_u \supset \mathrm{PR}_{\mathrm{test}}(u)$
		1R $\lvert \mathrm{T}_u^{\mathrm{r}} \cap \mathrm{PR}_{\mathrm{test}}(u) \rvert = 1$
	Non-relevant	**AN** $\mathrm{N}_u = \mathcal{C} - \mathrm{PR}_{\mathrm{test}}(u) - \mathrm{R}_{\mathrm{train}}(u)$
		NN Fixed $\lvert \mathrm{N}_u \rvert$, random sampling

Table 4.2. Design alternatives in target item set formation.

4.3.1 Target Item Sampling

We identify three significant design axes in the formation of the target item sets: candidate item selection, relevant item selection, and irrelevant item sampling. We consider two relevant alternatives for each of these axes, summarised in Table 4.2, which we describe next.

We shall use $\mathrm{R}(u)$ and $\mathrm{PR}(u)$ to denote the set of all and positively rated items by user u, respectively, and $\mathrm{r}(u)$, $\mathrm{pr}(u)$ to denote the respective size of those sets. With the subscripts "test" and "train" we shall denote the part of such sets (or their sizes) contained on the corresponding side of a data split. An equivalent notation $\mathrm{r}(i)$, $pr(i)$, and so on, will be used for the ratings of an item, and when no user or item is indicated, the total number of ratings is denoted. This notation and the rest to be used along the chapter are summarised in Table 4.3.

Let $\mathrm{N}_u = \mathrm{T}_u - \mathrm{PR}_{\mathrm{test}}(u)$ be the non-relevant target items for u. As a general rule, we assume non-relevant items are randomly sampled from a subset of candidate items $\mathcal{C} \subset \mathcal{I}$, the choice of which is a design option. We mainly find two significant alternatives for this choice: $\mathcal{C} = \mathcal{I}$ (e.g. (Shani and Gunawardana, 2011)) and $\mathcal{C} = \bigcup_{u \in \mathcal{U}} \mathrm{R}_{\mathrm{test}}(u)$ (e.g. (Bellogín et al., 2011a; Vargas and Castells, 2011)). The first one, which we denote as **AI** for "all items", matches the typical IR evaluation setting, where the evaluated systems take the whole collection as the candidate answers. The second, to which we shall refer as **TI** ("test items") is an advisable condition to avoid certain biases in the evaluation of RS, as we shall see.

Once $\mathcal{C}$ is set, for each user we select a set $\mathrm{N}_u \subset \mathcal{C} - \mathrm{PR}_{\mathrm{test}}(u) - \mathrm{R}_{\mathrm{train}}(u)$. N_u can be sampled randomly for a fixed size $\lvert \mathrm{N}_u \rvert$ (we call this option **NN** for "N non-relevant"), or all candidate items can be included in the target set, $\mathrm{N}_u = \mathcal{C} - \mathrm{PR}_{\mathrm{test}}(u) - \mathrm{R}_{\mathrm{train}}(u)$ (we refer to this as **AN** for "all non-relevant"). Some authors have even used $\mathrm{T}_u = \mathrm{R}_{\mathrm{test}}(u)$ (Basu et al., 1998; Jambor and Wang, 2010a; Jambor and Wang, 2010b), but we discard this option as it results in a highly overestimated precision (Bellogín et al., 2011a). The size of N_u is thus a configuration parameter of

the experimental design. For instance, in (Cremonesi et al., 2010) the authors propose $|\mathrm{N}_u| = 1{,}000$, whereas in (Bellogín et al., 2011a) the authors consider $\mathrm{N}_u = \bigcup_{v \in \mathcal{U}} \mathrm{R}_{\mathrm{test}}(v) - \mathrm{R}_{\mathrm{train}}(u) - \mathrm{PR}_{\mathrm{test}}(u)$, among other alternatives. To the best of our knowledge, the criteria for setting this parameter have not been analysed in detail in the literature, leaving it to common sense and/or trial and error. It is worth noting nonetheless that in general $|\mathrm{N}_u|$ determines the number of calls to the recommendation algorithms, whereby this parameter provides a handle for adjustment of the cost of the experiments.

Regarding the relevant item selection, two main options are reported in the literature, to which we shall refer as **AR** for "all relevant", and **1R** for "one relevant." In the AR approach all relevant items are included in the target set, i.e., $T_u \supset PR_{test}(u)$ (Bellogín et al., 2011a). In the 1R approach, for user u, several target item sets T_u^r are formed, each including a single relevant item (Cremonesi et al., 2010). This approach may be more sensitive to the lack of recommendation coverage, as we shall observe later on. The choice between an AR or a 1R design involves a difference in the way the ranking quality metrics are computed, as we shall discuss in the next section.

Symbol			Meaning
$\mathcal{U}$	$\mathcal{I}$	$\mathcal{C}$	Set of all users | all items | candidate items
r	r_{test}	r_{train}	Nr. of all | test | training ratings
pr	pr_{test}	pr_{train}	Nr. of all | test | training positive ratings
$R(u)$	$R_{test}(u)$	$R_{train}(u)$	Set of all | test | training items rated by u
$PR(u)$	$PR_{test}(u)$	$PR_{train}(u)$	Set of all | test | training items liked by u
$r(u)$	$pr(u)$	…	Nr. of items rated | liked | … by u
$r(i)$	$pr(i)$	…	Nr. of users who rated | like | … item i
T_u	T_u^r		Set of target items for u in AR | 1R
N_u	N_u^r		Non-relevant items added to build T_u | $\mathrm{T}_u^{\mathrm{r}}$
$\mathrm{P}_s^u@n(T_u)$			P@n of item set T_u as ranked by s for u
$top_s^u(T_u, n)$			Top n items in T_u as ranked by s for u
$\tau_s^u(i, S)$			Position of i in S ∋ i as ranked by s for u
$i_k^{u,s}$	$i_k^{u,r,s}$		The item ranked k-th in T_u | $\mathrm{T}_u^{\mathrm{r}}$ by s for u
σ			Split ratio: $\mathrm{r}_{\mathrm{test}}/\mathrm{r}$
ρ_u	ρ		$\rho_u = \lvert \mathrm{T}_u \cap \mathrm{PR}_{\mathrm{test}}(u) \rvert / \lvert \mathrm{T}_u \rvert$, $\rho = \mathrm{avg}_u\, \rho_u$
t			"Average" target set size: $1/\mathrm{avg}_{\mathrm{u}}(1/\lvert \mathrm{T}_u \rvert)$
δ			Relevance density in target sets: $\mathrm{pr}/(\mathrm{t}\lvert\mathcal{U}\rvert)$

Table 4.3. Notation summary.

4.3.2 AR vs. 1R Precision

Essentially, the way metrics are defined in AR and 1R differs in how they are averaged. In AR the metrics are computed on each target set T_u in the standard way as in IR, and then averaged over users (as if they were queries). As a representative and simple to analyse metric, we shall use $\mathrm{P}@n$ henceforth, but similar properties to all the ones discussed here are observed for other metrics such as MAP and nDCG. The mean AR precision of a recommender system s can be expressed as:

$$P_s@n = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\frac{1}{n}|top_s^u(T_u,n)\cap PR_{test}(u)|$$

where $top_s^u(T_u,n)$ denotes the top n items in T_u ranked by s for u.

In the 1R design, drawing from (Cremonesi et al., 2010), we compute and average the metrics over the T_u^r sets, as follows:

$$1RP_s@n = precision(n) = \frac{1}{pr_{test}}\sum_{u\in\mathcal{U}}\sum_{r=1}^{pr_{test}(u)} P_s^u@n(T_u^r) \qquad (4.1)$$

where $\mathrm{P}_s^u@n(T_u^r)$ is the standard precision of T_u^r for u. This form to express the metric is equivalent to the original formulation in (Cremonesi et al., 2010), but lets a straightforward generalisation to any other IR metric such as MAP and nDCG, by just using them in place of $\mathrm{P}_s^u@n$ in Equation (4.1). We shall intentionally use the same symbol P to refer both to 1R and AR precisions when there is no ambiguity. Whenever there may be confusion, or we wish to stress the distinction, we shall use 1RP to explicitly denote 1R precision.

AR precision basically corresponds to the standard precision as defined in IR, whereas 1R precision, while following essentially the same principle, departs from it in the formation of runs, and the way to average values. Additionally, note that the maximum value of 1RP@n is 1/n as we shall see in the next section, mainly since each run has only one relevant item. Besides, in Section 4.4 we shall establish a formal relation between both ways to compute precision.

4.3.3 Preliminary Test

In order to illustrate the effects of the different described alternatives, we show their results on three common collaborative filtering algorithms, based respectively on probabilistic Latent Semantic Analisys (pLSA) (Hofmann, 2004), matrix factorisation (MF) (Koren et al., 2009), and user-based nearest-neighbours (kNN) (Cremonesi et al., 2010). As additional baselines, we include recommendation by popularity and random recommendation. We use two datasets: the 1M version of MovieLens, and

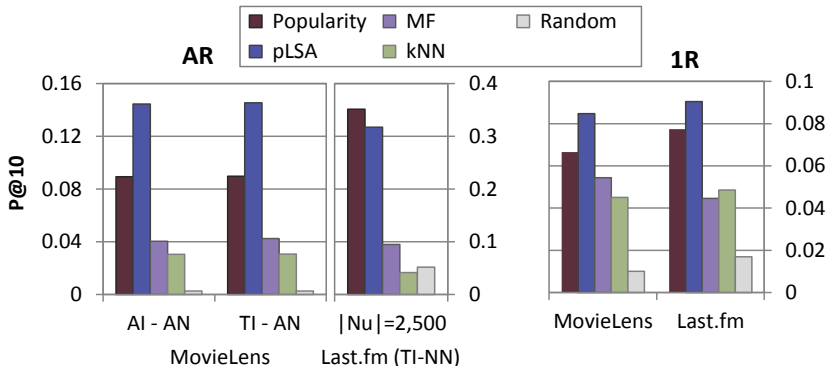


Figure 4.1. Precision of different recommendation algorithms on MovieLens 1M and Last.fm using AR and 1R configurations.

an extract from Last.fm published by Ò. Celma (Celma and Herrera, 2008). Details about the implementation and datasets partition are provided in Appendix A.

Figure 4.1 shows the P@10 results with AR and 1R configurations. For 1R we shall always use TI-NN, with $|T_u|$ = 100. This is a significantly lower value than $|T_u|$ = 1,001 reported in (Cremonesi et al., 2010), but we have found it sufficient to ensure statistical significance (e.g. Wilcoxon $p \ll 0.001$ for all pairwise differences between the recommenders in Figure 4.1), at a considerably reduced execution cost. We adopt the TI policy in 1R to avoid biases that we shall describe later. In the AR configuration we show TI-AN and AI-AN for MovieLens, though we shall generally stick to TI-AN in the rest of the chapter. In Last.fm we use only TI-NN and a temporal split, with $|N_u|$ = 2,500 for efficiency reasons, since $|\mathcal{I}|$ = 176,948 is considerably large in this dataset. We set the positive relevance rating threshold to 5 in MovieLens, as in (Cremonesi et al., 2010), whereas in Last.fm, we take any number above 2 playcounts as a sign of positive preference. We have experimented with other thresholds for positive ratings, obtaining equivalent results to all the ones that are reported here – the only difference is discussed in Section 4.6.

It can be seen that pLSA consistently performs best in most experimental configurations, closely followed by popularity, which is the best approach in Last.fm with AR, and that MF is generally superior to kNN. Some aspects strike our attention. First, even though P@10 is supposed to measure the same thing in all cases, the range of the metric varies considerably across configurations and datasets, and even the comparison is not always consistent. For instance, in AR popularity ranges from 0.08 on MovieLens to 0.35 on Last.fm; and AR vs. 1R produces some disagreeing comparisons on Last.fm. It may also be surprising that popularity, a non-personalised method, fares so well compared to other algorithms. This effect was already found recently in (Cremonesi et al., 2010) and (Steck, 2011). We also see that TI and AI produce almost the same results. This is because $\bigcup_{u \in \mathcal{U}} R_{test}(u) \sim \mathcal{I}$ in MovieLens; differences become noticeable in configurations where $\bigcup_{u \in \mathcal{U}} R_{test}(u)$ is significantly

smaller than $\mathcal{I}$, as we shall see in Section 4.6.2. As mentioned before, note that in this case, the upper bound of P@10 for the 1R methodology is 0.10.

Some of this variability may reflect actual strengths and weaknesses of the algorithms for different datasets, but we shall show that a significant part of the observed variations is due to statistical biases arising in the adaptation of the Cranfield methodology to recommendation data, and are therefore meaningless with respect to the assessment of the recommenders' accuracy. Specifically, we have found that the metrics are strongly biased to test data sparsity and item popularity. We shall analyse this in detail in Sections 4.4 and 4.5, but before that we establish a relation between AR and 1R precision that will help in this analysis.

4.3.4 Relation between AR and 1R

We have seen that AR and 1R precisions produce in general quite different values, and we shall show they display different dependencies over certain factors. We find nonetheless a direct relation between the two metrics. Specifically, 1R precision is bound linearly by NN-AR precision, that is, $1\mathrm{RP}_s@n = \Theta(\mathrm{P}_s@n)$, as we show next.

Lemma. Let us assume the irrelevant item sampling in 1R is done only once for all the test ratings of a user, that is, we select the same set of non-relevant items $N_u^r = N_u$ in the T_u^r target sets. If we denote $T_u = N_u \cup PR_{test}(u)$ – in other words, $T_u = \bigcup_r T_u^r$ –, we have:

$$\frac{|\mathcal{U}|P_s@n}{pr_{test}} \leq 1RP_s@n \leq \frac{\sum_{u\in\mathcal{U}} m_u P_s^u@m_u(T_u)}{n \cdot pr_{test}} \tag{4.2}$$

with $m_u = n + pr_{test}(u) - 1$, where $\mathrm{P}_s@n$ is the NN-AR precision computed with the target sets $\{T_u\}$.

Proof. Let i_u^r be the relevant item included in T_u^r, and let $\tau_s^u(i, S)$ denote the ranking position assigned to i by s for u within a set S, where $i \in S$. Since $T_u^r \subset T_u$, we have that $\tau_s^u(i_u^r, T_u^r) \leq \tau_s^u(i_u^r, T_u)$. This means that if i_u^r is ranked above n in T_u, then it is also above n in its target set T_u^r. Hence $\sum_{r=1}^{np_{test}(u)} |top_s^u(T_u^r, n) \cap PR_{test}(u)| \geq |top_s^u(T_u, n) \cap PR_{test}(u)|$. Summing on u, and dividing by n and pr_{test} we prove the first inequality of Equation (4.2).

On the other hand, it is easy to see that $\tau_s^u(i_u^r, T_u^k) \geq \tau_s^u(i_u^r, T_u) + pr_{test}(u) - 1$. Thus, if i_u^r is ranked above n in T_u^k, then it is above $m_u = n + pr_{test}(u) - 1$ in T_u. Thus $\sum_{r=1}^{np_{test}(u)} |top_s^u(T_u^r, n) \cap PR_{test}(u)| \leq |top_s^u(T_u, m_u) \cap PR_{test}(u)| = m_u \mathrm{P}_s@m_u(T_u)$. And the second inequality of Equation (4.2) follows again by summing on u, and dividing by n and pr_{test}. □

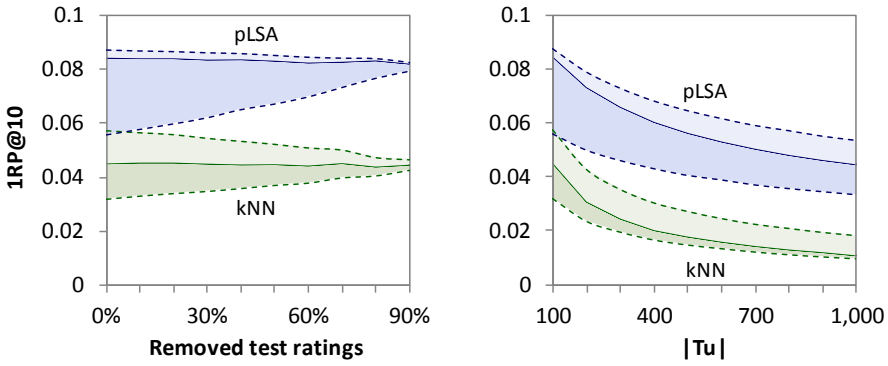


Figure 4.2. Empiric illustration of Equation (4.2). The curves show 1RP@10 and its bounds, for pLSA and kNN over MovieLens 1M. The light and dark shades mark the distance to the upper and lower bounds, respectively. The left side shows the evolution when progressively removing test ratings, and the right side displays the variation with $|T_u|$ ranging from 100 to 1,000.

Note that the assumption $N_u^r = N_u$ in the lemma is mild, inasmuch as the statistical advantage in taking different N_u^r for each r is unclear. Even in that case, $\mathrm{P}_s@n$ and $\mathrm{avg}_{u \in \mathcal{U}}\left(m_u \mathrm{P}_s^u @ m_u(T_u)\right)$ should be reasonably stable with respect to the random sampling of N_u^r, and thus Equation (4.2) tends to hold. Figure 4.2 illustrates the relation between the AR bounds and the 1R values. The empiric observation suggests they provide similar while not fully redundant assessments. We also see that the bounding interval reduces progressively as $|T_u|$ is increased (right), and even faster with test data sparsity (left) – in sum, the metric converges to its bounds as $|T_u| \gg \mathrm{avg}_u\, pr_{test}(u) = pr_{test}/|\mathcal{U}|$.

4.3.5 Limitations of error-based metrics

The analysis presented in (Bellogín et al., 2011a) leads to question again the suitability of error metrics. As in (McLaughlin and Herlocker, 2004), we found that there is no direct equivalence between results with error- and precision-based metrics. Common sense suggests that putting more relevant items in the top-N is more important for real recommendation effectiveness than being accurate with predicted rating values, which are usually not even shown to real users. Our study confirms that measured results differ between these two perspectives. An online experiment, where real users' feedback is contrasted to the theoretic measurements, may shed further light for an objective assessment and finer analysis of which methodology better captures user satisfaction.

Furthermore, the use of error-based metrics may not be applicable depending on the dataset or the recommender evaluated. For instance, log-based datasets and probabilistic (e.g. pLSA) or popularity-based recommenders cannot be evaluated using error-based metrics because no real ratings are available in the first case, and in

the second case because such recommenders do not necessarily predict a rating, not even a score in the range of ratings (Cremonesi et al., 2010).

The application of ranking-based metrics to recommendation, nonetheless, is far from being trivial. Firstly, there are obvious differences between the Cranfield paradigm and a standard recommendation context, as described in Section 4.2. Secondly, the evaluation methodology may be sensitive to any statistical bias which may appear in the process. In the next sections we shall analyse two of these sources of bias: sparsity and popularity.

4.4 Sparsity bias

As mentioned earlier, we identify two strong biases in precision metrics when applied to recommendation. The first one is a sensitivity to the ratio of the test ratings vs. the added non-relevant items. We study this effect by an analysis of the expected precision for non-personalised and random recommendations in the AR and 1R settings.

4.4.1 Expected Precision

Let $i_k^{u,s} \in T_u$ be the item ranked at position k in the recommendation output for u by a recommender system s, and let σ be the ratio of test data in the training-test data split. In an AR setup the expected precision at n (over the sampling space of data splits with ratio σ, the sampling of N_u, and any potential non-determinstic aspect of the recommender system – as e.g. in a random recommender) is:

$$E[P_s@n] = \underset{u \in \mathcal{U}}{avg}\left(\frac{1}{n}\sum_{k=1}^{n} p\left(rel \middle| i_k^{u,s}, u, T_u\right)\right)$$

where $p(rel|i,u)$ denotes the probability that item i is relevant for user u, i.e., the probability that $i \in PR_{test}(u)$. Now we may write $p\left(rel \middle| i_k^{u,s}, u, T_u\right) \overset{\text{def}}{=} p\left(rel, T_u \middle| i_k^{s,u}, u\right) / p\left(T_u \middle| i_k^{s,u}, u\right)$, where we have $p\left(T_u \middle| i_k^{s,u}, u\right) = p\left(i_k^{s,u} \in T_u\right) = |T_u|/|\mathcal{C}|$. On the other hand, $p\left(rel, T_u \middle| i_k^{s,u}, u\right) \overset{\text{def}}{=} p\left(i_k^{s,u} \in PR_{test}(u) \cap T_u\right) = p\left(i_k^{s,u} \in PR_{test}(u)\right) = p\left(rel \middle| i_k^{s,u}, u\right)$, since $PR_{test}(u) \subset T_u$ in the AR methodology. If s is a non-personalised recommender then $i_k^{u,s}$ and u are mutually independent, and it can be seen that $\text{avg}_{u \in \mathcal{U}}\, p\left(rel \middle| i_k^{u,s}, u\right) = \text{avg}_{u \in \mathcal{U}}\, p\left(rel \middle| i_k^{u,s}\right)$. All this gives:

$$E[P_s@n] = \frac{|\mathcal{C}|}{n \cdot t} \sum_{k=1}^{n} \underset{u \in \mathcal{U}}{avg}\, p\left(rel \middle| i_k^{u,s}\right)$$

where $1/t = \text{avg}_u(1/|T_u|)$ – if T_u have all the same size, then $t = |T_u|$. As all relevant items for each user are included in her target set, we have $p\left(rel\middle|i_k^{u,s}\right) = \text{E}\left[pr_{test}\left(i_k^{u,s}\right)\right]/|\mathcal{U}|$. If ratings are split at random into test and training, this is equal to $\sigma \cdot pr\left(i_k^{u,s}\right)/|\mathcal{U}|$. Hence, we have:

$$E[P_s@n] = \frac{\sigma|\mathcal{C}|}{n \cdot t\ |\mathcal{U}|}\sum_{k=1}^{n} \underset{u\in\mathcal{U}}{avg}\ pr\left(i_k^{u,s}\right) \tag{4.3}$$

Now, if items were recommended at random, we would have $\text{E}\left[pr\left(i_k^{u,RND}\right)\right] = pr/|\mathcal{I}|$, and therefore:

$$E[P_{RND}@n] = E[P_{RND}] \sim \frac{\sigma \cdot pr}{t\ |\mathcal{U}|} = \sigma \cdot \delta \tag{4.4}$$

where δ is the average density of known relevance – which depends on how many preferences for items the users have conveyed, and the size of the target test item sets.

On the other hand, in a 1R evaluation setup, we have:

$$E[1RP_s@n] = \frac{1}{n \cdot pr_{test}}\sum_{u\in\mathcal{U}}\sum_{r=1}^{pr_{test}(u)}\sum_{k=1}^{n} p\left(rel\middle|i_k^{u,r,s}, u, T_u^r\right)$$

where $i_k^{u,r,s} \in T_u^r$ denotes the item ranked at position k in T_u^r. For random recommendation, we have $p\left(rel\middle|i_k^{u,r,RND}, u, T_u^r\right) = 1/|T_u^r| = 1/t$ since all target sets have the same size, whereby we have:

$$E[1RP_{RND}@n] = E[1RP_{RND}] = 1/t \tag{4.5}$$

4.4.2 Test Sparsity Bias

The above results for the expected random precision provide a formal insight on strong metric biases to characteristics of the data and the experimental configuration. In both Equations (4.4) for AR and (4.5) for 1R, we may express the expected random precision as $\text{E}[\text{P}_{RND}@n] = \text{avg}_u\,\rho_u = \rho$, where ρ_u is the ratio of positively rated items by u in T_u (or T_u^r, for that matter), and $\rho \sim \sigma \cdot \delta$, or $\rho = 1/t$, depending on the experimental approach. In the AR approach the density δ, and thus the ρ ratio, are also inversely proportional to t. Precision in this methodology is therefore sensitive to (grows linearly with) σ and pr, and is inversely proportional to t, whereas 1R is only sensitive (inversely proportional) to t. The expected precision of random recommendation naturally provides a lower bound for any acceptable recommender. Note that in any configuration of AR and 1R, the total precision of any system is $\text{P}_s = \text{P}_{RND} = \rho = \text{E}[\text{P}_{RND}@n]$, since as all systems are required to return

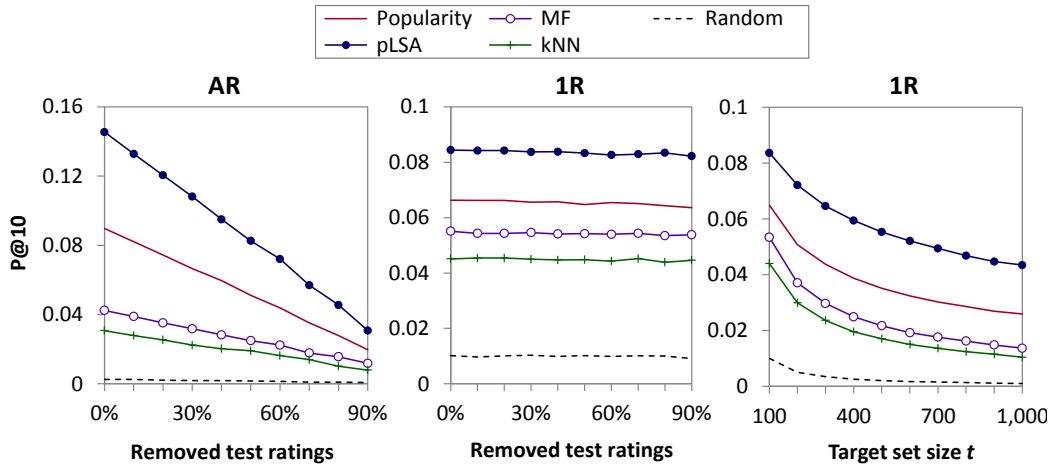


Figure 4.3. Evolution of the precision of different recommendation algorithms on MovieLens 1M, for different degrees of test sparsity. The x axis of the left and center graphics shows different amounts of removed test ratings. The x axis in the right graphic is the size of the target item sets.

(recommend) all items in the target sets T_u (or T_u^r), that is, the total precision does not depend on the ranking. At lower cutoffs, we expect to have $\mathrm{P}_s@n > \mathrm{E}[\mathrm{P}_{RND}@n] = \rho$. In other words, the lower bound – and so the expected range – for the $\mathrm{P}@n$ of recommender algorithms grows with the average ratio of relevant items per target item set.

The ρ ratio – hence the random precision – thus depends on several aspects of the experimental setup (the experimental approach, the split ratio σ, the number of non-relevant items in the target sets), and the test collection (the number of ratings, the number of users). Therefore, since ρ and the random precision can be adjusted arbitrarily by how the test sets are split, constructed, etc., we may conclude that **the specific value of the metric has a use for comparative purposes, but has no particular meaning by itself, unless accompanied by the corresponding average relevance ratio** ρ of the target test sets. This is naturally in high contrast to common IR datasets, where both the document collection and the relevance information are fixed and not split or broken down into subsets. In fact, the metric values reported in the TREC campaigns have stayed within a roughly stable range over the years (Armstrong et al., 2009a; Armstrong et al., 2009b). Note also that the sparsity bias we analyse here is different from the impact of training data sparsity in the performance of collaborative filtering systems. What we describe is a statistical bias caused by the sparsity of test data (as a function of overall data sparsity and/or test data sampling), and its effect does not reflect any actual variation whatsoever in the true recommendation accuracy.

The sparsity bias explains the precision range variations observed earlier in Figure 4.1. The empirically obtained values of random precision match quite exactly the

theoretically expected ones. To what extent the random recommendation analysis generalises to other algorithms can be further analysed empirically. Figure 4.3 illustrates the bias trends over rating density and target set size, using the experimental setup of Section 4.3.3 (with TI-AN in AR, and TI-NN in 1R). We show only the results in MovieLens – they display a similar effect on Last.fm. In the left and center graphics, we simulate test sparsity by removing test ratings. In the right graphic we vary $t = |T_u|$ in a 1R configuration. We observe that the empirical trends confirm the theoretical analysis: precision decreases linearly with density in the AR methodology (left graphic, confirming a linear dependence on δ), whereas precision is independent from the amount of test ratings in the 1R approach (center), and shows inverse proportionality to t (right). It can furthermore be seen that the biased behavior analytically described for random recommendation is very similarly displayed by the other recommenders (only differing in linear constants). This would confirm the explanatory power of the statistical trend analysis of random recommendation, as a good reference for similar biases in other recommenders. On the other hand, even though the precision values change drastically in magnitude, it would seem that the comparison between recommenders is not distorted by test sparsity. We find other biases in precision measurements, however, which do affect the comparison of recommenders, as we study in the next section.

4.5 Popularity bias

Sparsity is not the only bias the metric measurements are affected by. The high observed values for a non-personalised method such as recommendation by popularity raise the question of whether this really reflects a virtue of the recommender, or some other bias in the metric. We seek to shed some light on the question by a closer study.

4.5.1 Popularity-Driven Recommendation

Even though they contradict the personalisation principle, the good results of popularity recommendation can be given an intuitive explanation. By averaging over all users, precision metrics measure the overall satisfaction of the user population. A method that gets to satisfy a majority of users is very likely to perform well under such metrics. In other words, average precision metrics tend to favour the satisfaction of majorities, regardless of the dissatisfaction of minorities, whereby algorithms that target majority tastes will expectably yield good results on such metrics. This implicitly relies on the fact that on a random item split, the number of test ratings for an item correlates with its number of training ratings, and its number of positive ratings correlates with the total number of ratings. More formally, the advantage of

popularity-oriented recommendation comes from the fact that in a random rating split, $\mathrm{E}[pr_{test}(i)] \propto pr(i) \propto \mathrm{E}[pr_{train}(i)] \propto r_{train}(i)$, which means that the items with many training ratings will tend to have many positive test ratings, that is, they will be liked by many users according to the test data. We analyse this next, more formally and in more detail.

In a popularity recommender $i_k^{u,POP}$ is the k-th item in the target set with most ratings in the training set – i.e., the system ranks items by decreasing order of $r_{train}(i_k^{u,POP})$. This ranking is almost user-independent (except for those, statistically negligible, user items already in training which are excluded from the ranking) and therefore, for an AR experimental design, Equation (4.3) applies. Since we have $\sum_{k=1}^{n} pr(i_k^{u,POP}) = \max_s \sum_{k=1}^{n} pr(i_k^{u,s})$ (as far as $\mathrm{E}[pr_{test}(i)] \propto r_{train}(i)$ for a random training-test split), the popularity recommendation is the best possible non-personalised system, maximising $\mathrm{E}[\mathrm{P}_s@n]$. Popularity thus achieves a considerably high precision value, just for statistical reasons.

For a 1R experimental design, using Equation (4.2) (lemma) we have:

$$\frac{|\mathcal{U}|E[P_s@n]}{pr_{test}} \leq E[1RP_s@n] \leq \frac{\sum_u E[m_u P_s^u@m_u(T_u)]}{n \cdot pr_{test}}$$

Now, since $\mathrm{P}_s@n$ and $\mathrm{P}_s^u@m_u$ above are computed by AR, we may elaborate from Equation (4.3) for a non-personalised recommender, and we get:

$$\frac{|\mathcal{I}|}{n \cdot t \cdot pr} \underset{u \in \mathcal{U}}{avg} \sum_{k=1}^{n} pr(i_k^{u,s}) \leq E[1RP_s@n] \leq \frac{|\mathcal{C}|}{n \cdot t \cdot pr} \underset{u \in \mathcal{U}}{avg} \sum_{k=1}^{m_u} pr(i_k^{u,s})$$

This experimental approach is thus equally biased to popular items, since the latter optimise $\sum_{k=1}^{n} pr(i_k^{u,s})$.

Note that the advantage of popularity over other recommenders is highly dependent on the skewness in the distribution of ratings over items: if all items were equally popular, the popularity recommender would degrade to random recommendation – in fact slightly worse, as $pr_{test}(i) \propto r_{test}(i) = r/|\mathcal{I}| - r_{train}(i)$, so popular items would have fewer positive test ratings. On the other extreme, if a few items (less than n) are liked by most users, and the rest are liked by very few, then popularity approaches the maximum precision possible.

4.5.2 Popularity Distributions

In order to illustrate how the dependence between the popularity precision and the background popularity distribution evolves, we simulate different degrees of skewness in rating distributions. As a simulated distribution pattern we use a shifted power law $r(i_k) = c_1 + \beta(c_2 + k)^{-\alpha}$, where α determines the skewness (e.g. $\alpha \sim 1.4$ for MovieLens 1M). Figure 4.4 (left) shows the shape of generated distributions ranging

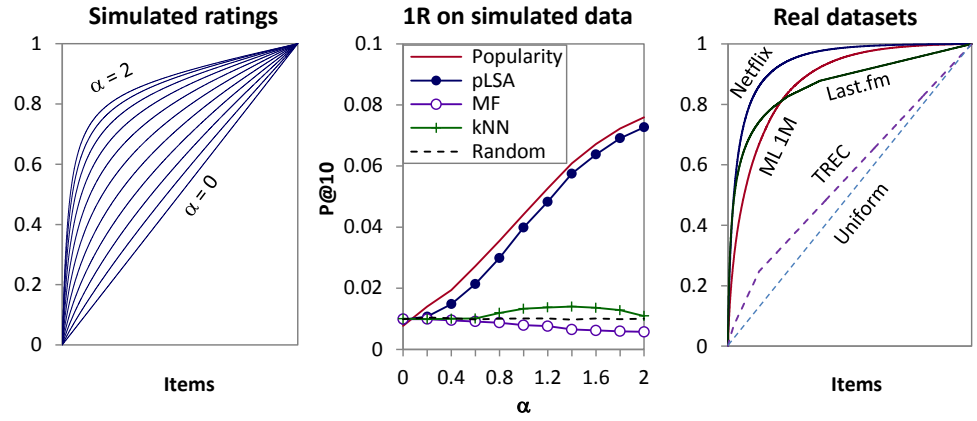


Figure 4.4. Effect of popularity distribution skewness on the popularity bias. The left graphic shows the cumulated popularity distribution of artificial datasets with simulated ratings, with skewness ranging from α = 0 to 2. The x axis represents items by popularity rank, and the y axis displays the cumulative ratio of ratings. The central graphic shows the precision of different recommendation algorithms on each of these simulated datasets. The right graphic shows the cumulative distribution of positive ratings in real datasets.

from uniform ($\alpha = 0$) to a very steep long-tailed popularity distribution ($\alpha = 2$), and (center) how the measured precision evolves in this range. The artificial data are created with the same number of users, items, and ratings (therefore the same rating density) as in MovieLens 1M, setting c_1 and c_2 by a fit to this dataset, and enforcing these constraints by adjusting β. The rating values are assigned randomly on a 1-5 scale, also based on the prior distribution of rating values in Movie-Lens.

The results in Figure 4.4 (center) evidence the fact that the precision of popularity-based recommendation is heavily determined by the skewness of the distribution. It benefits from steep distributions, and degrades to slightly below random (0.0077 vs. 0.0100) when popularity is uniform. This slightly below-random performance of popularity recommendation at $\alpha = 0$ is explained by the fact that $E[pr_{test}(i)] \propto E[r_{test}(i)] = r(i) - E[r_{train}(i)]$ is inverse to the popularity ranking by $r_{train}(i)$ when $r(i)$ is uniform, as predicted at the end of the previous section. kNN and MF stay essentially around random recommendation. This is because the data are devoid of any consistent preference pattern (as collaborative filtering techniques would assume) in this experiment, since the ratings are artificially assigned at random, and the results just show the "pure" statistical dependency to the popularity distribution. pLSA does seem to take advantage of item popularity, as it closely matches the effectiveness of popularity recommendation. We show only the 1R design, but the effect is the same in AR.

This observation also explains the difference between datasets from IR and those from recommendation with regards to the popularity bias. Figure 4.4 (right) shows the cumulative distribution of positive user interaction data per item in three

datasets: Netflix, MovieLens, and Last.fm (the dataset in Section 4.3.3). The shapes of the curves are typical of long-tailed distributions, where a few popular items accumulate most of the preference data (Celma, 2010; Celma and Cano, 2008). This contrasts with the distribution of positive relevance judgments over documents in TREC data (same figure) – where we have aggregated 30 individual tracks, filtering out the documents that are not relevant to any query, and obtaining a set of 703 queries, 129,277 documents, and 149,811 positive judgments. The TREC distribution is considerably flatter, not far from uniform: 87.2% of documents are relevant to just one query, and the maximum number of positive assessments per document is 25 (3.6% of queries), whereas the top popular item in Netflix, MovieLens, and Last.fm, is liked by 20.1%, 32.7% and 73% of users, respectively.

Several reasons account for this difference between retrieval and recommender datasets. First, in IR queries are selected by design, intending to provide a somewhat varied testbed to compare retrieval systems. Hence, including similar queries with overlapping relevance would not make much sense. Second, queries in natural search scenarios are generally more specific and narrower than global user tastes for recommendation, whereby the corresponding relevant sets have much less intersection. Furthermore, the TREC statistics we report are obtained by aggregating the data of many tracks, in order to seek any perceptible popularity slant. The typical TREC experiments are actually run on separate tracks comprising typically 50 queries, where very few documents, if any, are relevant to more than one query. Note also that even though we have filtered out over 0.7 million non-relevant plus nearly 5 million unlabeled documents in the TREC statistics, the non-relevant documents actually remain as input to the systems, contrarily to experiments in the recommender domain, thus making up an even flatter relevance distribution. Moreover, in the usual IR evaluation setting, the systems have no access to the relevance data – thus, they have no means to take a direct bias towards documents with many judgments –, whereas in recommendation, this is the primary input the systems (particularly collaborative filtering recommenders) build upon. The popularity phenomenon has therefore never been an issue in IR evaluation, and neither the metrics nor the methodologies have had to even consider this problem, which arises now when bringing them to the recommendation setting – where the overlap between user preferences is not only common, but actually needed by collaborative filtering algorithms.

4.6 Overcoming the popularity bias

After analysing the effects of popularity in precision metrics, the issue remains: to what extent do the good results of popularity recommendation reflect only a statistical bias in a metric, or any degree of actual recommendation quality? The same question should be raised for pLSA, which seems to follow the popularity trends quite

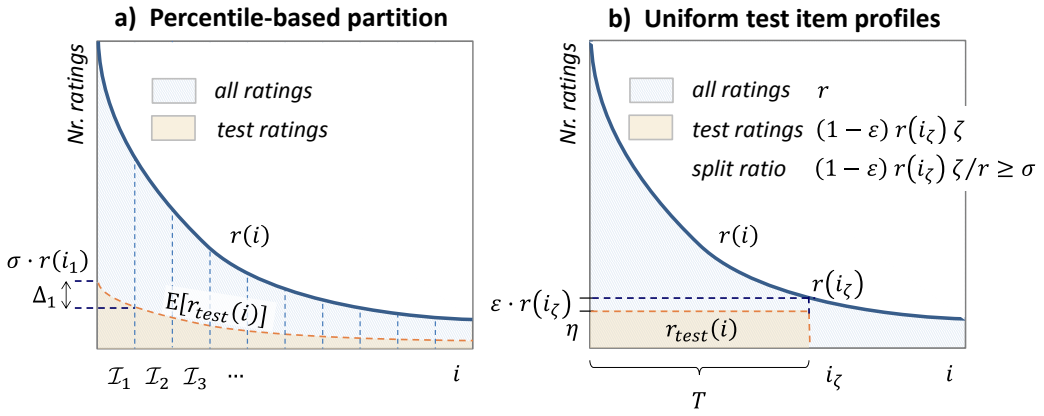


Figure 4.5. Rating splits by a) a popularity percentile partition (left), and b) a uniform number of test ratings per item (right). On the left, the red dashed split curve represents $E[pr_{test}(i)]$ – i.e., the random split ratio needs not be applied on a per-item basis – whereas on the right it does represent $pr_{test}(i)$.

closely. We address the question by proposing and examining alternative experimental configurations, where the statistical role of popularity gets reduced, as we propose next.

4.6.1 Percentile-Based Approach (P1R)

We propose a first approach to neutralise the popularity bias, which consists in partitioning the set of items into m popularity percentiles $\mathcal{I}_k \subset \mathcal{I}$, breaking down the computation of accuracy by such percentiles, and averaging the m obtained values. By doing so, in a common long-tailed popularity distribution, the margin for the popularity bias is considerably reduced, as the difference Δ_k in the number of positive test ratings per item between the most and least popular items of each percentile is not that high. The popularity recommender is forced to recommend as many unpopular as popular items, thus leveling the statistical advantage to a significant extent. It remains the optimal non-personalised algorithm, but the difference – and thus the bias – is considerably reduced. The technique is illustrated in Figure 4.5a.

A limitation of this approach is that it restricts the size of the target sets by $|T_u| \leq |\mathcal{I}|/m$. For instance, for $m = 10$ in MovieLens 1M, this imposes a limit of $|T_u| \leq \sim 370$, which seems acceptable for 1R. The restriction can be more limiting in the AR approach, e.g. the TI and AI options cannot be applied (except within the percentiles). For this reason, we will only apply the percentile technique in the 1R design, a configuration to which we shall refer as **P1R**.

4.6.2 Uniform Test Item Profiles (UAR, U1R)

We now propose a second technique consisting of the formation of data splits where all items have the same amount of test ratings. The assumption is that the items with a high number of training ratings will no longer have a statistical advantage by having more positive test ratings. That is, the relation $\mathrm{E}[pr_{test}(i)] \propto r_{train}(i)$ described in Section 4.5.1 breaks up. The approach consists of splitting the data by picking a set T of candidate items, and a number η of test ratings per item so that $|T|\eta/r = \sigma$. For this to be possible, it is necessary that $(1-\varepsilon)\, r(i) \geq \eta, \forall i \in T$, where ε is a minimum ratio of training ratings per item we consider appropriate. In particular, in order to allow for n-fold cross-validation, we should have $\varepsilon \geq 1/n$. The selection of T can be done in several ways. We propose to do so in a way that it maximises $|T|$, i.e., to use as many different target test items as possible, avoiding a biased selection towards popular items. If we sort $i_k \in \mathcal{I}$ by popularity rank, it can be seen that this is achieved by picking $T = \{i_k \in \mathcal{I} | k \leq \zeta\}$ with $\zeta = \max\{k|\,(1-\varepsilon)\, r(i_k)\, k/r \geq \sigma\}$, so that $\eta = (1-\varepsilon)\, r(i_\zeta)$. Figure 4.5b illustrates this procedure.

The expected effect of this approach is that the statistical relation $\mathrm{E}[pr_{test}(i)] \propto pr(i)$ no longer holds, and neither should hold now, as a consequence, the rationale described in Section 4.5.1 for popularity being the optimum non-personalised recommender. In fact, since $\mathrm{E}[pr_{test}(i)] = \eta \cdot pr(i)/r(i)$ for any $i \in T$, and $\eta = \sigma \cdot r/|T|$, it can be seen that if $\mathcal{C} = T$ (TI policy) Equation (4.3) for AR yields:

$$E[P_s@n] = \frac{\sigma \cdot r}{n \cdot t|\mathcal{U}|} \sum_{k=1}^{n} \underset{u \in \mathcal{U}}{avg} \frac{pr\left(i_k^{u,s}\right)}{r\left(i_k^{u,s}\right)}$$

for any non-personalised recommender. If the ratio $pr\left(i_k^{u,s}\right)/r\left(i_k^{u,s}\right)$ of positive ratings does not depend on k, we have $\mathrm{E}[\mathrm{P}_s@n] = \mathrm{E}[\mathrm{P}_{RND}@n] = \sigma \cdot \delta$. This means that popularity recommendation may get some advantage over other recommenders only if – and to the extent that – popular items have a higher ratio of positive ratings than unpopular items, and popularity recommendation will degrade to random precision otherwise. On the other hand, it can be seen that if $\mathcal{C} \supsetneq T$ (i.e., the TI policy is not adhered to), then $\mathrm{E}[\mathrm{P}_{RND}@n]$ would get reduced by a factor of $|T|/|\mathcal{C}|$.

For a non-personalised recommender in a 1R design, elaborating from Equations (4.2) and (4.3) we get:

$$\frac{r}{n \cdot t \cdot pr} \underset{u \in \mathcal{U}}{avg} \sum_{k=1}^{n} \frac{pr\left(i_k^{u,s}\right)}{r\left(i_k^{u,s}\right)} \leq E[1RP_s@n] \leq \frac{r}{n \cdot t \cdot pr} \underset{u \in \mathcal{U}}{avg} \sum_{k=1}^{m_u} \frac{pr\left(i_k^{u,s}\right)}{r\left(i_k^{u,s}\right)},$$

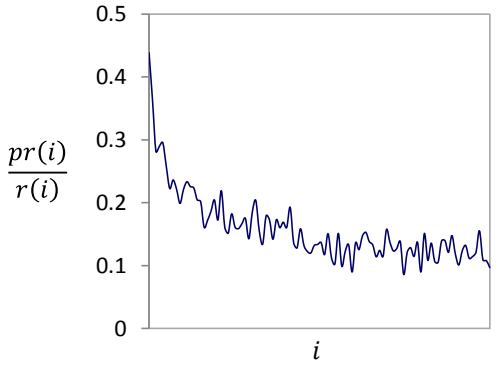


Figure 4.6. Positive ratings ratio vs. popularity rank in MovieLens 1M. The graphic plots $pr(i)/r(i)$, where items are ordered by decreasing popularity. We display averaged values for 100 popularity segments, for a smoothed trend view.

an equivalent situation where the measured precision of popularity recommendation is bound by the potential dependence between the ratio of positive ratings and popularity.

Figure 4.6 shows this ratio as $pr(i)/r(i)$ with respect to the item popularity rank in MovieLens 1M. It can be seen that indeed the ratio grows with popularity in this dataset, which does lend an advantage for popularity recommendation. Even so, we may expect the bias to be moderate – but this has to be tested empirically, as it depends on the dataset. Note also that in applications where all ratings are positive (as e.g. in our Last.fm setup), popularity – and any non-personalised recommender – would drop exactly to random precision ($\mathrm{E}[\mathrm{P}_s@n] = \sigma \cdot \delta$ in AR and $1/t$ in 1R).

A limitation of this approach is that the formation of T may impose limits on the value of σ, and/or the size of T. If the popularity distribution is very steep, T may turn out small and therefore biased to a few popular items. Moreover, there is in general a solution for T only up to some value of σ – it is easy to see (formally, or just visually in Figure 4.5) that as $\sigma \to 1$ there is no item for which $(1-\varepsilon)\, r(i_k)\, k/r \geq \sigma$, unless the popularity distribution was uniform, which is never the case in practice. We have however not found these limitations to be problematic in practice, and common configurations turn out to be feasible without particular difficulty. For instance, in MovieLens 1M we get $|T|$ = 1,703 for $\sigma = 0.2$ with ε = 0.2 (allowing for a 5-fold cross-validation), resulting in η = 118 test ratings per item.

This method can be used, as noted, in both the AR and 1R approaches. We shall refer to these combinations as **UAR** and **U1R** respectively, where 'U' stands for the "uniform" number of item test ratings. In U1R it is important to set $\mathcal{C} = T$ in order to sample non-relevant items within T (i.e., $N_u \subset T$, for the TI policy). Otherwise, popularity would have a statistical advantage over other recommenders, as it would systematically rank irrelevant items in $N_u - T$ below any relevant item in T, whereas

other algorithms might not. The same can be considered in UAR, unless the experimental setup requires $|T_u| > |T|$, as e.g. in the AI design. In that case a slight popularity bias would arise, as we shall see next.

4.6.3 Experimental Results

Figure 4.7 compares the results measured by 1R, AR and their corresponding popularity-neutralising variants. The setup is the same as in previous sections, except that for AR, we take TI-NN with $|N_u|$ = 1,700, to level with UAR in random precision. All the results correspond to MovieLens 1M except Last.fm where indicated. It can be seen that P1R, U1R and UAR effectively limit the popularity bias. The techniques seem to be more effective on 1R than AR: U1R and (even more) P1R actually place the popularity algorithm by the level of random recommendation, whereas the measured popularity precision decreases in UAR, but remains above kNN. The advantage of popularity over randomness in U1R and P1R is explained by the bias in the ratio of positive ratings in popular items (Figure 4.6). This ratio is constant in Last.fm, whereby popularity drops to random in U1R, as predicted by our analysis in the previous section, proving that the popularity bias remaining in the uniform-test approach is caused by this factor. This residual bias is higher in U1R than P1R, because in the former, N_u is sampled over a larger popularity interval ($|T|$ = 1,703 vs. $|\mathcal{I}|$ / 10 = 370 items), giving a higher range for advantage by popularity, which also explains why the latter still overcomes kNN in UAR. We may observe the importance of using the TI policy in UAR, without which (in AI-UAR) a higher bias remains. We also show the effect of removing the 10% most popular head items from the test data (and also from $\mathcal{C}$, i.e., they are excluded from N_u sampling) in 1R, as a simple strategy to reduce the popularity bias (Cremonesi et al., 2010). We see that this technique reduces the measured precision of popularity, but it is not quite as effective as the proposed approaches.

It is finally worth emphasising how **the percentile and uniform-test approaches discriminate between pure popularity-based recommendation and an algorithm like pLSA**, which does seem to take popularity as one of its signals, but not the only one. The proposed approaches allow uncovering the difference, neutralising popularity but not pLSA, which remains the best algorithm in all configurations.

As we mentioned in Section 4.3, we have taken precision as a simple and common metric for our study, but all the presented analysis and proposed alternatives straightforwardly generalise to other standard IR metrics, such as MAP, nDCG, and Mean Reciprocal Rank (MRR). Their application is direct in the AR setting; and they can be applied in 1R by simply introducing them in place of precision in the internal summation of Equation (4.1). Figure 4.7 shows results for nDCG, where we see that the analysed patterns hold just the same. The AR approach provides room for a

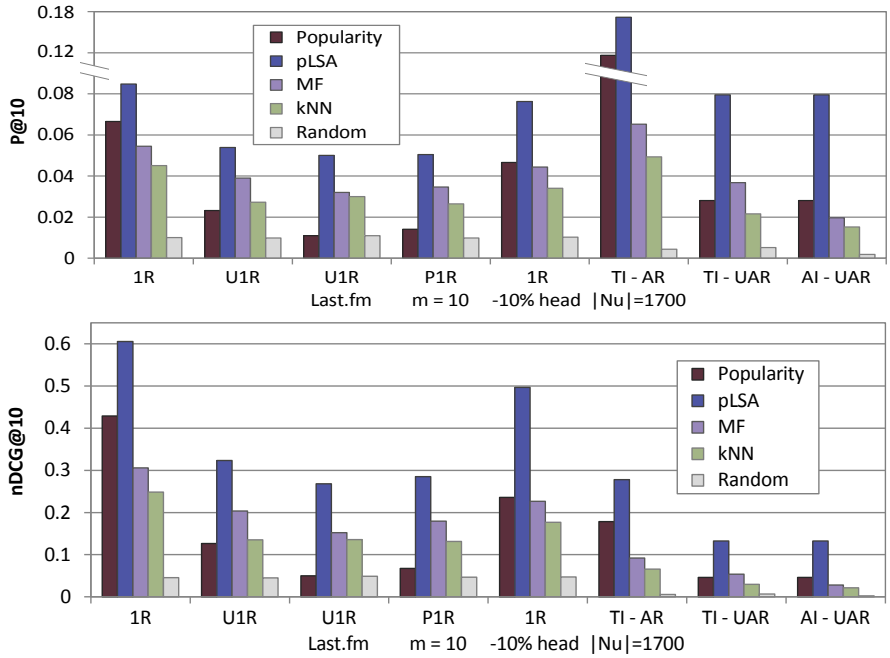


Figure 4.7. Precision and nDCG of recommendation algorithms on MovieLens 1M (and Last.fm only where indicated) using the 1R, U1R, P1R (m = 10 percentiles), AR, and UAR methodologies. The "-10% head" bars show the effect of removing the 10% most popular items from the test data (Cremonesi et al., 2010).

slightly wider metric variety than 1R, in the sense that some metrics reduce to each other in 1R. For instance, for a single relevant item, MAP is equivalent to Mean Reciprocal Rank ($\mathrm{MRR} = 1/k$ where k is the rank of the first relevant item). And nDCG is insensitive to relevance grades in 1R (the grade of the single relevant item cancels out), whereas grades do make a difference in AR.

4.7 Conclusions

The application of Information Retrieval methodologies to the evaluation of recommender systems is not necessarily as straightforward as it may seem. Hence, it deserves close analysis and attention to the differences in the experimental conditions, and their implications on the explicit and implicit principles and assumptions on which the metrics build. We have proposed a systematic characterisation of design alternatives in the adaptation of the Cranfield paradigm to recommendation tasks, aiming to contribute to the convergence of evaluation approaches. We have identified assumptions and conditions underlying the Cranfield paradigm which are not granted in usual recommendation experiments. We have detected and examined resulting statistical biases, namely test sparsity and item popularity, which do not arise in common test collections from IR, but do interfere in recommendation experi-

ments. Sparsity is clearly a noisy variable that is meaningless with respect to the value of a recommendation. Whether popularity is in the same case is less obvious; we propose experimental approaches that neutralise this bias, leaving way to an unbiased observation of recommendation accuracy, isolated from this factor. With a view to their practical application, we have identified and described the pros and cons of the array of configuration alternatives and variants analysed in this study.

In general, we have found that evaluation metrics computed in AR and 1R approaches differ in how they are averaged. This means, more specifically, that precision obtained by approaches following a 1R design is bound linearly by precision of AR approaches. Moreover, we have observed that a percentile-based evaluation considerably reduces the margin for the popularity bias, although the main limitation of this approach is that it specifies a constraint on the size of the possible target sets. Additionally, a uniform-test approach removes any statistical advantage provided by having more positive test ratings. Furthermore, we have found that both approaches discriminate between pure popularity-based recommendation and an algorithm like pLSA.

The main goal of our research addresses a second-order problem: we aim to predict the accuracy of the predictions of recommendation algorithms. As we shall see, the (second-order) evaluation of our researched methods relies on the (first-order) evaluation metrics and methodologies by which the recommendation algorithms' accuracy is measured. In order to consistently evaluate our methods, the primary recommendation evaluation has to be reliable and well-understood. Any bias in the process would lead to inconclusive or misleading results about the predictive power of our methods. For this reason, the results presented in this chapter are a necessity for the main goal of this thesis, but the outcome can be of more general use. Specifically, in the following chapters we shall compare how the different methodologies (with and without neutralised biases) may impact the observations on the predictive power of our predictors.

The popularity effects in recommender systems have started to be reported in recent work (Cremonesi et al., 2011; Cremonesi et al., 2010; Steck, 2011). Our research complements such findings by seeking principled theoretical and empirical explanations for the biases, and providing solutions within the frame of IR evaluation metrics and methodology – complementarily to the potential definition of new special-purpose metrics (Steck, 2011). The extent to which popularity is a noisy signal may be further analysed by developing more complete metric schemes incorporating gain and cost dimensions, where popular items would expectably score lower. Such metrics may e.g. account for the benefits (to both recommendation consumers and providers) drawn from novel items in typical situations (Vargas and Castells, 2011), as a complement to plain accuracy. Online tests with real users should also be valuable for a comparative assessment of offline observations, and the validation of experimental alternatives.

Part III

Predicting performance in recommender systems

You can never get a cup of tea large enough or
a book long enough to suit me.
Clive Staples Lewis

Chapter 5

Performance prediction in Information Retrieval

Information retrieval performance prediction has been mostly addressed as a query performance issue, which refers to the performance of an information retrieval system in response to a specific query. It also relates to the appropriateness of a query as an expression of the user's information needs. In general, performance prediction methods have been classified into two categories depending on the used data: pre-retrieval approaches, which make the prediction before the retrieval stage using query features, and post-retrieval approaches, which use the rankings produced by a retrieval engine. In particular, the so-called clarity score predictor – of special interest for this thesis – has been defined in terms of language models, and captures the ambiguity of a query with respect to the utilised document collection, or a specific result set.

In this chapter we provide an overview of terminology, techniques, and evaluation related to performance prediction in Information Retrieval. In Section 5.1 we introduce terminology and foundamental concepts of the performance prediction problem. In Section 5.2 we describe the different types of performance prediction approaches, which are mainly classified in the two categories mentioned above: pre-retrieval and post-retrieval approaches. Then, in Section 5.3 we provide a thorough analysis on the use of clarity score as a performance prediction technique, including examples, adaptations, and applications found in the literature. Finally, in Section 5.4 we introduce the general methodology used to evaluate performance predictors, along with the most common methods to measure their quality.

5.1 Introduction

Performance prediction has received little attention, if any, to date in the Recommender Systems field. Our research, however, finds a close and highly relevant reference in the adjacent Information Retrieval discipline, where performance prediction has gained increasing attention since the late 90's, and has become an established research topic in the field. Performance prediction finds additional motivation in personalised recommendation, inasmuch the applications they are integrated in may decide to produce recommendations or hold them back, delivering only the sufficiently reliable ones. Moreover, the ability to predict the effectiveness of individual algorithms can be envisioned as a strategy to optimise the combination of algorithms into ensemble recommenders, which currently dominate the field – rarely if ever are individual algorithms used alone in working applications, neither are they found individually in the top ranks of evaluation campaigns and competitions (Bennett and Lanning, 2007).

In Information Retrieval performance prediction has been mostly addressed as a query performance problem (Cronen-Townsend et al., 2002). Query performance refers to the performance of an information retrieval system in response to a particular query. It also relates to the appropriateness of a query as an expression of a user's information needs. Dealing effectively with poorly-performing queries is a crucial issue in Information Retrieval since it could improve the retrieval effectiveness significantly (Carmel and Yom-Tov, 2010).

In general, performance prediction techniques can be useful from different perspectives (Zhou and Croft, 2006; Yom-Tov et al., 2005a):

- From the user's perspective, it provides valuable feedback that can be used to direct a search, e.g. by rephrasing the query or suggesting alternative terms.
- From the system's perspective, it provides a means to address the problem of information retrieval consistency. The consistency of retrieval systems can be addressed by distinguishing poorly performing queries. A retrieval system may invoke different retrieval strategies depending on the query, e.g. by using query expansion or ranking functions based on the predicted difficulty of the query.
- From the system administrator's perspective, it may let identify queries related to a specific subject that are difficult for the search engine. According to such queries, the collection of documents could be extended to better answer insufficiently covered topics.
- From a distributed information retrieval's perspective, it can be used to decide which search engine (and/or database) to use, or how much weight give to different search engines when their results are combined.

Specifically, the performance prediction task in Information Retrieval is formalised based on the following three core concepts: **performance predictor**, **retrieval quality assessment**, and **predictor quality assessment**. In this context, the performance predictor is defined as a function that receives the query (and the result list D_q retrieved by the system, the set of relevant documents R_q, collection statistics $\mathcal{C}$, etc.), and returns a prediction of the retrieval quality for that query. Then, by means of a predictor quality assessment method, the predictive power of the performance predictor is estimated.

Based on the notation given in (Carmel and Yom-Tov, 2010), the problem of performance prediction consists of estimating a true retrieval quality metric $\mu(q)$ (retrieval quality assessment) of an information retrieval system for a given query q. Hence, a performance predictor $\hat{\mu}(q)$ has the following general form:

$$\hat{\mu}(q) \leftarrow \gamma\left(q, R_q, D_q, \mathcal{C}\right) \tag{5.1}$$

The prediction methods proposed in the literature establish different functions γ, and use a variety of available data, such as the query's terms, its properties with respect to the retrieval space (Cronen-Townsend et al., 2002), the output of the retrieval system – i.e., D_q and R_q – (Carmel et al., 2006), and the output of other systems (Aslam and Pavlu, 2007).

According to whether or not the retrieval results are used in the prediction process, such methods can be classified into pre-retrieval and post-retrieval approaches, which are described in Sections 5.2.1 and 5.2.2, respectively. Another relevant distinction is based on whether the predictors are trained or not, but this classification is less popular, and will not be considered here.

Moreover, the standard methodology to measure the effectiveness of performance prediction techniques (that is, the predictor quality assessment method) consists of comparing the rankings of several queries based on their actual precision – in terms of a an evaluation metric such as MAP – with the rankings of those queries based on their performance scores, i.e., their predicted precision. In Section 5.4 we detail this methodology, along with several techniques for comparing the above rankings.

5.1.1 Notion of performance in Information Retrieval

In order to identify good performance predictors, validating or assessing their potential, we first have to define metrics of actual performance. Performance metrics and evaluation have been a core research and standardisation area for decades in the Information Retrieval field. In this section we introduce and summarise the main performance metrics and evaluation methodologies developed in the field.

The notion of performance in general, and in Information Retrieval in particular, leads itself to different interpretations, views and definitions. A number of methods

for measuring performance have been proposed and adopted (Hauff et al., 2008a; Hauff, 2010), the most prominent of which will be summarised herein; see (Baeza-Yates and Ribeiro-Neto, 2011) for an extended discussion.

As a result of several decades of research by the Information Retrieval community, a set of standard performance metrics has been established as a consensual reference for evaluating the goodness of information retrieval systems. These metrics generally require a collection of documents and a query (or alternative forms of user input such as item ratings), and assume a ground truth notion of relevance – traditional notions consider this relevance as binary, while others, more recently proposed, consider different relevance degrees.

One of the simplest and widespread performance metrics in Information Retrieval is **precision**, which is defined as the ratio of retrieved documents that are relevant for a particular query. In principle, this definition takes all the retrieved documents into account, but can also consider a given cut-off rank as the **precision at n** or **P@n**, where just the top-n ranked documents are considered. Other related and widespread metric is **recall**, which is the fraction of relevant documents retrieved by the system. These two metrics are inversely related, since increasing one generally reduces the other. For this reason, usually, they are combined into a single metric – e.g. the **F-measure**, and the **Mean Average Precision** or **MAP** –, or the values of one metric are compared at a fixed value of the other metric – e.g. the **precision-recall curve**, which is a common representation that consists of plotting a curve of precision versus recall, usually based on 11 standard recall levels (from 0.0 to 1.0 at increments of 0.1).

An inherent problem of using MAP for poorly performing queries, and in general of any query-averaged metric, is that changes in the scores of better-performing queries mask changes in the scores of poorly performing queries (Voorhees, 2005b). For instance, the MAP of a baseline system in which the effectiveness is 0.02 for a query A, and 0.40 for a query B, is the same as the MAP of a system where query A doubles its effectiveness (0.04) and query B decreases a 5% (0.38). In this context, in (Voorhees, 2005a) two metrics were proposed to measure how well information retrieval systems avoid very poor results for individual queries: the **%no measure**, which is the percentage of queries that retrieved no relevant documents in the top 10 ranked results, and the **area measure**, which is the area under the curve produced by plotting MAP(X) versus X, where X ranges over the worst quarter queries. These metrics were shown to be unstable when evaluated in small sets of 50 queries (Voorhees, 2005b). A third metric was introduced in (Voorhees, 2006): **gmap**, the geometric mean of the average precision scores of the test set of queries. This metric emphasises poorly performing queries while it minimises differences between larger scores, remaining stable in small sets of queries (e.g. 50 queries) (Voorhees, 2005b).

Nonetheless, despite the above metrics and other efforts made to obtain better measures of query performance, MAP, and more specifically the **Average Precision per query**, are still widely used and accepted. See (Carmel et al., 2006; Cronen-Townsend et al., 2002; Hauff et al., 2008b; He and Ounis, 2004; He et al., 2008; Kompaoré et al., 2007; Zhao et al., 2008; Zhou and Croft, 2006; Zhou and Croft, 2007), among others.

Almost as important as the performance metric is the query type, which can be related to the differerent user information needs (Broder, 2002). Most work on performance prediction has focused on the traditional ad-hoc retrieval task where query performance is measured according to topical relevance (also known as content-based queries). Some work – such as (Plachouras et al., 2003) and (Zhou and Croft, 2007) – has also addressed other types of queries such as named page finding queries, i.e., queries focused on finding the most relevant web page assuming the queries contain some form of the "name" of the page being sought (Voorhees, 2002a).

When documents are timed (e.g. a newswire system), we can also distinguish two main types of queries that have been only partially exploited in the literature (Diaz and Jones, 2004; Jones and Diaz, 2007): those queries that favour very recent documents, and those queries for which there are more relevant documents within a specific period in the past.

Finally, we note that most of the research ascribed to predict performance has been focused not on predicting the "true" performance of a query (whatever that means), but on discriminating those queries where query expansion or relevance feedback algorithms have proved to be efficient from those where these algorithms fail, such as polisemic, ambiguous, and long queries. These are typically called *bad-to-expand* queries (Cronen-Townsend et al., 2006), illustrating the implicit dependence on their final application.

5.1.2 A taxonomy of performance prediction methods

Existing prediction approaches are typically categorised into pre-retrieval methods and post-retrieval methods (Carmel and Yom-Tov, 2010). Pre-retrieval methods make the prediction before the retrieval stage, and thus only exploit the query's terms and statistics about these terms gathered at indexing time. In contrast, post-retrieval methods use the rankings produced by a search engine, and, more specifically, the score returned for each document along with statistics about such documents and their vocabulary.

Category	Sub-category	Performance predictor (name and reference)
Pre-retrieval	Linguistics	Morphological, syntactic, semantic: (Mothe and Tanguy, 2005), (Kompaoré et al., 2007)
	Statistics	Coherency: coherence (He et al., 2008); term variance (Zhao et al., 2008) Similarity: collection query similarity (Zhao et al., 2008) Specificity: IDF-based (Plachouras et al., 2004), (He and Ounis, 2004); query scope (He and Ounis, 2004), (Macdonald et al., 2005); simplified clarity: (He and Ounis, 2004) Term relatedness: mutual information (Hauff et al., 2008a)
Post-retrieval	Clarity	Clarity (Cronen-Townsend et al., 2002), (Cronen-Townsend et al., 2006) Improved clarity (Hauff, 2010) (Hauff et al., 2008b) Jensen-Shannon Divergence (Carmel et al., 2006) Query difficulty (Amati et al., 2004)
	Robustness	Cohesion: clustering tendency (Vinay et al., 2006); spatial autocorrelation (Diaz, 2007); similarity (Kwok et al., 2004), (Grivolla et al., 2005) Document perturbation: ranking robustness (Zhou and Croft, 2006); document perturbation (Vinay et al., 2006) Query perturbation: query feedback (Zhou and Croft, 2007); autocorrelation (Diaz and Jones, 2004) (Jones and Diaz, 2007); query perturbation (Vinay et al., 2006); sub-query overlap (Yom-Tov et al., 2005a) Retrieval perturbation: (Aslam and Pavlu, 2007)
	Score analysis	Normalised Query Commitment: (Shtok et al., 2009) Standard deviation of scores: (Pérez-Iglesias and Araujo, 2009), (Cummins et al., 2011) Utility Estimation Framework: (Shtok et al., 2010) Weighted Information Gain: (Zhou and Croft, 2007)

Table 5.1. Overview of predictors presented in Section 5.2 categorised according to the taxonomy presented in (Carmel and Yom-Tov, 2010).

Pre-retrieval performance predictors do not rely on the retrieved document set, but on other information mainly extracted from the query issued by the user, such as statistics computed at indexing time (e.g. inverse term document frequencies). They have the advantage that predictions can be produced before the system's response is even started to be elaborated, which means that predictions can be taken

into account to improve the retrieval process itself. However, they have a potential handicap with regards to their accuracy on the predictions, since extra retrieval effectiveness cues available with the system's response are not exploited (Zhou, 2007). Pre-retrieval query performance has been studied from two main perspectives: based on probabilistic methods (and more generally, on collection statistics), and based on linguistic approaches. Most research on the topic has followed the former approach. Some researchers have also explored inverse document frequency (IDF) and related features as predictors, along with other collection statistics

Post-retrieval performance predictors, on the other hand, make use of the retrieved results. Broadly speaking, techniques in this category provide better prediction accuracy compared to pre-retrieval performance predictors. However, many of these techniques suffer from high computational costs. Besides, they cannot be used to improve the retrieval strategies without a post-processing step, as the output from the latter is needed to compute the predictions in the first place. In (Carmel and Yom-Tov, 2010) post-retrieval methods are classified as follows: 1) clarity based methods that measure the coherence (clarity) of the result set and its separability from the whole collection of documents; 2) robustness based methods that estimate the robustness of the result set under different types of perturbations; and 3) score analysis based methods that analyse the score distribution of results.

Table 5.1 shows a number of representative approaches on performance prediction, which will be described in the next section. These approaches are categorised according to the taxonomy and sub-categories proposed in (Carmel and Yom-Tov, 2010). In the table we can observe that the statistics category has been the most popular approach for pre-retrieval performance prediction. Several predictors have been categorised in the robustness category, probably due to its broad meaning (query, document, and retrieval perturbation). Finally, we note that recent effort from the community has been focused on the score analysis category.

5.2 Query performance predictors

In this section we explain the distinct performance predictors proposed in the literature. As mentioned before, based on whether or not retrieval results are needed to compute performance scores, predictors can be classified into two main types: pre-retrieval and post-retrieval predictors. In the following we summarise some of the approaches of each of the above types. For additional information, the reader is referred to (Carmel and Yom-Tov, 2010), (Hauff, 2010), and (Pérez Iglesias, 2012).

5.2.1 Pre-retrieval predictors

Pre-retrieval performance predictors do not rely on the retrieved document set, and exploit other collection statistics, such as the inverse document frequency (IDF). In this context, performance prediction has been studied from three main perspectives: based on linguistic methods, based on statistical methods, and based on probabilistic methods.

Linguistic methods

In (Mothe and Tanguy, 2005) and (Kompaoré et al., 2007) the authors consider 16 query features, and study their correlation with respect to average precision and recall. These features are classified into three different types according to the linguistic aspects they model:

- Morphological features:
  - **Number of words**.
  - **Average word length** in the query.
  - **Average number of morphemes per word**, obtained using the CELEX[7] morphological database. The limit of this method is the database coverage, which leaves rare, new, and misspelled words as mono-morphemic.
  - **Average number of suffixed tokens**, obtained using the most frequent suffixes from the CELEX database (testing if each lemma in a topic is eligible for a suffix from this list).
  - **Average number of proper nouns**, obtained by POS (part-of-speech) tagger's analysis.
  - **Average number of acronyms**, detected by pattern matching.
  - **Average number of numeral values**, also detected by pattern matching.
  - **Average number of unknown tokens**, marked by a POS tagger. Most unknown words happen to be constructed words such as "mainstreaming", "postmenopausal" and "multilingualism."
- Syntactic features:
  - **Average number of conjunctions**, detected through POS tagging.
  - **Average number of prepositions**, also detected through POS tagging.
  - **Average number of personal pronouns**, again detected through POS tagging.
  - **Average syntactic depth**, computed from the results of a syntactic analyser. It is a straightforward measure of syntactic complexity in terms of

[7] CELEX, English database (1993). Available at www.mpi.nl/world/celex

hierarchical depth; it simply corresponds to the maximum number of nested syntactic constituents in the query.

- **Average syntactic links span**, computed from the results of a syntactic analyser; it is the average pairwise distance (in terms of number of words) between individual syntactic links.

- Semantic features:
  - **Average polysemy value**, computed as the number of synsets in the WordNet[8] database that a word belongs to, and averaged over all terms of the query.

In the above papers the authors investigated the correlation between these features, and precision and recall over datasets with different properties, and found that the only feature that positively correlated with the two performance metrics was the number of proper nouns. Besides, many variables did not obtain significant correlations with respect to any performance metric.

Statistical methods

Inverse document frequency is one of the most useful and widely used magnitudes in Information Retrieval. It is usually included in the information retrieval models to properly compensate how common terms are. Its formulation usually takes an ad hoc, heuristic form, even though formal definitions exist (Roelleke and Wang, 2008; Aizawa, 2003; Hiemstra, 1998). The main motivation for the inclusion of an IDF factor in a retrieval function is that terms that appear in many documents are not very useful for distinguishing a relevant document from a non-relevant one. In other words, it can be used as a measure of the specificity of terms (Jones, 1972), and thus as an indicator of their discriminatory power. In this way, IDF is commonly used as a factor in the weighting functions for terms in text documents. The general formula of IDF for a term t is the following:

$$\mathrm{IDF}(t) = \log\frac{N}{N_t} \tag{5.2}$$

where N is the total number of documents in the system, and n_t is the number of documents in which the term t appears.

Some research work on performance prediction has studied IDF as a basis for defining predictors. He and Ounis (2004) propose a predictor based on the **standard deviation of the IDF** of the query terms. Plachouras et al. (2004) represent the quality of a query term by a modification of IDF where instead of the number of documents, the number of words in the whole collection is used (**inverse collection term**

[8] WordNet, lexical database for the English language. Available at http://wordnet.princeton.edu/

frequency, or ICTF), and the query length acts as a normalising factor. These IDF-based predictors displayed moderate correlation with query performance.

Other authors have taken the similarity of the query into account. Zhao et al. (2008) compute the vector-space based query similarity with respect to the collection, considered as a large document composed of concatenation of all the documents. Then, different **collection query similarity** predictors are defined based on the SCQ values (defined below) for each query term, by summing, averaging, or taking the maximum values:

$$\mathrm{SCQ}(t) = (1 + \log \mathrm{TF}(t)) \cdot \mathrm{IDF}(t) \tag{5.3}$$

The similarity of the documents returned by the query has also been explored in the field. The inter-similarity of documents containing query terms is proposed in (He et al., 2008) as a measure of **coherence**, by using the cosine similarity between every pair of documents containing each term. Additionally, two predictors based on the **pointwise mutual information** (PMI) are proposed in (Hauff et al., 2008a). The PMI of two terms is computed as follows:

$$\mathrm{PMI}(t_1, t_2) = \log \frac{p(t_1, t_2)}{p(t_1)p(t_2)} \tag{5.4}$$

where these probabilities can be approximated by maximum likelihood estimations, that is, based on collection statistics, where $p(t_1, t_2)$ is proportional to the number of documents containing both terms, and $p(t) \propto \mathrm{TF}(t)$. In that paper a first predictor is defined by computing the average PMI of every pair of terms in the query, whereas a second predictor is defined based on the maximum value. The predictive power of these techniques remains competitive, and is very efficient at run time.

Probabilistic methods

These methods measure characteristics of the retrieval inputs to estimate performance. He and Ounis (2004) propose a **simplified** version of the **clarity score** (see next section) in which the query model is estimated by the term frequency in the query:

$$\mathrm{SCS} = \sum_{w} P_{ml}(w|q) \log_2 \frac{P_{ml}(w|q)}{P(w|\mathcal{C})} \tag{5.5}$$

$$P_{ml}(w|q) = \frac{\mathrm{qtf}}{\mathrm{ql}}; P(w|\mathcal{C}) = \frac{\mathrm{TF}(w)}{|V|}$$

where qtf is the number of occurrences of a query term w in the query, ql is the query length, $\mathrm{TF}(w)$ is the number of occurrences of a query term in the whole collection, and $|V|$ is the total number of terms in the collection.

Despite its original formulation, where the clarity score can be considered as a pre-retrieval predictor (Cronen-Townsend et al., 2002), Cronen-Townsend and colleagues use result sets to improve the computation time. For this reason, it is typically classified as a post-retrieval predictor (Zhou, 2007; Hauff et al., 2008a), and thus, we describe it with more detail in the next sections.

Kwok et al. (2004) build a query predictor using support vector regression, by training classifiers with features such as document frequencies and query term frequencies. In the conducted experiments they obtained a small correlation between predicted and actual query performances. He and Ounis (2004) propose the notion of **query scope** as a measure of the specificity of a query, which is quantified as the percentage of documents that contain at least one query term in the collection, i.e., $\log(N_Q/N)$, being N_Q the number of documents containing at least one of the query terms, and N the total number of documents in the collection. Query scope has shown to be effective in inferring query performance for short queries in ad hoc text retrieval, but very sensitive to the query length (Macdonald et al., 2005).

5.2.2 Post-retrieval predictors

Post-retrieval performance predictors make use of the retrieved results, in contrast to pre-retrieval predictions. Furthermore, computational efficiency is usually a problem for many of these techniques, which is balanced by better prediction accuracy. In the following we present the most representative approaches of each of the different sub-categories described in Section 5.1.2: clarity, robustness, and score analysis.

Clarity-based predictors

Cronen-Townsend et al. (2002) define **query clarity** as a degree of (the lack of) query ambiguity. Because of the particular importance and use of this predictor in the findings of this thesis, we shall devote a whole section (Section 5.3) for a thorough description and discussion about it. It is worth noting that the concept of query clarity has inspired a number of similar techniques. Amati et al. (2004) propose the **query difficulty** predictor to estimate query performance. In that work query performance is captured by the notion of the amount of information ($Info_{DFR}$) gained after the ranking. If there is a significant divergence in the query-term frequencies before and after the retrieval, then it is assumed that the divergence is caused by a query that is easy to respond to. $Info_{DFR}$ showed a significant correlation with average precision, but did not show any correlation between this predictor and the effectiveness of query expansion. The authors hence concluded that although the performance gains by query expansion in general increase as query difficulty decreases, very easy queries hurt the overall performance.

Adaptations of the query clarity predictor such as the one proposed in (Hauff et al., 2008b) will be discussed later in Section 5.3. Additionally, apart from the Kullback-Leibler divergence, the Jensen-Shannon Divergence on the retrieved document set and the collection also obtains a significant correlation between average precision and the distance measured (Carmel et al., 2006).

Robustness-based predictors

More recently, a related concept has been coined: **ranking robustness** (Zhou and Croft, 2006). It refers to a property of a ranked list of documents that indicates how stable a ranking is in the presence of *uncertainty* in its documents. The idea of predicting retrieval performance by measuring ranking robustness is inspired by a general observation in noisy data retrieval. The observation is that the degree of ranking robustness against noise is positively correlated with retrieval performance. This is because the authors assumed that regular documents also contain *noise*, if noise is interpreted as uncertainty. The robustness score performs better than, or at least as well as, the clarity score.

Regarding document and query perturbation, Vinay et al. (2006) propose four metrics to capture the geometry of the top retrieved documents for prediction: the **clustering tendency** as measured by the Cox-Lewis statistic, the sensitivity to **document perturbation**, the sensitivity to **query perturbation**, and the **local intrinsic dimensionality**. The most effective metric was the sensitivity to document perturbation, which is similar to the robustness score. Document perturbation, however, did not perform well for short queries, for which prediction accuracy dropped considerably when alternative state-of-the-art retrieval techniques (such as BM25 or a language modelling approach) were used instead of the TF-IDF weighting (Zhou, 2007).

Several predictors have been defined based on the concept of query perturbation. Zhou and Croft (2007) propose two performance predictors are defined based on this concept specifically oriented for Web search. First, the **Weighted Information Gain** predictor measures the amount of information gained about the quality of retrieved results (in response to a query) from an imaginary state that only an average document (represented by the whole collection) is retrieved to a posterior state that the actual search results are observed. This predictor was very efficient and showed better accuracy than clarity scores. The second predictor proposed in that work is the **Query Feedback**, which measures the degree of corruption that results from transforming Q to L (the output of the channel when the retrieval system is seen as a noisy channel, i.e., the ranked list of documents returned by the system). The authors designed a decoder that can accurately translate L back into a new query Q', whereupon the similarity between the original query Q and the new query Q' is taken as a performance predictor, since the authors interpreted the evaluation of the quality of

the channel as the problem of predicting retrieval effectiveness. The computation of this predictor requires a higher computational cost than the previous one, being a major drawback of this technique.

Additionally, in (Diaz and Jones, 2004) and (Jones and Diaz, 2007) the authors exploited **temporal features** (time stamps) of the document retrieved by the query. They found that although temporal features are not highly correlated to performance, using them together with clarity scores improves prediction accuracy. Similarly, Diaz (2007) proposes to use the spatial autocorrelation as a metric to measure spatial similarities between documents in an embedded space, by computing the Moran's coefficient over the normalised scores of the documents. This predictor obtained good correlations results, although the author explicitly avoided collections such as question-answering and novelty related under the hypothesis that documents with high topical similarity should have correlated scores and, thus, in those collections the predictor would not work properly.

Other predictor was proposed in (Jensen et al., 2005), where visual features such as document titles and snippets are used from a surrogate document representation of retrieved documents. Such predictor was trained on a regression model with manually labelled queries to predict precision at the top 10 documents in Web search. The authors reported moderate correlation with respect to precision.

In (Yom-Tov et al., 2005a) two additional performance predictors are proposed. The first predictor builds a **histogram of the overlaps** between the results of each sub-query that agree with the full query. The second predictor is similar to the first one, but is based on a decision tree (Duda et al., 2001), which again uses overlaps between each sub-query and the full query. The authors apply these predictors to selective query expansion detecting missing content, and distributed information retrieval, where a search engine has to merge ranks obtained from different datasets. Empirical results showed that the quality of the prediction strongly depends on the query length.

The following predictors have been based on the cohesion of the retrieved documents. Kwok et al. (2004) propose predicting query performance by analysing similarities among retrieved documents. The main hypothesis of this approach is that relevant documents are similar to each other. Thus, if relevant documents are retrieved at the top ranking positions, the similarity between top documents should be high. The preliminary results, however, were inconclusive since negligible correlations were obtained. A similar approach is proposed in (Grivolla et al., 2005), where the entropy and pairwise similarity among top results are investigated. First, the entropy of the set of the K top-ranked documents for a query was computed. In this case it was assumed that the entropy should be higher when the performance for a given query is bad. Second, the mean cosine similarity between documents was proposed, using the base form of TF-IDF term weighting to define the document vec-

tors. Correlation between average precision and the proposed predictors was not consistent along the different systems used in the experiment, although the predictors could still be useful for performance prediction, especially when used in combination.

Predictors based on score analysis

Finally, the last family of post-performance predictors analyses the score distributions of the results for each query. We have to note that the Weighted Information Gain predictor (Zhou and Croft, 2007) explained above is sometimes categorised into this group. In the following we present other predictors where the retrieved scores are explicit in the predictor computation.

For instance, the **Normalised Query Commitment** (NQC) predictor (Shtok et al., 2009) measures the standard deviation of the retrieval scores, and applies a normalisation factor based on the score of the whole collection:

$$\mathrm{NQC}(q) = \frac{\sqrt{1/|D_q| \sum_{d \in D_q} \left(s(d) - \mu_q\right)^2}}{|s(\mathcal{C})|} \tag{5.6}$$

where μ_q is the mean score of results in D_q (the retrieved set of documents for a query q). This predictor measures the divergence of results from their centroid, a "pseudo non-relevant document" that exhibits a relatively high query similarity (Carmel and Yom-Tov, 2010).

The **utility estimation framework** (UEF) was proposed in (Shtok et al., 2010) to estimate the utility of the retrieved ranking. In this framework three methods have to be specified to derive a predictor: a sampling technique for the document sets, a representativeness measure for relevance-model estimates, and a measure of similarity between ranked lists. Other authors have proposed approaches where standard deviation does not need to be computed for all the document scores in the retrieved results. Pérez-Iglesias and Araujo (2009) use a cutoff to decide how many documents are considered in the standard deviation computation. Moreover, Cummins et al. (2011) use different strategies to automatically select such cutoff.

Recently, Cummins (2012) has used Monte Carlo simulations to understand the correlations between average precision and the standard deviation of the scores in the head of a ranked list. The author found that the standard deviation of the list is positively correlated with the mean score of relevant documents, which in turn is positively correlated with average precision.

5.3 Clarity score

Cronen-Townsend et al. (2002) defined clarity score for Web retrieval as a measure of the lack of ambiguity of a particular query. More recently, it has been observed that this predictor also quantifies the diversity of the result list (Hummel et al., 2012). In this section we provide a deep analysis of this performance predictor since we shall use it along the rest of this thesis. We also describe examples and adaptations of the clarity score.

5.3.1 Definition of the clarity score

The clarity score predictor is defined as a Kullback-Leibler divergence between the query and the collection language model. It estimates the coherence of a collection with respect to a query q in the following way, given the vocabulary $\mathcal{V}$ and a subset of the document collection R_q consisting of those documents that contain at least one query term:

$$\text{clarity}(q) = \sum_{w \in \mathcal{V}} p(w|q) \log_2 \frac{p(w|q)}{\text{p}(w|\mathcal{C})} \tag{5.7}$$

$$p(d|q) = p(q|d)p(d)$$

$$p(q|d) = \prod_{w_q \in q} p(w_q|d)$$

$$p(w|q) = \sum_{d \in R_q} p(w|d)p(d|q)$$

$$p(w|d) = \lambda p_{\text{ml}}(w|d) + \ (1-\lambda)p_{\text{c}}(w)$$

The clarity value can thus be reduced to an estimation of the prior $\text{p}(w|\mathcal{C})$ (collection language model), and the posterior $p(w|q)$ of the query terms w (query language model) using $p(w|d)$ over the documents $d \in R_q$ and based on term frequencies and smoothing. It should be emphasised that if the set R_q is chosen as the whole collection $\mathcal{C}$, then this technique could be classified as a pre-retrieval performance predictor, since no information about the retrieval would be used. The importance of the size of the relevance set R_q (or number of feedback documents) has been studied in (Hauff et al., 2008b), where an adaptation of the predictor was proposed in order to automatically set the number of documents to consider.

As first published in (Cronen-Townsend et al., 2002) and (Cronen-Townsend et al., 2006), query ambiguity is defined as "the degree to which a query retrieves documents in the given collection with similar word usage." Cronen-Townsend and

colleagues found that queries whose highly ranked documents are a mix of documents from disparate topics receive lower scores than if they result in a topically-coherent retrieved set, and reported a strong correlation between the clarity score and the performance of a query. Because of that, the clarity score method has been widely used in the area for query performance prediction.

Some applications and adaptations of the clarity score metric include query expansion (anticipating poorly performing queries that should not be expanded), improving performance in the link detection task (more specifically, in topic detection and tracking by modifying the measure of similarity of two documents) (Lavrenko et al., 2002), and document segmentation (Brants et al., 2002). More applications can be found in Section 5.3.3.

Zhou (2007) provides a complementary formulation of the clarity score by re-writing the formulation used above as follows:

$$\text{clarity}(q) = \sum_{w \in \mathcal{V}} \sum_{d \in R_q} p(w|d)p(d|q) \log \frac{\sum_{d \in R_q} p(w|d)p(d|q)}{p(w|\mathcal{C})} \tag{5.8}$$

In this way, Zhou emphasises, among other issues, the differences between the query clarity and the Weighted Information Gain predictor. Indeed, the author proposes the following generalisation of both formulations (for WIG and clarity). Specifically, the clarity formulation presented in Equations (5.7) and (5.8) is unified as follows:

$$\text{score}(q, \mathcal{C}, R) = \sum_{\xi \in T} \sum_{d \in R_q} \text{weight}(\xi, d) \log \frac{p(\xi, d)}{p(\xi, \mathcal{C})} \tag{5.9}$$

where T is a feature space, and R_q is a (ranked) document list. Besides this, $d \in R_q \subseteq \mathcal{C}$ must be comparable somehow with elements $\xi \in T$, in order to make sensible functions $\text{weight}(\xi, d)$ and $p(\xi, d)$. In this context, the query clarity as defined in (Cronen-Townsend et al., 2002) is an instantiation of Equation (5.9) where the following three aspects are considered:

- The feature space T is the whole vocabulary, consisting of single terms.
- The weight function is defined as $\text{weight}(\xi, d) = p(w|d)p(d|q)$.
- The function $p(\xi, d)$ is defined as $\sum_{d \in R_q} p(w|d)p(d|q)$, that is, it uses a document model averaged over all documents in the ranked list.

These observations help to discriminate between the underlying models used by these two predictors. In particular, for the query clarity, they also contribute to capture not so obvious divergences between a query and the collection, as we shall see in the next section.

train (0.33)	train dog (0.65)	obedience train dog (2.43)
		railroad train dog (0.67)
	railroad train (0.73)	railroad train caboose (1.46)

Table 5.2. Examples of clarity scores for related queries.

5.3.2 Interpreting clarity score in Information Retrieval

Aiming to better understand how the clarity score predictor behaves in Information Retrieval, and to what extent it is able to capture the difficulty or ambiguity of queries, in this section we summarise examples reported in the literature that let a clear interpretation of the predictor's values.

In a seminal paper (Cronen-Townsend et al., 2002) Cronen-Townsend and colleagues present the example shown in Table 5.2, which provides the clarity scores of a number of related queries that share some of their terms. These queries are related to each other in the sense that a particular query is formed by extending other query with an additional term, starting with an initial query formed by a single term, 'train' in the example. According to the queries of the table, we can observe that the term 'train' has different meanings for the largest queries; it refers to 'teach' in the query 'train dog', to the 'locomotive vehicle' in the query 'railroad train', and can refer to any of both meanings in the query 'railroad train dog.' The clarity scores capture the ambiguity of the queries (due to their different meanings for the term 'train'), independently from their length. In fact, the middle rightmost query 'railroad train dog' receives the lowest clarity score, corresponding to the most ambiguous query where the two considered meanings of 'train' are involved.

In the same paper, Cronen-Townsend and colleagues present the distribution of the language models for two queries, a clear query and a vague query (see Figure 2 in (Cronen-Townsend et al., 2002)). Each distribution is presented by plotting $p(w|q)\log_2 p(w|q)/\mathrm{p}(w|\mathcal{C})$ against the query terms w. The authors show that the distribution of the values of this function for the clear query dominates the distribution of the values of the vague query. This makes sense since the clarity score is computed by summing the probability values in the distribution of every term in the collection. Additionally, the authors show that the clear query presents spikes in its query language model when $p(w|q)$ is plotted against the terms, and compared with the collection probability $p(w|\mathcal{C})$. Hence, some of the terms with high contribution from the query language model (i.e., with high $p(w|q)$ values) obtain low collection

probabilities ($p(w|\mathcal{C})$), thus evidencing a query that is different to the collection in its term usage (i.e., it is a non ambiguous query).

The above examples involve the (implicit) assumption known as *homogeneity assumption*, which specifies that the clarity score is higher if the documents in the considered collection are topically homogeneous. Hauff (2010) analyses the sensitivity of results with respect to that assumption. Specifically, the author computes the clarity score for three different ranked document lists: the relevant documents for a query, a non-relevant random sample, and a collection-wide random sample. The difference between the last two lists is that the second one is derived from documents judged as non-relevant, whereas the third one could contain any document in which at least one query term. Hauff shows how the clarity score is different depending on the origin of ranked document list, leading to a higher (lower) score by using relevant (non-relevant) documents for such list. However, we have to note that, as stated by Hauff, the quality in the separation of the clarity scores computed by each document list is different depending on the utilised dataset and queries.

The clarity score has been analysed in detail in Information Retrieval, mainly because its predictive power is superior to other performance predictors (in fact, it is one of the best performing post-retrieval predictors according to the overview presented in (Hauff, 2010)), but also because it provides interpretable results and high explanatory power in different IR processes, as we shall describe in the next section. Apart from that, the interest in this predictor is clear because of its probabilistic formulation and tight relationship with Language Models (Ponte and Croft, 1998).

5.3.3 Adaptations and applications of the clarity score

Cronen-Townsend and colleagues showed in (Cronen-Townsend et al., 2002) that clarity is correlated with performance, proving that the result quality is largely influenced by the amount of uncertainty involved in the inputs a system takes. In this sense, queries whose highly ranked documents belong to diverse topics receive lower scores than queries for which a topically-coherent result set is retrieved. Several authors have exploited the clarity score functionality and predictive capabilities (Buckley, 2004; Townsend et al., 2004; Dang et al., 2010), supporting its effectiveness in terms of performance prediction and high degree of adaptation. For instance, the predictor has been used for personalisation (Teevan et al., 2008) because of its proven capability of predicting ambiguity. In that paper the authors use more or less personalisation depending on the predicted ambiguity.

One of the first variants proposed in the area is the simplified clarity score proposed in (He and Ounis, 2004), presented in Section 5.2.1. In that paper He and Ouni changed the estimations of the posterior $p(w|q)$ to simple maximum likelihood estimators. Hauff et al. (2008b) proposed the Improved Clarity – called Adapted Clarity in (Hauff, 2010) –, in which the number of feedback documents

(R_q) is set automatically, and the term selection is made based on the frequency of the terms in the collection to minimise the contribution of terms with a high document frequency in the collection.

An alternative application of the clarity score is presented in (Allan and Raghavan, 2002), where the score obtained for the original set of documents returned by a query is compared against that obtained for a modified query, which was presumed to be more focused than the original one. Similarly, in (Buckley, 2004) Buckley uses the clarity score to measure the stability of the document rankings and compare it against a measure that uses the Mean Average Precision of each ranking (AnchorMap).

In (Sun and Bhowmick, 2009), Sun and Bhowmick adapted the concept of query clarity to image tagging, where a tag is visually representative if all the images annotated with that particular tag are visually similar to each other. In previous work (Sun and Datta, 2009) Sun and Datta proposed a similar concept, but in the context of blogging: a tag would receive a high clarity score if all blog posts annotated by the tag are topically cohesive.

Finally, an extension of the Kullback-Leibler divergence was proposed in (Aslam and Pavlu, 2007), where the Jensen-Shannon divergence was used instead. This distance is defined as the average of the Kullback-Leibler divergences of each distribution with respect to the average (or centroid) distribution. In this way, it is possible to compute the divergence between more than two distributions. Besides, the Jensen-Shannon divergence is symmetric, in contrast to the divergence used in the clarity score, and thus, a metric can be derived from it (Endres and Schindelin, 2003).

5.4 Evaluating performance predictors

In this section we describe the approaches proposed in the literature to evaluate the predictive power of a performance predictor. We define the different functions used to compute the quality of the performance predictors, most of them based on well known correlation coefficients between the true query performance values, and the expected or predicted performance values.

5.4.1 Task definition

Based on the notation presented in Section 5.1, in the following we present different techniques and functions to assess the effectiveness of performance predictors. Once the retrieval quality has been assessed ($\mu(q)$), and the value of the performance predictor for each query is calculated ($\hat{\mu}(q)$, using the function γ), the predictor quality is computed by using a predictor quality assessment function f^{qual} that measures the agreement between the true values of performance and the estimations, that is:

$$\text{Quality}(\gamma) = f^{qual}(\{\mu(q_1), \cdots, \mu(q_n)\}, \{\hat{\mu}(q_1), \cdots, \hat{\mu}(q_n)\}) \quad (5.10)$$

True quality values for each query are typically obtained by computing the per-query performance of a selected retrieval method (Cronen-Townsend et al., 2002; Hauff et al., 2008a), or by averaging the values obtained by several engines (Mothe and Tanguy, 2005), in order to avoid biases towards a particular method. As we shall see in the next section, the function f^{qual} typically represents a correlation coefficient; however, different possibilities are available and may be more appropriate depending on the prediction task.

In fact, in (Hauff et al., 2009) three estimation tasks were considered, by discriminating the output of the predictor function $\hat{\mu}$. **Query difficulty** estimation could be defined as a classification task where $\hat{\mu} \rightarrow \{0,1\}$ indicates whether the query is estimated to perform well or poorly. The standard estimation of **query performance**, nonetheless, would be defined by a function $\hat{\mu} \rightarrow \mathbb{R}$, in order to provide a ranking of queries, where the highest score denotes the best performing query. Furthermore, as stated in (Hauff et al., 2009), this function by itself does not directly estimate the performance metric μ. In order to do that we need to have normalised scores, such that the range of $\hat{\mu}$ is compatible with that of the metric, which typically requires $\hat{\mu} \rightarrow [0,1]$. In this case, we would be considering the **normalised query performance** task.

The methodology described above is general enough to be applicable to any of these three tasks, but is clearly inspired by the second one, that is, the estimation of query performance and it can be easily applied also to third one (normalised performance prediction). Because of that, we describe next a recently proposed methodology more focused on the (binary) query classification task or query difficulty prediction described in (Pérez-Iglesias and Araujo, 2010).

Let us suppose that, instead of continuous values of the performance metric μ, we are interested in estimating *as accurately as possible* the different difficulty grades of the queries, that is, $\mu \rightarrow \{1, \cdots, k\}$, where k is the number of difficulty grades available. Obviously, the output of the predictor $\hat{\mu}$ also has to be grouped in one of the k classes. Typically, we would have $k = 3$, representing "Easy", "Average", and "Hard" queries, although a binary partition could also be acceptable. In these terms the performance prediction problem is stated as a classification problem, where the goal is to effectively predict the query class.

Furthermore, this technique lets set, at the quality computation step, whether we want to weight uniformly each of the k classes, or if we are more interested in only one of them, by building, for instance, a confusion matrix, and applying standard Machine Learning evaluation metrics to a subset of it. In the next section we describe the most popular techniques for doing this, along with a new metric introduced in (Pérez-Iglesias and Araujo, 2010) oriented to the problem of performance prediction.

5.4.2 Measuring the quality of the predictors

There are several methods for measuring the quality of the performance prediction function $\hat{\mu}$ defined in the previous section. In particular, the quality function f^{qual} may be able to capture linear relations, take into account the importance implied by the scores or the ordering given by each variable (true and estimated performance, i.e., μ and $\hat{\mu}$), and exploit the implicit partitions derived by the method.

The most commonly used quality function is correlation, which has been measured by three well-known metrics: Pearson's, Spearman's, and Kendall's correlation coefficients. **Pearson's r correlation** captures linear dependencies between the variables, whereas **Spearman's ρ** and **Kendall's τ** correlation coefficients are used in order to uncover non-linear relationships between the variables. They are generally computed as follows, although in special situations (in presence of ties, or when there are missing values in the data) alternative formulations may be used:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{5.11}$$

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d(x_i, y_i)^2}{n(n^2 - 1)} \tag{5.12}$$

$$\tau = 1 - \frac{4Q(x, y)}{n(n-1)} \tag{5.13}$$

where x and y represent the two variables of interest, $\bar{x}$ and $\bar{y}$ denote their means, $d(x_i, y_i)$ is the difference in ranks between x_i and y_i, and $Q(x, y)$ is the minimum number of swaps needed to convert the rank ordering of x to that of y. All these coefficients return values between -1 and $+1$, where -1 denotes a perfect anti-correlation, 0 denotes statistical independence, and $+1$ denotes perfect correlation.

It can be observed that Spearman's ρ computes a Pearson's r between the ranks induced by the scores of the variables. Moreover, Kendall's τ is the number of operations required to bring one list to the order of the other list using the *bubble sort* algorithm. Besides, although Spearman's and Kendall's correlations seem more general than Pearson's since they are able to capture non-parametric relations between the variables, we have to consider that distances between the scores are ignored in the rank-based coefficients, and thus, it is typically suggested to report one correlation coefficient of each type.

It is important to note that the number of points used to compute the correlation values affects the significance of the correlation results. The confidence test for a Pearson's r correlation, modeled as the t-value of a t-distribution (assuming normality) with $N - 2$ degrees of freedom (being N the size of the sample), is defined by the following equation (Snedecor and Cochran, 1989):

p-value	N		
	50	100	500
$p < 0.05$	1.677	1.661	1.648
$p < 0.01$	2.407	2.365	2.334

Pearson's r value	N		
	50	100	500
0.1	0.696	0.995	**2.243**
0.2	1.414	**2.021**	**4.555**
0.3	**2.179**	**3.113**	**7.018**
0.4	**3.024**	**4.320**	**9.739**

Table 5.3. Left: minimum t-value for obtaining a significant value with different sample sizes (N). Right: t-value for a given Pearson's correlation value and N points. In bold when the correlation is significative for $p < 0.05$, and underlined for $p < 0.01$.

$$t = r\sqrt{\frac{N-2}{1-r^2}} \tag{5.14}$$

The t-value therefore depends on the size of the sample, and thus, the significance of a Pearson's correlation value r may change depending on the number of test queries. In particular, for small samples, we may eventually obtain strong but non-significant correlations; whereas for large samples, on the other hand, we may obtain significant differences, even though the strength of the correlation values may be lower. The above also applies to the correlations computed using the Spearman's coefficient, but only under the null hypothesis or large sample sizes (greater than 100) (Snedecor and Cochran, 1989; Zar, 1972). For Kendall's correlation, the confidence test can be computed using an exact algorithm when there are no ties based on a power series expansion in N^{-1}, depending again, thus, on the sample size (Best and Gipps, 1974).

Table 5.3 shows the minimum t-value for obtaining a significant value with different sample sizes and p-values, along with the t-value computed using Equation (5.14) for different correlation values and sample sizes. In the table we can observe that the same correlation value may be significant or not depending on the size of the sample, for instance, with 50 queries, observations are significant with $p < 0.05$ for correlation values equal or above 0.3, whereas for 100 queries it is enough to obtain Pearson's correlation values of 0.2. This observation is related to the one presented in (Hauff et al., 2009), where Hauff and colleagues compared the confidence intervals of the three correlation coefficients described before, and observed how, due to the small query set sizes, most of the predictors analysed (pre-retrieval approaches such as clarity, IDF-based, and PMI) presented no significant differences, despite having very different values. In particular, this generated a subset of the analysed predictors that were not statistically different to the best performing predictor reported, and thus, any of the predictors in subset may be used in a later application since they obtain statistically similar (strictly speaking, not statistically different) correlations.

Furthermore, in the same paper, Hauff and colleagues proposed to use the **Root Mean Squared Error** (RMSE) as a quality function. The rationale behind this is that the RMSE squared is the function being minimised when performing a linear regression, and thus, it should also be able to capture the (linear) relation between the variables. In fact, there is a close relation between the RMSE and the Pearson's r coefficient, by means of the residual sum of squares (Carmel and Yom-Tov, 2010):

$$r^2 = 1 - \frac{SS_{err}}{SS_{tot}} = 1 - \frac{\sum_{i=1}^{n}(x_i - y_i)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{5.15}$$

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n-1}} = \sqrt{\frac{SS_{err}}{n-1}} \tag{5.16}$$

Additional extensions to these correlation coefficients have been proposed. Most of these extensions have been focused on incorporating weights in the computation of the correlation (Melucci, 2009; Yilmaz et al., 2008). However, despite these metrics have an evident potential in the performance prediction area, to the best of our knowledge there is no work using them in order to evaluate the quality of the predictors (Pérez Iglesias, 2012).

Finally, a different family of quality functions can be considered in the query difficulty task, that is, when the performance prediction is cast as a classification problem. These techniques are based on the accuracy of the classification provided by the performance predictor, and thus, classic Machine Learning techniques could be used. In (Pérez-Iglesias and Araujo, 2010), Pérez-Iglesias and Araujo propose to use the **F-measure**:

$$F = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{5.17}$$

Additionally, in the same paper, Pérez-Iglesias and Araujo introduced a new metric (**distance based error measure**, or DBEM) along with a methodology that is focused on the misclassified difficulty classes between the predictor and the true classes. With this goal in mind, the authors apply a clustering algorithm to both the performance metric values and their estimations, aimed to minimise the distance between elements in the same group, and maximise the distance between elements in different groups. Specifically, Pérez-Iglesias and Araujo used the k-means algorithm, setting the value of k to the number of relevance grades, $k = 3$ in their paper. The metric DBEM is defined as follows:

$$\text{DBEM} = \frac{\sum_i^n \text{dist}\big(c(x_i), c(y_i)\big)}{\sum_i^n \max_j \text{dist}\Big(c(x_i), c\big(x_j\big)\Big)}$$
$$\text{dist}\big(c_i, c_j\big) = \|i - j\|, 0 < i, j \leq k \tag{5.18}$$

where $c(x)$ is the function which assigns the proper class or partition to a given score x, according to the clustering algorithm. This metric captures the distance between every partition, normalised by the maximum possible distance. In this case, lower distances imply a better predictor quality.

5.5 Summary

Improvement of the predictive capabilities to infer the performance or difficulty of a query is consolidated as a major research topic in Information Retrieval, where it has been mostly applied to ad-hoc retrieval. Several performance predictors have been defined based on many different information sources, demonstrating the usefulness of such predictors in different tasks, mainly for query expansion, but also for rank fusion, distributed information retrieval, and text segmentation.

Some issues are, however, still open in the field, mostly regarding the evaluation of performance prediction. Performance prediction methods have been usually evaluated on traditional TREC document collections, which typically consist of no more than one million relatively homogenous newswire articles, and few research work has exploited these techniques with larger datasets; see, e.g. (Carmel et al., 2006; Zhou, 2007; Hauff, 2010) for some exceptions. Furthermore, reported correlation coefficient values have been typically computed using a small number of points (e.g. 50 queries for standard tracks in TREC), not always providing enough confidence to derive conclusions. And more importantly, how predictors have to be evaluated and which metric has to be used are still open research questions, that have generated some fruitful discussion in recent publications (Hauff, 2010; Pérez Iglesias, 2012), although a definitive answer has not been obtained yet.

We may presume that in the future other information retrieval applications may benefit from the framework derived by these techniques, and may develop tailored performance predictors by using purpose-designed performance metrics and evaluation methodologies, such as the recently developed concept of document difficulty in (Alvarez et al., 2012). This thesis is an example of such an application in the Recommender Systems field. More specifically, as we shall see in the next chapter, we translate the problem of performance prediction to the Recommender Systems area, where it has been barely studied. We focus our research on the query clarity predictor as a basis for the recommendation performance predictors, although additional techniques could be used, as we shall also present in Chapter 6. Finally, among the array of evaluation strategies presented above, we have decided to use correlations since it is the most common one in the literature, and provides a fair notion about the interpretability of the results.

Chapter 6

Performance prediction in recommender systems

In this chapter, we state and address the recommendation performance prediction problem, proposing and evaluating different prediction schemes. After laying out a formal frame for the problem, we start by researching the adaptation of principles and prediction techniques that have been proposed and developed in ad-hoc Information Retrieval. More specifically, we draw from the notion of query clarity as a basis for finding suitable performance predictors that provide a well grounded theoretical formalisation. In analogy to query clarity, we hypothesise that the amount of uncertainty involved in user and item data (reflecting ambiguity in user's tastes and item popularity patterns) may also correlate with the accuracy of the system's recommendations. This uncertainty can be captured as the clarity of users and/or the clarity of items by an adaptation of the query clarity formulation. This adaptation, however, is not straightforward, as we shall describe. Besides the approaches elaborating on the notion of clarity, we propose new predictors based on theories and models from Information Theory and Social Graph Theory.

In Section 6.1 we formulate the research problem we aim to address. Next, in Sections 6.2, 6.3, and 6.4 we propose several performance predictors for recommender systems, some of them based on the clarity score, information theoretical related concepts – such as entropy –, and graph-based metrics. The proposed predictors are defined upon three different spaces, namely ratings, logs, and social networks. Moreover, we also provide specific correlations of the described predictors in Section 6.5 in order to show their predictive power under different conditions along with a discussion of the results. Finally, in Section 6.6 we provide some conclusions.

6.1 Research problem

Performance prediction finds a special motivation in recommender systems. Contrary to query-based information retrieval, as far as the initiative relies on the system, a performance prediction approach may provide a basis to decide producing recommendations or holding them back, depending on the expected level of performance on a per case basis, delivering only the sufficiently reliable cases. On the other hand, recommenders based on a single algorithm are not competitive in practice, and real applications heavily rely on hybridisations and ensembles of algorithms.

The capability to foresee which algorithm can perform better in different circumstances can therefore be envisioned as a good approach to enhance the performance of the combination of algorithms by dynamically adjusting the reliance on each subsystem. Furthermore, it is well-known in the recommender systems field that the performance of individual recommendation methods is highly sensitive to different conditions, such as data sparsity, quality and reliability, which are subject to an ample dynamic variability in real settings. Hence, being able to estimate in advance which recommenders are likely to provide the best output in a particular situation opens up an important window for performance enhancement. Alternatively, estimating which users of a system are likely to receive worse recommendations allows for modifications in the recommendation algorithms to predict this situation, and react in advance.

The problem of performance prediction has been however barely addressed in the Recommender Systems field. The issue has been nonetheless mentioned in the literature – evidencing the relevance of the problem – and is in some way often implicitly addressed by means of ad hoc heuristic tweaks such as significance weighting in nearest neighbour recommenders (Herlocker et al., 1999) and confidence scores (Wang et al., 2008a), along with additional computations (mainly normalisations) which are introduced into the recommendation methods aimed to better estimate the predicted ratings.

In the recommendation context, the problem of performance prediction can be stated as follows. We define a performance predictor as a function that takes a certain input, and returns a real value that correlates with some utility dimension of a recommender system. This is an instantiation of the problem presented in Section 5.1 but in the recommendation setting. For such purpose, we first specify more precisely what the input space of predictors consists of, and how the predictor input and output relate to the data involved in recommendation. Thus, a utility predictor handles the following information:

Input variables

- The specific configuration of the recommender system. For instance, for a nearest neighbour recommender input parameters could be the neighbour map (that assigns a set of neighbours to each user) and a user similarity metric.
- Any input of the recommender, such as the active user and the active item.
- Background/context information: any known user, item, and user-item interaction data, such as user ratings, user features, item features, social network information, data timestamps, etc. We have to note that, even though the predictor will generally use this type information, we consider it as implicit input and do not include it explicitly in our notation to avoid making it needlessly cumbersome.

Output variable

- A value in $\mathbb{R}$.

A predictor is thus a function $\gamma: R \times \mathcal{U} \times \mathcal{I} \to \mathbb{R}$ (R being the set of all recommenders) that estimates the performance of the system, possibly using additional information available in the background. A predictor can be independent from some of these inputs, which would be then omitted in the previous notation. For instance, in this chapter we shall present predictors of the form $\gamma: \mathcal{U} \to \mathbb{R}$ and $\gamma: \mathcal{I} \to \mathbb{R}$. Additionally, a predictor may assume a specific parameterised recommender algorithm family (e.g. nearest neighbour collaborative filtering), and needs some element of its configuration as input. It may also happen that a predictor does not make any assumption on the recommender – it does not depend on it – but still the predictor works well only for certain types of recommenders. It would be syntactically possible and correct to apply the predictor with other recommenders, although it may work badly. In general, what it means for a predictor to work "well" may depend on the application, but we generally assume it can be evaluated in terms of its correlation to some utility dimension of recommendations, such as an accuracy metric (RMSE, precision, nDCG) or alternative metrics such as novelty, diversity, etc.

If a recommender system can be decomposed into its internal configuration, then a predictor can directly take as input the components of the recommender configuration. For instance, neighbourhood-based collaborative filtering recommenders can be represented in $R \equiv \mathcal{E} \times \mathcal{N} \times \mathcal{S}$, where $\mathcal{E}: \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}^+$ is a preference estimation function (based on k similarity values between the target user and her neighbours, and k neighbours' ratings on the target item), $\mathcal{N}: \mathcal{U} \to P(\mathcal{U})$ is a neighbourhood assignment map, and $\mathcal{S}$ is a similarity metric. Upon such a model, we would have $\gamma: \mathcal{E} \times \mathcal{N} \times \mathcal{S} \times \mathcal{U} \times \mathcal{I} \to \mathbb{R}$.

We may also constrain some inputs to a relevant condition they should meet. For instance, we could limit ourselves to a neighbourhood map that considers a user v as a candidate neighbour. In that case, this map can be essentially represented by v, and then we would have $\gamma: \mathcal{E} \times \mathcal{U} \times \mathcal{S} \times \mathcal{U} \times \mathcal{I} \rightarrow \mathbb{R}$ (note that the first $\mathcal{U}$ in the Cartesian product stands for neighbour users, and the second $\mathcal{U}$ for target users).

It is important to note that when the predictor takes as input some of the inputs of the recommender, namely the active user and/or the active item, then the predictor's correlation with the recommender's utility must be measured on a per-input basis. For instance, if the predictor just takes users as input arguments, it should correlate with the average utility per user.

Moreover, predictors can also be used to enhance hybrid recommenders by favouring strategies that are predicted to produce better results. This can be done by relating activation switches in the recommenders to predictor values, so that one recommender or the others are activated or favoured depending on the predictor's estimation.

The way in which these activation switches are related to predictors is typically application-dependent. For instance, in ensemble recommenders consisting of a unique (Boolean) selection among a set of recommenders, the selection/discarding of recommenders can be a binary function of a predictor for each recommender. If the ensemble consists of a linear combination of recommenders, the weights in the combination can also be a function of the predictors. In neighbourhood-based collaborative filtering, activation switches can be the weights of neighbours in the prediction of user ratings. Indeed, relating predictor values to activation switches is a non-trivial problem and generally requires some research on itself.

Based on all the above mentioned issues, the general research problem we address consists of a) finding effective predictors of recommendation utility, and b) identifying and testing useful applications for the found predictors. In the reminder of this chapter we propose different predictors of recommendation utility using different types of input, namely ratings, logs, and social information. In Chapters 7 and 8 we shall exploit and evaluate such predictors in two applications: dynamic hybrid recommendation, and dynamic neighbour weighting in collaborative filtering.

6.2 Clarity for preference data: adaptations of query clarity

In this thesis, we propose different adaptations for the concept of query clarity to recommender systems. First, we deal with the definition of user clarity when rating-based preference data is available, where alternative ground models are proposed, depending on which random variables want to be considered in the computation of

the user clarity. Then, we define the concept of user clarity for log-based preference data. Additionally, for ratings we also define the concept of item clarity.

Now we propose a fairly general adaptation of query clarity, which may be instantiated into different schemes, depending on the input spaces considered. At an abstract level, we consider an adaptation that equates users in the recommendation domain to queries in the search domain, as the corresponding available representations of user needs in the respective domains. This adaptation results in the following formulation for **user clarity**:

$$\text{clarity}(u) = \sum_{x \in \mathcal{X}} p(x|u) \log_2 \frac{p(x|u)}{p(x)} \tag{6.1}$$

As we can observe, the clarity formulation strongly depends on a "vocabulary" space $\mathcal{X}$, which further constrains the user-conditioned model (or user model for short) $p(x|u)$, and the background probability $p(x)$. In ad-hoc information retrieval, this space is typically the space of words, and the query language model is a probability distribution over words (Cronen-Townsend et al., 2002). In recommender systems, however, we may have different interpretations, and thus, different formulations for such a probabilistic framework, as we shall show. In all cases, we will need to model and estimate two probability distributions: first, the probability that some event (depending on the current probability space $\mathcal{X}$) is generated by the user language model (user model); and second, the prior probability of generating that event (background model).

Under this formulation, user clarity is in fact the difference (Kullback-Leibler divergence) between a user model and a background model. The use of user and background distributions as a basis to predict recommendation performance lies on the hypothesis that a user probability model being close to the background (or collection) model is a sign of ambiguity or vagueness in the evidence of user needs, since the generative probabilities for a particular user are difficult to single out from the model of the collection as a whole. In Information Retrieval, this fact is interpreted as a query for which the relevant documents are a mix of articles about different topics (Cronen-Townsend et al., 2002).

As an additional step, we generalise the adaptation stated in Equation (6.1) to allow for different reference probability models parameterised by a generic variable θ.

$$\text{clarity}(u) = \mathbb{E}_\theta \left[\sum_{x \in \mathcal{X}} p(x|u,\theta) \log_2 \frac{p(x|u,\theta)}{p(x|\theta)} \right] \tag{6.2}$$

This generalisation will allow for the development of further varieties of the clarity scheme, and simplifies to Equation (6.1) whenever we implicitly consider a fixed θ, as we shall see next. Equivalently, the variable θ may be integrated in both user and background models by exploiting a multidimensional vocabulary space:

$$\text{clarity}(u) = \sum_{x \in \mathcal{X}, \theta \in \Theta} p(x, \theta|u) \log_2 \frac{p(x, \theta|u)}{p(x, \theta)} \quad (6.2b)$$

It is easy to see that Equations (6.2) and (6.2b) are fully equivalent, and thus allow two interpretations for the same magnitude.

As stated in (Cronen-Townsend et al., 2002), language models capture statistical aspects of the generation of language. Therefore, if we use different vocabularies, we may capture different aspects of the user. The probabilistic relations between the variables involved in Equation (6.2) also depend on the nature of the data, and the different possible generative models induced by the recorded observations of user-item interactions (the input to a recommender system). In this thesis we consider two types of interaction data records: users-rating-items (where the atomic event is a user rating an item with a value), and users "consuming" items (a user accesses an item at some time instant). The first type fits a dataset such as MovieLens and CAMRa, and the second fits well Last.fm data – the datasets on which we shall test the methods to be developed here. Across these two types, in our research we explore mainly three vocabulary spaces for $\mathcal{X}$: ratings, items, and time. Each of the vocabulary spaces induces its own user-specific interpretation, as we shall see. As for the optional contextual parameter θ, we shall consider here only the space of items ranging over the set of items – thus fully leveraging the triadic nature of the user-item-rating and user-item-time spaces. The scheme is however open to the exploration of further possibilities, as is the vocabulary space itself, beyond the options researched here.

In the following sections we thus explore several alternatives for rating-based and log-based data spaces (and their induced generative models).

6.2.1 Rating-based clarity

As just mentioned, in the rating space, we consider a set of user-item-rating tuples, where each user-item pair appears in a unique tuple (i.e., users only rate items once). We consider two possible vocabulary spaces: items and ratings, and two context alternatives: items (which make only sense in the rating vocabulary) and none. The resulting clarity schemes are summarised in Table 6.1, and have each their own interpretation.

The rating-based clarity model captures how differently a user uses rating values (regardless of the items the values are assigned to) with respect to the rest of users in the community. The item-based clarity takes into account which items have been rated by a user, and therefore, whether she rates (regardless of the rating value) the most rated items in the system or not. Finally, the item-and-rating-based clarity computes how likely a user would rate each item with some particular rating value, and compares that likelihood with the probability that the item is rated with some particular rating value. In this sense, the item-based user model makes the assumption that some items are more likely to be generated for some users than for others de-

User clarity	Vocabulary $\mathcal{X}$ / Context θ	User model	Background model	Formulation
Rating-based	Ratings / None	$p(r\|u)$	$p_c(r)$	$\sum_r p(r\|u)\log_2\frac{p(r\|u)}{p(r)}$
Item-based	Items / None	$p(i\|u)$	$p_c(i)$	$\sum_i p(i\|u)\log_2\frac{p(i\|u)}{p(i)}$
Item-and-rating-based	Ratings / Items	$p(r\|u,i)$	$p_{ml}(r\|i)$	$\sum_{r,i} p(i)p(r\|u,i)\log_2\frac{p(r\|u,i)}{p(r\|i)}$

Table 6.1. Three possible user clarity formulations, depending on the interpretation of the vocabulary and context spaces.

pending on their previous preferences. The rating-based model, on the other hand, captures the likelihood of a particular rating value being assigned by a user, which is an event not as sparse as the previous one, with a larger number of observations. Finally, the item-and-rating-based model is a combination of the two previous models into a unified model incorporating items and ratings. As we mentioned before, this could be made more explicit by considering the user model $p(r,i|u)$ in the Equation (6.2b), which would be equivalent to this model under some indepence assumptions, i.e., when $p(r,i|u) = p(r|u,i)p(i)$.

Ground models for user clarity

We ground the different clarity measures defined in the previous section upon a rating-oriented probabilistic model very similar to the approaches taken in (Hofmann, 2004) and (Wang et al., 2008a). The sample space for the model is the set $\mathcal{U} \times \mathcal{I} \times \mathcal{R}$, where $\mathcal{U}$ stands for the set of all users, $\mathcal{I}$ is the set of all items, and $\mathcal{R}$ is the set of all possible rating values. Hence, an observation in this sample space consists of a user assigning a rating to an item. We consider three natural random variables in this space: the user, the item, and the rating value, involved in a rating assignment by a user to an item. This gives meaning to the distributions expressed in the different versions of clarity as defined in the previous section. For instance, $p(r|i)$ represents the probability that a specific item i is rated with a value r – by a random user –, $p(i)$ is the probability that an item is rated – with any value by any user –, and so on.

The probability distributions upon which the proposed clarity models are defined can use different estimation approaches, depending on the independence assumptions one would consider, and the amount of involved information. Background models are estimated using relative frequency estimators, that is:

$$p_c(r) = \frac{|\{(u,i) \in \mathcal{U} \times \mathcal{I} | r(u,i) = r\}|}{|\{(u,i) \in \mathcal{U} \times \mathcal{I} | r(u,i) \neq \emptyset\}|} \tag{6.3}$$

$$p_c(i) = \frac{|\{u \in \mathcal{U} | r(u,i) \neq \emptyset\}|}{|\{(u,j) \in \mathcal{U} \times \mathcal{I} | r(u,j) \neq \emptyset\}|}$$

$$p_{ml}(r|i) = \frac{|\{u \in \mathcal{U} | r(u,i) = r\}|}{|\{u \in \mathcal{U} | r(u,i) \neq \emptyset\}|}$$

$$p_{ml}(r|u) = \frac{|\{i \in \mathcal{I} | r(u,i) = r\}|}{|\{i \in \mathcal{I} | r(u,i) \neq \emptyset\}|}$$

These are maximum likelihood estimations in agreement with the meaning of the random variables as defined above. Starting from these estimations, user models can be reduced to the above terms by means of different probabilistic expansions and Bayesian reformulations, which we define next for the three models introduced in the previous section.

Item based model. The $p(i|u)$ model can be simply expanded through marginalisation over ratings, but under two different assumptions: the item generated by the model only depends on the rating value, independently from the user or, on the contrary, depends on both the user and the rating. These alternatives lead to the following developments, respectively:

$$p_R(i|u) = \sum_{r \in \mathcal{R}} p_{ml}(i|r) p_{ml}(r|u) \tag{6.4}$$

$$p_{UR}(i|u) = \sum_{r \in \mathcal{R}} p(i|u,r) p_{ml}(r|u) \tag{6.5}$$

Rating based model. This model assumes that the rating value generated by the probability model depends on both the user and the item at hand. For this model, we sum over all possible items in the following way:

$$p(r|u) = \sum_{r(u,i)=r} p(r|u,i) p(i|u) \tag{6.6}$$

where the $p(i|u)$ term can be developed as in the item-based model above. The term $p(r|u,i)$ requires further development, which we define in the next model.

Item-and-rating based model. Three different models can be derived depending on how the Bayes' rule is applied. In these models, item probability is assumed to be uniform and thus it can be ignored in the computation of the expectation in Equation (6.2). In the same way as proposed in (Wang et al., 2008a), three relevance models can be defined, namely a user-based, an item-based, and a unified relevance model:

$$p_U(r|u,i) = \frac{p(u|r,i) p_{ml}(r|i)}{\sum_{r \in \mathcal{R}} p(u|r,i) p_{ml}(r|i)} \tag{6.7}$$

$$p_I(r|u,i) = \frac{p(i|u,r)p_{ml}(r|u)}{\sum_{r\in\mathcal{R}} p(i|u,r)p_{ml}(r|u)} \tag{6.8}$$

$$p_{UI}(r|u,i) = \frac{p(u,i|r)p_{\mathrm{c}}(r)}{\sum_{r\in\mathcal{R}} p(u,i|r)p_{\mathrm{c}}(r)} \tag{6.9}$$

The first derivation induces a user-based relevance model because it measures by $p(u|r,i)$ how probable it is that a user rates item i with a value r. The item-based relevance model is factorised proportional to an item-based probability, i.e., $p_I(r|u,i) \propto p(i|u,r)$. Finally, in the unified relevance model, we have $p_{UI}(r|u,i) \propto p(u,i|r)$. These estimations correspond respectively with the Equations 20a, 20b, and 21 from (Wang et al., 2008a); to make the thesis self-contained and facilitate the comparison between the different probability models, we present now these equations from (Wang et al., 2008a):

$$p(u|r,i) = \frac{1}{|S(r,i)|}\sum_{v\in S(r,i)} \frac{1}{h_u^{|\mathcal{I}|}} K\left(\frac{\boldsymbol{u}-\boldsymbol{v}}{h_u}\right) \tag{6.10}$$

$$p(i|u,r) = \frac{1}{|S(r,u)|}\sum_{j\in S(r,u)} \frac{1}{h_i^{|\mathcal{U}|}} K\left(\frac{\boldsymbol{i}-\boldsymbol{j}}{h_i}\right) \tag{6.11}$$

$$p(u,i|r) = \frac{1}{|S(r)|}\sum_{(v,j)\in S(r)} \frac{1}{h_u^{|\mathcal{I}|}} K\left(\frac{\boldsymbol{u}-\boldsymbol{v}}{h_u}\right)\frac{1}{h_i^{|\mathcal{U}|}} K\left(\frac{\boldsymbol{i}-\boldsymbol{j}}{h_i}\right) \tag{6.12}$$

where $K(\cdot)$ is a Parzen Kernel function (Duda et al., 2001). In this formulation, $\boldsymbol{u}$ denotes the user u represented as a vector by her ratings in the space of items. Unrated items can be filled with the average rating value or with other constant value, such as 0 or the average rating in the community. Respectively, $\boldsymbol{i}$ represents the item i in the user space. h_u and h_i are the bandwidth window parameter for the user and item vector, respectively; $S(\cdot)$ denotes the set of observed samples where event $(\cdot)$ has happened. For example, $S(r,i)$ denotes the set of observed samples with event $(R=r, I=i)$. More specifically:

$$S(r,i) = \{u \in \mathcal{U} | r(u,i) = r\} \tag{6.13}$$

$$S(r,u) = \{i \in \mathcal{I} | r(u,i) = r\} \tag{6.14}$$

$$S(r) = \{(u,i) \in \mathcal{U}\times\mathcal{I} | r(u,i) = r\} \tag{6.15}$$

In the experiments, we used a Gaussian Kernel function, i.e., $K(x) = e^{-x^2/2}/\sqrt{2\pi}$, and $h_i = h_u = 0.9$ as suggested in (Wang et al., 2008a).

User clarity name	User dependent model	Background model
RatUser	$p_U(r\|u,i); p_{UR}(i\|u)$	$p_c(r)$
RatItem	$p_I(r\|u,i); p_{UR}(i\|u)$	$p_c(r)$
ItemSimple	$p_R(i\|u)$	$p_c(i)$
ItemUser	$p_{UR}(i\|u)$	$p_c(i)$
IRUser	$p_U(r\|u,i)$	$p_{ml}(r\|i)$
IRItem	$p_I(r\|u,i)$	$p_{ml}(r\|i)$
IRUserItem	$p_{UI}(r\|u,i)$	$p_{ml}(r\|i)$

Table 6.2. Different user clarity models implemented.

Finally, different combinations of distribution formulations and estimations result in a fair array of alternatives. Among them, we focus on a subset that is shown in Table 6.2, which provide the most interesting combinations, in terms of experimental efficiency, of user and background distributions for each clarity model. These alternatives are further analysed in detail below (with examples) and in Section 6.5.1 where correlations obtained by each model are presented.

Qualitative observation

In order to illustrate the proposed prediction framework and give an intuitive idea of what user characteristics the predictors are capturing, we show the relevant aspects of specific users that result in clearly different predictor values, in a similar way to the examples provided in (Cronen-Townsend et al., 2002) for query clarity. We compare three user clarity models out of the seven models presented in Table 6.2: one for each formulation included in Table 6.1. In order to avoid distracting biases on the clarity scores that a too different number of ratings between users might cause, we have selected pairs of users with a similar number of ratings. This effect would be equivalent to that found in Information Retrieval between the query length and its clarity for some datasets (Hauff, 2010).

Table 6.3 shows the details of two sample users on which we will illustrate the effect of the predictors. As we may see in the table, u_2 has a higher clarity value than u_1 for the three models analysed. That is, according to our theory, u_2 is less "ambiguous" than u_1. Figure 6.1 shows the clarity contribution in a term-by-term basis for one of the item-and-rating-based clarity models – where, in this case, terms are equivalent to a pair (rating, item) – as analysed in (Cronen-Townsend et al., 2002). In the figure, we plot $p(r|u,i)\log_2(p(r|u,i)/p(r|i))$ for the different terms in the collection, sorted in descending order of contribution to the user model, i.e.,

User	Number of ratings	ItemUser clarity	RatItem clarity	IRUserItem clarity
u_1	51	216.015	28.605	6.853
u_2	52	243.325	43.629	13.551

Table 6.3. Two example users, showing the number of ratings they have entered, and their performance prediction values for three user clarity models.

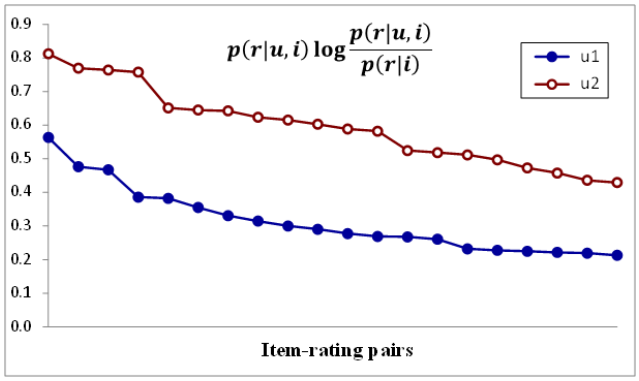


Figure 6.1. Term contributions for each user, ordered by their corresponding contribution to the user language model. IRUserItem clarity model.

$p(r|u,i)$, for each user. For the sake of clarity, only the top 20 contributions are plotted. We may see how the user with the smaller clarity value receives lower contribution values than the other user. This observation is somewhat straightforward since the clarity value, as presented in Equation (6.1), is simply the sum of all these contributions, over the set of terms conforming the vocabulary. In fact, the figures are analogous for the rest of the models, since one user always obtains higher clarity value than the other.

Let us now analyse more detailed aspects in the statistical behaviour of the users that explain their difference in clarity. The IRUserItem clarity model captures how differently a user rates an item with respect to the community. Take for instance the top item-rating pairs for users 1 and 2 in the above graphic. The top pair for u_2 is (4, "McHale's Navy"). This means that the probability of u_2 rating this movie with 4 is much higher than the background probability (considering the whole user community) of this rating for this movie. Indeed, we may see that u_2 rated this movie with a 3, whereas the community mode rating is 1 – quite farther away from 4. This is the trend in a clear user. On the other extreme of the displayed values, the bottom term in the figure for u_1 is (2, "Donnie Brasco"), which is rated by this user with a 5, and the community mode rating for this item is 4, thus showing a very similar trend between both. This is the characteristic trend of a non-clear user.

Furthermore, if we compare the background model with the user model, we obtain more insights about how our models are discriminating distinctive from mainstream behaviour. This is depicted in Figure 6.2. In this situation, we select those terms which maximise the difference between the user and background models. Then, for this subset of the terms, we sort the vocabulary with respect to its collection probability, and then we plot the user probability model for each of the terms in the vocabulary.

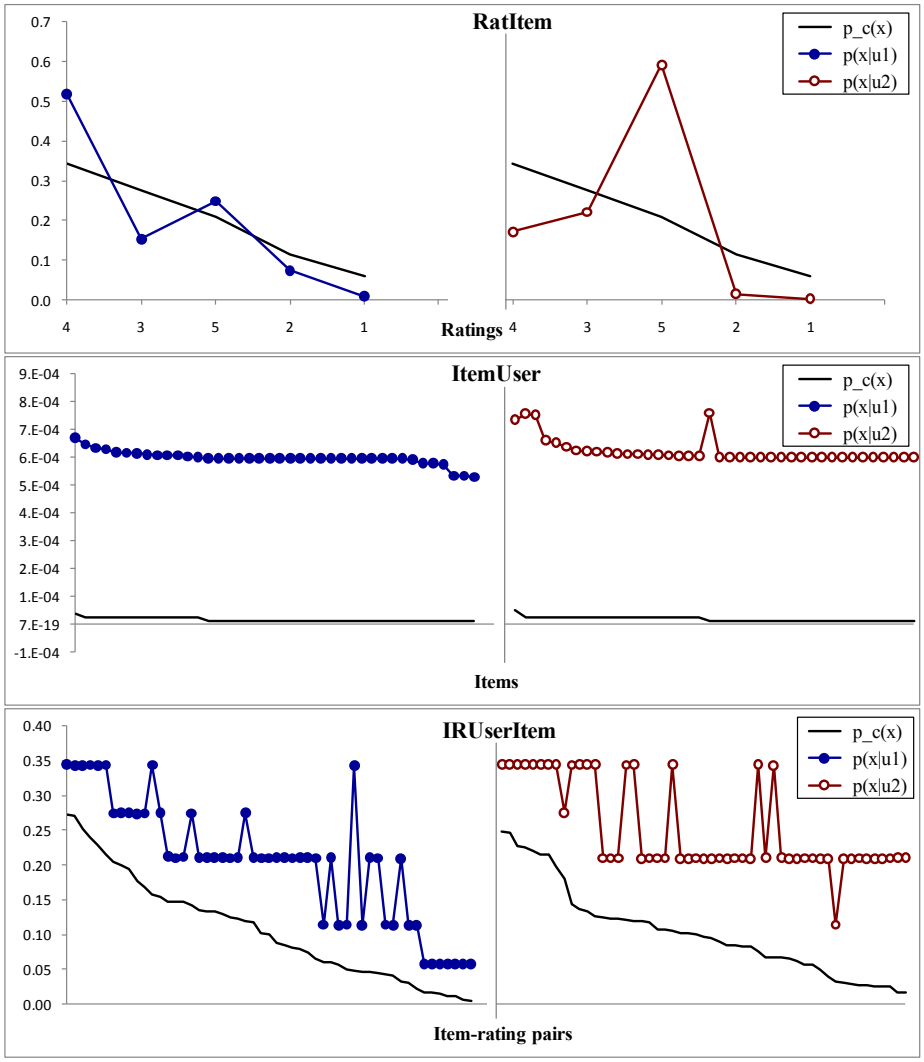


Figure 6.2. User language model sorted by collection probability.

These figures show how the most ambiguous user obtains a similar distribution to that of the background model, while the distribution of the less ambiguous user is more different. In the rating-based model this effect is clear, since the likelihood of not so popular rating values (i.e., a '5') is larger for u_2 than for u_1, and at the same time, the most popular rating value (a '4') is much more likely for u_1. The figure about the ItemUser model is less clear in this aspect, although two big spikes are observed for u_1 with respect to the collection distribution, which correspond with two unpopular movies: 'Waiting for Guffman' and 'Cry, the beloved country', both with a very low collection probability. Finally, the figure about the IRUserItem model successfully shows how u_2 has more spikes than u_1, indicating a clear divergence from the background model; in fact, u_1's distribution partially mimics that of the collection. In summary, the different models proposed are able to successfully sepa-

Item clarity	Vocabulary $\mathcal{X}$ / Context θ	Item model	Background model	Formulation
Rating-based	Ratings / None	$p(r\|i)$	$p_c(r)$	$\sum_r p(r\|i)\log_2\frac{p(r\|i)}{p(r)}$
User-based	Users / None	$p(u\|i)$	$p_c(u)$	$\sum_u p(u\|i)\log_2\frac{p(u\|i)}{p(u)}$
User-and-rating-based	Ratings / Users	$p(r\|u,i)$	$p_{ml}(r\|u)$	$\sum_{r,u} p(u)p(r\|i,u)\log_2\frac{p(r\|i,u)}{p(r\|u)}$

Table 6.4. Three possible item clarity formulations, depending on the interpretation of the vocabulary and context spaces.

rate information concerning the user and that from the collection, in order to infer whether a user is different or similar from the collection as a whole.

Item clarity

Alternatively to user-based predictors, we can also consider item-based predictors, where the performance prediction is made on an item-basis. Item predictors can be defined analogously as those defined previously for users, the equation for **item clarity** being as follows:

$$\text{clarity}(i) = \mathbb{E}_\theta\left[\sum_{x\in\mathcal{X}} p(x|i,\theta)\log_2\frac{p(x|i,\theta)}{p(x|\theta)}\right] \tag{6.16}$$

The formulation of the item predictors we propose is basically equivalent to the user-based scheme but swapping users and items. That is, we have the three formulations presented in Table 6.4 where the vocabulary now may be either ratings or users, and the context variable is the user space. Based on these three formulations, and on derivations analogous to those presented before, we propose the seven item predictors defined in Table 6.5 which are further evaluated in Section 6.5.2.

In some of the instantiations of the item clarity predictor, we may observe that there are item probability models statistically equivalent to some of the user probability models, such as the $p_U(r|i,u)$ and $p_U(r|u,i)$. For this reason, we now only spec-

Item clarity name	Item dependent model	Background model
RatItem	$p_I(r\|i,u); p_{IR}(u\|i)$	$p_c(r)$
RatUser	$p_U(r\|i,u); p_{IR}(u\|i)$	$p_c(r)$
UserSimple	$p_R(u\|i)$	$p_c(u)$
UserItem	$p_{IR}(u\|i)$	$p_c(u)$
URItem	$p_I(r\|i,u)$	$p_{ml}(r\|u)$
URUser	$p_U(r\|i,u)$	$p_{ml}(r\|u)$
URItemUser	$p_{IU}(r\|i,u)$	$p_{ml}(r\|u)$

Table 6.5. Different item clarity models implemented.

ify those probability models which have not been defined before, for the rest of estimations see Equation (6.3):

$$p_R(u|i) = \sum_{r \in \mathcal{R}} p_{ml}(u|r) p_{ml}(r|i) \tag{6.17}$$

$$p_{IR}(u|i) = \sum_{r \in \mathcal{R}} p(u|i,r) p_{ml}(r|i) \tag{6.18}$$

$$p_{IU}(r|i,u) = p_{UI}(r|i,u) \tag{6.19}$$

6.2.2 Log-based clarity

In this section we adapt some of the previous models proposed for user clarity when the preference data come in the form of user-item interaction logs. Log data has a particularity we aim to exploit: the number of times a user consumes (purchased, listened, browsed, etc.) an item may be higher than one, in contrast with rating-based preferences, where the relation between a user and an item is summarised as a unique value, the rating. Moreover, the timestamp of the interactions has a stronger meaning in the implicit approach, as it informs of the very instant the user decided to use the item, rather than the time when the user decided to reflect on her quality of experience with the item (rating time). Specialised recommendation algorithms have been proposed in the literature that exploit such features in order to obtain better recommendations (Xiang et al., 2010; Lee et al., 2008). Additional alternatives for the definition of the vocabulary may be proposed, but we shall focus on these two: log co-occurrences and timestamps.

Specifically, based on Equation (6.2) and the three instantiations of $\mathcal{X}$ and θ shown in Table 6.1, in principle only an instantiation analogous to the second one ($\mathcal{X} = \mathcal{I}$, no context – to which we shall refer as frequency-based clarity) makes sense here, as there is no rating space. However, it is possible to consider an additional space, which leads to structurally similar instantiations by taking time as the $\mathcal{X}$ vocabulary. The similarity is only syntactic, as the meaning and implications of the resulting magnitude, to which we shall refer as time-based clarity, are quite different from rating-based clarity – in other words, ratings and time are quite different dimensions –, as we shall describe later below.

Frequency-based clarity

As mentioned above, we may define the following instantiation of the Equation (6.2) based on frequencies as follows:

$$\text{frequency-based clarity}(u) = \sum_i p(i|u) \log_2 \frac{p(i|u)}{p(i)} \tag{6.20}$$

where now the estimations of the user and background models are computed using directly the frequencies of the co-occurrences of some particular user-item interaction in the data:

$$p(i) = \frac{\text{freq}(i)}{\sum_{j \in \mathcal{I}} \text{freq}(j)}$$
$$p(i|u) = \frac{\text{freq}(i,u)}{\sum_{j \in \mathcal{I}_u} \text{freq}(j,u)} \quad (6.21)$$

An alternative to such estimations is to use transformations from implicit log-based to explicit ratings, such as the one proposed in (Celma, 2008). In that approach, any of the predictors based on ratings proposed in the previous section could be applied, since these transformations give the additional vocabulary space of ratings that was absent in principle in log data.

Time-based clarity

As introduced earlier, the second dimension susceptible to be exploited when log-based preference data are available is time. The time dimension is being paid increasing attention in Information Retrieval, where, for instance, it has been integrated into language models as a means to capture some temporal information needs from the user (Berberich et al., 2010), and the temporal query dynamics are being increasingly considered in the field (Kulkarni et al., 2011). In fact, temporal query features have also been used for query performance prediction, showing low or moderate correlation with query performance by themselves, although higher correlation is obtained when such features are combined with query clarity (Diaz, 2007; Diaz and Jones, 2004).

Furthermore, time has an inherent place in recommendation: recommender systems take as input (potentially long) histories of user interaction with items (Lathia, 2010; Zimdars et al., 2001; Burke, 2010). Time is an essential dimension in making sense of the data, and in explaining, analysing and interpreting the motivations behind the actions of users recorded over time. We propose to bring these ideas to recommender systems, in particular, to adapt the temporal features studied by Díaz and colleagues on a recommender system dataset. More specifically, we use the temporal Kullback-Leibler divergence described in (Diaz and Jones, 2004) as a starting point, which we generalise and elaborate upong by considering the instantiation of Equation (6.2) for a time-based space $\mathcal{X}$, and the space of items as a possible contextual dimension, as presented in Table 6.6. In the following, we define the specific instantiations of the temporal clarity formulations presented in this table.

User clarity	Vocabulary $\mathcal{X}$ / Context θ	User model	Background model	Formulation
Time-based	Time / None	$p(t\|u)$	$p(t)$	$\sum_t p(t\|u)\log_2\frac{p(t\|u)}{p(t)}$
Item-and-time-based	Time / Items	$p(t\|u,i)$	$p(t\|i)$	$\sum_{t,i} p(i)p(t\|u,i)\log_2\frac{p(t\|u,i)}{p(t\|i)}$

Table 6.6. Two temporal user clarity formulations, depending on the interpretation of the vocabulary space.

Time based model. We denote as **TimeSimple clarity** the most direct adaptation for temporal clarity, which does not use any further extension over other dimensions. It simply computes $p(t|u)$ using smoothing (see below) and $p_c(t)$ from the collection frequencies.

Item-and-time based model. Like in the previous section, we develop conditional probabilities into sums with respect to a third variable: the items rated by the user. Here, we define two temporal clarity predictors depending on the distribution assumed for the items in the summation. If the distribution is uniform we denote such predictor as **ItemTime clarity** and $p(i) = 1/|\mathcal{I}|$. If, on the other hand, we also want to incorporate the popularity of the item for – which we have more data in this context and makes more sense than in rating data, since there the interaction between a user and an item is binary –, we include the prior item probability as $p(i) = p_c(i)$, which can be estimated considering the frequency by which i is accessed based on the interaction log.

The probabilities presented above are estimated as follows:

$$
\begin{aligned}
p_c(t) &= \frac{|\{(v,j,t) \in \mathcal{L} | v \in \mathcal{U}, j \in \mathcal{I}\}|}{|\mathcal{L}|} \\
p_c(i) &= \frac{|\{(v,i,s) \in \mathcal{L} | v \in \mathcal{U}, s \in \mathcal{S}\}|}{|\mathcal{L}|} \\
p_{ml}(t|i) &= \frac{|\{(v,i,t) \in \mathcal{L} | v \in \mathcal{U}\}|}{|\{(v,i,s) \in \mathcal{L} | v \in \mathcal{U}, s \in \mathcal{S}\}|} \\
p_{ml}(t|u) &= \frac{|\{(u,j,t) \in \mathcal{L} | j \in \mathcal{I}\}|}{|\{(u,j,s) \in \mathcal{L} | j \in \mathcal{I}, s \in \mathcal{S}\}|} \\
p_{ml}(t|u,i) &= \frac{|\{(u,i,t) \in \mathcal{L}\}|}{|\{(u,i,s) \in \mathcal{L} | s \in \mathcal{S}\}|}
\end{aligned}
\tag{6.22}
$$

Note that the variable t in (u,i,t) in the above expressions denotes a timestamp in the discretised time segment (e.g. day, week) represented by t. Furthermore, these are simple estimations of the distributions; hence, it is also possible to introduce non-parametric estimations or additional expansions through similar users or items (Wang et al., 2006a; Wang et al., 2008a). Moreover, distributions can also be modeled by

other statistical theories or hypothesis (such as Bayesian inversion), and distribution fitting/modelling from time series theory could also be studied (Diaz and Jones, 2004; Wang et al., 2008b).

In particular, we have smoothed these estimations using Jelinek-Mercer as follows:

$$\begin{aligned} p(t|i) &= \lambda\, p_{ml}(t|i) + (1-\lambda)\, p_c(t) \\ p(t|u) &= \lambda\, p_{ml}(t|u) + (1-\lambda)\, p_c(t) \\ p(t|u,i) &= \lambda\, p_{ml}(t|u,i) + (1-\lambda)\, p_c(t) \end{aligned} \tag{6.23}$$

6.3 Predictors based on social topology

Social information is widespread nowadays. As we surveyed in Chapter 2, recommender systems that use social information are proliferating in the research literature, as well as in the recommender system industry, because of the effectiveness they are being found to have. It seems therefore sensible to consider social information as a potentially useful input for predicting the performance of recommendation. The motivation for this approach is obvious when applied to social recommender systems, though we will also explore its potential properties in relation to non-explicitly social recommendation, in order to study whether social topologies may have an indirect effect on the results of the algorithms for different users.

With this goal in mind, we explore the use of graph-based measures as indicators of the user strength in the social network, which may in turn correlate with the ease or difficulty of users as recommendation targets. Graph-based measures developed from link-analysis theory are straightforward to interpret where they are often used to understand the structure of communities within a population (De Choudhury et al., 2010; Albert and Barabási, 2002). As a basis for user performance prediction they may thus bring an advantage in terms of explaining the predictions.

More specifically, the utilised indicators of the user strength in the network are based on the following vertex measures computed over the social network for each user, where a user is represented as a node in the graph, and the user's friends correspond with the node's neighbours:

- **Average neighbour degree**: mean number of friends of each user's friend (Kossinets and Watts, 2006).
- **Betweenness centrality**: indicator of whether a user can reach others on relative short paths (Freeman, 1977).
- **Clustering coefficient**: probability that the user's friends are friends themselves (Watts and Strogatz, 1998).

- **Degree**: number of the user's friends in the social network (Milgram, 1967).
- **Ego components size**: number of connected components remaining when the user and her friends are removed (Newman, 2003).
- **HITS**: Kleinberg, 1999) defines two complementary measures which assign recursively a weight to each vertex (user) depending on the topology of the network. In this way, they define hubs and authorities: a vertex is a hub when it links to authoritative vertices, and is an authority when it links to hub vertices. Since the social network used here (see Appendix A.1.3) is undirected, hub and authority scores are redundant and we only report one, denoted as **HITS**.
- **PageRank** score: well-known measure of connectivity relevance within a social network based on a random walk over the vertices, where a probability ($\alpha = 0.2$ in our experiments) of jumping to any other vertex is introduced (Brin and Page, 1998).
- **Two-hop neighbourhood** size: count of all the user's friends plus all the user's friend's friends (De Choudhury et al., 2010).

6.4 Other approaches

As a reference for comparison, we shall also test further predictors besides the ones proposed in the thesis, directly drawn from the literature, and not necessarily based on probabilistic formalisations, but following more loose formalisations, or heuristic approaches. As a further sanity check, we shall also examine obvious and simple functions (such as the amount of activity of a user), as a reference for the justification of elaborate approaches as proposed. Next, we present these predictors which are evaluated and compared in Section 6.5.

6.4.1 Using rating-based preference data

A fairly simple user predictor against which we would like to compare more elaborate functions is the **count** predictor, namely the number of items a user has rated at some specific moment. This predictor, as we shall see later, can be defined in the training set and in the test set, and although its rationale is the same, the output has different implications. Whereas in training this predictor is measuring how much information a recommender knows about some specific user, in test this value would be related to the amount of relevance used to obtain the performance metric. Furthermore, as observed in Chapter 4, the amount of relevance would be different depending on the evaluation methodology considered. However, we have to note that, due to statistical effects, the training count (profile size in training) and test count

(profile size in test) would probably be related if the training/test split is performed randomly.

$$\text{count}(u) = |\mathcal{I}_u| = |\{(u,\cdot,\cdot)\}| \tag{6.24}$$

Two additional heuristic predictors can be defined by looking at user statistics such as the **mean** and the **standard deviation** of the user's ratings. It seems plausible that such predictors would not be equally powerful for any type of recommender: it would depend on whether these statistics are used by the recommender. For instance, one might have the intuition that the higher the standard deviation, the lower the recommendation performance as one may figure out uniform user ratings to be a somewhat easier target.

$$\text{mean}(u) = \frac{1}{|\mathcal{I}_u|}\sum_{i\in\mathcal{I}_u} r(u,i) \tag{6.25}$$

$$\text{std}(u) = \sqrt{\frac{1}{|\mathcal{I}_u|}\sum_{i\in\mathcal{I}_u}\big(r(u,i) - \text{mean}(u)\big)^2} \tag{6.26}$$

Alternatively to these heuristic predictors, we have also experimented with a predictor defined upon the past observed recommender's performance. In this way, this predictor – denoted as **training performance** from now on – use a validation set (as a subset of the original training set) to evaluate the performance of each user with respect to a specific recommender; then, this value is the one returned by the predictor at test time. This approach is inspired in the Machine Learning techniques which aim to learn a feature (in this case, the user's performance) by using some training information. For this predictor, this training information is the performance computed on the validation set.

Additionally, we propose to measure the **entropy** of the user's preferences as a quantification of the uncertainty associated with a probability distribution (Cover and Thomas, 1991). We may therefore assess the uncertainty involved in the system's knowledge about a user's preferences by the entropy of the item distribution (the probability to choose an item) given the information in the user profile, using the ground models presented in Section 6.2.1. Hence, we define this predictor as follows:

$$\text{entropy}(u) = \sum_{i\in\mathcal{I}_u} p(i|u)\log_2 p(i|u) \tag{6.27}$$

Alternative measures from Information Theory could be used to define user-based predictors, like Information Gain (Bellogín, 2009), but we leave them out of this analysis because its application to Recommender Systems is neither clear nor principled and their predictive results are not optimal. Furthermore, other measures already proposed in the literature such as inverse user frequency (Breese et al., 1998)

and the analogous inverse item frequency (Bellogín, 2009), and other manipulations of the same concept, are also ignored here because they are simply transformations of the previously presented count predictor. Finally, the concept of power users (Lathia et al., 2008) could also be used as a proxy for well-performing users, but preliminary results have not shown strong predictive power.

6.4.2 Using log-based preference data

As we have observed in the previous section, recommendation performance usually has obvious predictors, obvious in the sense that they do not involve any interesting finding or insightful kind of analysis, or anything to learn from. We include in our analysis some of these obvious predictors, framed as baseline performance predictors that basically count how many interactions a user has had with the system. In this sense, these predictors are slightly different to the ones presented in the previous section, namely because in log-based datasets repetitions of items are allowed in a user's profile. In order to account for this difference, apart from **count**, **mean**, and **standard deviation** predictors, we propose to normalise the count predictor by the number of items consumed by each user, that is, we define the **average count** predictor as follows:

$$\text{average count}(u) = \frac{|\{(u,j,s) \in \mathcal{L} | j \in \mathcal{I}, s \in \mathcal{S}\}|}{|\{j \in \mathcal{L}: (u,j,s) \in \mathcal{L}, s \in \mathcal{S}\}|} \tag{6.28}$$

We also test more elaborate predictors based on the temporal dimension, such as the ones defined in (Diaz and Jones, 2004). First-order autocorrelation (or temporal self-correlation) can be considered with a reinterpretation of the random variables. Specifically, this predictor, in contrast with the temporal Kullback-Leibler divergence where the similarity with the temporal background model is assessed, captures the structure of the query time series. For instance, a uniform distribution would have an autocorrelation value of 0, whereas a query time series with strong inter-day (or whatever segment size is used to build the discrete time series) dependency will obtain a high autocorrelation value.

Thus, we define the **autocorrelation** user predictor as follows:

$$\text{autocorrelation}(u) = \frac{\sum_{t=1}^{T}(p(t|u) - 1/\mathrm{T})(p(t+1|u) - 1/\mathrm{T})}{\sum_{t=1}^{T}(p(t|u) - 1/\mathrm{T})^2} \tag{6.29}$$

where T is the total number of time units in the time interval. We can observe how this predictor captures the similarity between two consecutive observations.

Extensions of this predictor could use the probabilities defined in Section 6.2.2, like $p(t|u,i)$, instead of $p(t|u)$. Similarly, other predictors proposed by Díaz and Jones in (Diaz and Jones, 2004) and (Jones and Diaz, 2007) such as the kurtosis or

the burst model could be adapted to recommender systems, but we leave such extensions for future work.

6.5 Experimental results

In this section we provide correlation results where all the predictors – heuristic, social, and clarity-based – are compared against each other using an array of recommendation methods and evaluation methodologies.

6.5.1 User predictors using rating-based preference data

In this section we compare the correlations obtained for the clarity-based predictors defined in Section 6.2.1, the user entropy defined in Equation (6.27), and the baselines presented in Equations (6.24), (6.25), and (6.26) using the MovieLens 1M dataset. The λ parameter for the language model smoothing was not optimised for this task and a default value of 0.6 was used in all the models as originally used in (Cronen-Townsend et al., 2002). Here, we focus on Pearson's correlation and P@10. Additional results are reported in Appendix A.4.1.

Table 6.7 shows the correlation values when the AR methodology is used. We can observe fairly high correlation values for recommenders pLSA, ItemPop, TFL2, and kNN, comparable to results in the query performance literature. A slightly lower correlation is found for TFL1, whereas no correlation is found for CB and IB. These results are consistent when other performance metrics are used such as nDCG, and at different cutoff points. Spearman's correlation yields similar values. Here we also include the count predictor in test, which is obviously not a predictor in strict sense,

Predictor	Random	CB	IB	ItemPop	kNN	pLSA	TFL1	TFL2	**Median**	**Mean**
Count (training)	0.135	0.164	0.042	0.512	0.424	0.442	0.198	0.644	0.311	0.320
Count (test)	0.135	0.172	0.042	0.520	0.431	0.452	0.200	0.647	0.316	0.325
Training performance	0.024	0.176	0.258	0.429	0.296	0.357	0.215	0.485	0.277	0.280
Mean	0.019	0.067	-0.002	0.015	0.022	0.108	0.026	-0.018	0.021	0.030
Standard deviation	0.008	0.008	0.011	-0.029	-0.031	-0.032	0.011	-0.051	-0.011	-0.013
ItemSimple Clarity	0.149	0.191	0.046	0.549	0.453	0.489	0.222	0.683	0.338	0.348
ItemUser Clarity	0.134	0.166	0.048	0.493	0.416	0.428	0.215	0.634	0.316	0.317
RatUser Clarity	0.135	0.160	0.048	0.514	0.442	0.435	0.214	0.651	0.325	0.325
RatItem Clarity	0.127	0.159	0.039	0.475	0.402	0.405	0.203	0.611	0.303	0.303
IRUser Clarity	0.128	0.157	0.027	0.486	0.382	0.408	0.182	0.599	0.282	0.296
IRItem Clarity	0.122	0.165	0.034	0.446	0.352	0.386	0.188	0.551	0.270	0.281
IRUserItem Clarity	0.128	0.158	0.033	0.479	0.379	0.403	0.193	0.594	0.286	0.296
Entropy	0.121	0.168	0.025	0.492	0.389	0.483	0.140	0.589	0.279	0.301
Median	0.128	0.162	0.037	0.489	0.396	0.418	0.196	0.605		
Mean	0.112	0.145	0.033	0.413	0.338	0.367	0.166	0.511		

Table 6.7. Pearson's correlation between rating-based user predictors and P@10 for different recommenders using the AR methodology (MovieLens dataset).

since it uses a different input than the other predictors, but we include it in our analysis as a further reference to check behaviours.

As mentioned in Chapter 5, the standard procedure in Information Retrieval for this kind of evaluation is to compute correlations between the predictor(s) and one retrieval model (like in (Cronen-Townsend et al., 2002) and (Hauff et al., 2008a)) or an average of several methods (Mothe and Tanguy, 2005). This approach may hide the correlation effect for some recommenders, as we may observe from the median and mean correlation values included in the table, which are still very large despite the fact that two of the recommenders analysed have much lower correlations. Nonetheless, these aggregated values, i.e., the mean and the median, provide competitive correlation values when compared with those in the literature.

The difference in correlation for CB and IB recommenders may be explained considering two factors: the actual recommender performance and the input sources used by the recommender. With regards to the first factor, as presented in Table 6.8, the IB algorithm performs poorly (in terms of the considered ranking quality metrics, such as precision and nDCG) in comparison to the rest of recommenders. It seems natural that a good predictor for a well performing algorithm (specifically, pLSA is the best performing recommender in this context) would hardly correlate at the same time with a poorly performing one.

This does not explain however the somewhat lower correlation with the content-based recommender, which has better performance than TFL1. The input information that this recommender and the predictors take in are very different: the latter compute probability distributions based on ratings given by users to items, while the former uses content features from items, such as directors and genres. Furthermore, the CB recommender is not coherent with the inherent probabilistic models described by the predictors, since the events modeled by each of them are different: CB would be related to the likelihood that an item is described by the same features as those items preferred by the user, whereas predictors are related to the probability that an item is rated by a user. Moreover, the predictors' ground models coherently fit in the standard collaborative framework (Wang et al., 2008a), which reinforces the suitability of the user performance predictors presented herein, at least for collaborative filtering recommenders.

It is worth noting to this respect that most clarity-based query performance predic-

Recommender	Random	CB	IB	ItemPop	kNN	pLSA	TFL1	TFL2
AR methodology	0.0025	0.0163	0.0001	0.0897	0.0307	0.1454	0.0024	0.0696
1R methodology	0.0099	0.0221	0.0074	0.0649	0.0437	0.0836	0.0221	0.0690
U1R methodology	0.0100	0.0223	0.0068	0.0406	0.0381	0.0718	0.0294	0.0524
P1R methodology	0.0101	0.0197	0.0208	0.0282	0.0265	0.0604	0.0203	0.0348

Table 6.8. Summary of recommender performance using different evaluation methodologies (evaluation metric is P@10 with the MovieLens dataset).

tion methods in Information Retrieval study their predictive power on language modelling retrieval systems (Cronen-Townsend et al., 2002; Hauff et al., 2008a; Zhou and Croft, 2007) or similar approaches (He and Ounis, 2004). This suggests that a well performing predictor should be defined upon common spaces, models, and estimation techniques as the retrieval system the performance of which is meant to be predicted.

Finally, the correlation values found by the training performance predictors, although sometimes strong, are not as high as those of the baselines predictors – such as training count – in most situations, in particular, they are always lower except for the IB and TFL1 recommenders. This highlights the importance of having a more general model for predicting the performance of a user, since these predictors in fact depend considerably on the properties of the validation (and test) partition of the data, such as the amount of sparsity, type of items evaluated and so on.

Unbiased performance prediction

In Chapter 4 we already demonstrated that some methodologies may be biased towards more popular items or sparsity constraints. We can observe in the previous table that trivial predictors such as count (either in training or in test) obtain significant (and positive) correlation, no matter the recommender. We argue whether this is because these predictors are really capturing an interesting effect or the evaluation methodology is prone to such effect. In order to overcome this problem, now we present the same correlation analysis but with the different methodologies presented in Chapter 4.

In Table 6.9 we show results with the methodology 1R. Here we can observe that most of the correlation values are lower than in the previous case; interestingly, the correlation with the Random recommender now is almost 0 for every predictor (and in particular, for the training and test profile size). This is evidence that per-

Predictor	Random	CB	IB	ItemPop	kNN	pLSA	TFL1	TFL2
Count (training)	0.061	-0.038	0.092	0.258	0.108	0.303	0.086	0.394
Count (test)	0.063	-0.033	0.091	0.266	0.115	0.312	0.089	0.398
Training performance	0.012	0.332	0.168	0.272	0.266	0.133	0.303	0.240
Mean	0.036	0.082	-0.029	0.028	0.111	0.117	0.145	0.031
Standard deviation	-0.010	0.006	0.051	-0.060	-0.116	-0.080	-0.040	-0.114
ItemSimple Clarity	0.066	-0.033	0.094	0.265	0.115	0.322	0.105	0.409
ItemUser Clarity	0.059	-0.038	0.087	0.236	0.100	0.287	0.096	0.375
RatUser Clarity	0.057	-0.054	0.083	0.245	0.130	0.285	0.086	0.372
RatItem Clarity	0.057	-0.044	0.069	0.225	0.110	0.268	0.094	0.352
IRUser Clarity	0.056	-0.020	0.053	0.250	0.069	0.280	0.077	0.364
IRItem Clarity	0.051	-0.010	0.058	0.205	0.029	0.235	0.074	0.310
IRUserItem Clarity	0.056	-0.020	0.052	0.242	0.066	0.273	0.081	0.357
Entropy	0.091	0.021	0.144	0.354	0.169	0.460	0.114	0.543

Table 6.9. Pearson's correlation between rating-based user predictors and P@10 for different recommenders using the 1R methodology (MovieLens dataset).

Predictor	Random	CB	IB	ItemPop	kNN	pLSA	TFL1	TFL2
Count (training)	0.048	-0.012	0.237	0.162	0.115	0.140	0.022	0.235
Count (test)	0.049	-0.001	0.226	0.135	0.110	0.137	0.036	0.213
Mean	0.023	0.051	-0.035	0.009	0.108	0.075	0.155	-0.006
Standard deviation	0.015	0.032	0.023	-0.047	-0.098	-0.038	-0.061	-0.049
ItemSimple Clarity	0.055	-0.005	0.241	0.166	0.128	0.153	0.042	0.241
ItemUser Clarity	0.046	-0.009	0.232	0.142	0.109	0.133	0.028	0.216
RatUser Clarity	0.045	-0.028	0.234	0.155	0.137	0.130	0.022	0.225
RatItem Clarity	0.043	-0.025	0.212	0.136	0.119	0.117	0.033	0.203
IRUser Clarity	0.044	0.002	0.180	0.153	0.069	0.134	0.029	0.210
IRItem Clarity	0.036	0.011	0.178	0.114	0.035	0.108	0.014	0.173
IRUserItem Clarity	0.042	0.003	0.178	0.147	0.065	0.130	0.028	0.203
Entropy	0.078	0.044	0.278	0.227	0.169	0.249	0.073	0.321

Table 6.10. Pearson's correlation between rating-based user predictors and P@10 for different recommenders using the U1R methodology (MovieLens dataset).

formance results using the AR methodology are higher for users with more items in their test, independently from the recommendation algorithm complexity (see correlations with Random recommender in Table 6.7). In the same way, the U1R (Table 6.10) and P1R (Table 6.11) methodologies also obtain negligible correlation values for the Random recommender, which confirms the suitability of these methodologies for our purposes. We also have to note that we have not applied the training performance predictor in these methodologies because their restrictions do not let to replicate the same conditions in a validation split. Furthermore, as stated in Chapter 4, both approaches aim to remove the bias towards more popular items. Here, we can observe how the correlation with respect to the ItemPop recommender is comparable to that with the Random recommender with the P1R methodology, confirming again the ability of this methodology to produce unbiased results (at least, with respect to popular items).

The main difference in the results obtained between these three methodologies (1R, U1R, and P1R) seems to be more at the recommender level rather than at the

Predictor	Random	CB	IB	ItemPop	kNN	pLSA	TFL1	TFL2
Count (training)	0.073	-0.005	0.253	0.088	0.103	0.160	-0.001	0.307
Count (test)	0.076	0.000	0.253	0.093	0.108	0.168	0.003	0.308
Mean	0.034	0.073	-0.033	0.008	0.110	0.085	0.188	-0.026
Standard deviation	-0.010	0.009	0.014	-0.058	-0.104	-0.044	-0.061	-0.051
ItemSimple Clarity	0.078	0.000	0.254	0.084	0.111	0.169	0.019	0.313
ItemUser Clarity	0.072	-0.001	0.249	0.075	0.101	0.156	0.005	0.303
RatUser Clarity	0.071	-0.016	0.252	0.086	0.128	0.148	0.003	0.297
RatItem Clarity	0.067	-0.011	0.234	0.077	0.113	0.138	0.016	0.288
IRUser Clarity	0.066	0.002	0.200	0.086	0.066	0.147	0.006	0.274
IRItem Clarity	0.059	0.010	0.192	0.061	0.037	0.123	-0.006	0.242
IRUserItem Clarity	0.066	0.003	0.200	0.082	0.065	0.145	0.006	0.272
Entropy	0.092	0.038	0.286	0.133	0.128	0.266	0.039	0.379

Table 6.11. Pearson's correlation between rating-based user predictors and P@10 for different recommenders using the P1R methodology (MovieLens dataset).

predictor level, in the sense that the trend in predictor effectiveness is similar for each methodology but the correlations obtained for each recommender vary dramatically from one methodology to another. For instance, IB recommender obtains near zero correlations with 1R but higher (significative) values for U1R and P1R; a similar situation occurs with the TFL2 recommender, where the correlations are lower for the U1R methodology and higher for 1R and P1R. Note that the training and test sets are the same for all the methodologies except for U1R, which means that the performance predictors are entirely new for that methodology. Thus, a priori it would not be clear that such an agreement between the different methodologies should appear at the predictor level unless they are really capturing the same nuance about the user, no matter the evaluation methodology used.

It is worth noting that the correlation values of these three methodologies have been found after a careful examination of the available data, where two different trends emerged: one where the performance values were more or less uniformly distributed in the interval [0, 0.1] – recall that 0.1 is the maximum value for the metric P@10 with the 1R methodology, since there is only one relevant item – ; and a second one where a fixed value was obtained. This second trend, against which our predictors shown no correlation at all (since the performance had a zero standard deviation, and thus the correlation was impossible to calculate) is able to degrade the correlation coefficient almost to negligible values, mainly because it accounts for half of the number of points. This problem with correlation coefficients, and with Pearson's correlation in particular, is well known in the literature of performance prediction (Hauff, 2010; Pérez Iglesias, 2012). For this reason, we have divided the performance values and computed two correlations in order to account for these two trends: the values with respect to the first trend are those presented in the previous tables, whereas the correlation with respect to the second trend was not computable because the variable had a zero standard deviation.

In summary, there seems to be no clear winner among the set of performance predictors proposed. The predictive power of each of them is clearly influenced by the actual recommender its performance aims to be predicted and the evaluation methodology in use. Nonetheless, **the proposed predictors usually obtain higher correlation values than baseline predictors** such as the mean or the standard deviation, evidencing their predictive power **independently from the evaluation methodology**. Surprisingly, the ItemSimple clarity predictor obtains very good results in most of the situations, although more complex predictors like IRUser or IRUserItem clarity obtain stronger correlations for some recommenders.

6.5.2 Item predictors using rating-based preference data

In the same way we have assessed the predictive power of user predictors, we now aim to estimate the predictive power of item predictors. However, the true perform-

	u_1	u_2	u_3
	i_1 0.8	i_2 0.6	i_3 0.9
	* i_2 0.7	* i_3 0.5	* i_4 0.6
	* i_3 0.6	* i_4 0.4	* i_1 0.5
	i_4 0.5	i_1 0.3	i_2 0.1
P@2	0.5	0.5	0.5

i_1	i_2	i_3	i_4
* u_1 0.8	u_1 0.7	u_3 0.9	* u_3 0.6
* u_3 0.5	u_2 0.6	* u_1 0.6	u_1 0.5
u_2 0.3	u_3 0.1	* u_2 0.5	*u_2 0.4
1.0	0.0	0.5	0.5

Table 6.12. Procedure to obtain ranking for items from user rankings generated by a standard recommender. * denotes a relevant item, and the numbers are the score predicted by the recommendation method.

ance value for an item is not straightforward to compute, since the process has to produce unbiased results in the space of items (as described in Chapter 4) but with the characteristic that the item dimension is not the main input of the recommendation process, and thus, sone new approach has to be put in place.

There are basically two possibilities for computing the true performance on an item: either starting from the results obtained using a standard procedure (obtain a ranking for each user by recommending items to users), then transposing users and items (generating, thus, user rankings for each item) and computing the per-ranking performance as usual; or transpose the original rating matrix in order to effectively "recommend users" for each item. This would implicitly imply a transposition of the recommendation task, which may also make sense: find the most suitable users to recommend an item – this would be the scenario, e.g. in advertisement targeting when a new product is released on the market. Here, we use the former approach since the latter does not produce consistent results in our experiments, probably because the recommendation problem is not completely symmetric and, thus, this method is not able to properly capture the recommender's performance for each item. On the other hand, non-personalised recommenders (such as recommendation by item popularity) cannot be applied in the symmetric formulation: since the same item ranking is built for all users, the user ranking for an item would be a global tie on all users. Table 6.12 shows an example of how we may transpose users and items from an item ranking for three users. We show that the precision for all the users is the same, whereas for the items is completely diverse, ranging from zero to perfect precision.

In our experiments, we have tested the different methodologies already presented along with a modified version of the U1R evaluation methodology (user-uniform U1R, or uuU1R). The rationale for the uuU1R design goes as follows: in the U1R methodology we force the same number of ratings (or, equivalently, users) for the items in the test set, however, users are freely assigned to each item. Now, when we transpose users and items this situation may produce a new problem, since there

Predictor	Random	CB	ItemPop	kNN	pLSA
Count (training)	0.414	0.060	-0.151	-0.021	-0.269
Count (test)	-------	-------	-------	-------	-------
Mean	0.602	0.125	0.096	0.040	-0.038
Standard deviation	-0.313	0.025	-0.006	-0.003	0.075
UserSimple Clarity	0.467	0.080	-0.120	-0.015	-0.240
UserItem Clarity	0.419	0.064	-0.145	-0.018	-0.261
RatItem Clarity	0.440	0.075	-0.127	-0.015	-0.230
RatUser Clarity	0.451	0.085	-0.103	-0.004	-0.201
URItem Clarity	0.396	0.053	-0.174	-0.026	-0.289
URUser Clarity	0.408	0.072	-0.132	-0.004	-0.243
URItemUser Clarity	0.409	0.061	-0.161	-0.021	-0.277
Entropy	0.381	-0.001	-0.216	-0.055	-0.442

Table 6.13. Pearson's correlation for rating-based item predictors and precision using the uuU1R methodology (MovieLens dataset).

could be users assigned to more items which would bias the ranking's performance towards items contained in the test set of heavy raters. Therefore, if we impose a uniform distribution also on the user's dimension, this bias should decrease. We refer to the reader to Appendix A.3 for more details.

However, despite these efforts, we have not found a reliable methodology to evaluate the item performance. We present in Table 6.13 the results using the uuU1R methodology and the predictors defined in Table 6.5 for the precision metric. Recall that, since we transpose users and items from the generated rankings, to obtain a similar measure of P@10 we only use the top 10 items from each original ranking and then compute precision over the whole ranking for each item. We may observe in the table that the correlations with the Random recommender are very strong, questioning the validity of such results. Besides, the entropy predictor obtains stronger correlation than clarity-based in this case, and most of them (except for URItem) show little difference to training count. Note that it is not possible to compute a correlation with the test count predictor since that predictor has a constant value with zero standard deviation (see Equation (5.11) for more details on Pearson's correlation) since every item has the same number of ratings in the test set in the uuU1R methodology.

As a conclusion, we have found that **a proper evaluation of item performance is not obvious**, mainly because the task of suggesting users to items is not completely symmetric with respect to the standard task of recommendation. We have devised different methodologies to estimate the recommendatoin performance of an item, however the difficulty lies mainly in forming consistent lists of "recommended" users for items, a difficulty which is not conceptual (ranking target users to whom an item may be recommended does make sense as a task in many scenarios), but technical (obtaining balanced result lists that allow for undistorted performance measurements).

6.5.3 User predictors using log-based preference data

In this section we analyse the correlation obtained between the predictors defined in Sections 6.2.2 and 6.4.2 and five recommenders using the 1R methodology on two versions of the Last.fm dataset – one where a temporal partition is performed and another where the partition is randomly made (more details about the splits in Appendix A.1.2). No smoothing was used in the language models since preliminary tests obtained better results with lower values of λ. Besides, for comparison purposes, we also include one of the clarity models proposed for rating-based preference data using the transformation proposed in Section 6.2.2 to use such predictors with log data along with the frequency-based clarity proposed in Equation (6.20). Like in the previous section, Pearson's correlation with the P@10 evaluation metric is reported; for additional metrics, see Appendix A.4.2.

First, we can observe in Table 6.14 (temporal split) that ItemPriorTime clarity obtains strong correlation values, especially for the ItemPop and kNN recommenders. It is interesting to compare the correlations between this predictor and the ItemTime clarity, which are much lower. This is probably because the ItemPriorTime clarity predictor, as opposed to ItemTime clarity, incorporates a component that measures the item popularity, i.e., $p(i)$. The TimeSimple and the frequency-based clarity predictors, on the other hand, obtain strong correlation but negative values for all the recommenders except the ItemPop for the TimeSimple predictor. Furthermore, the ItemSimple clarity (a predictor based on explicit information) obtains negligible correlations except for the ItemPop and kNN recommenders.

Table 6.15, on the other hand, shows the results when a random split is used. We have to note that such split does not preserve the temporal continuity of the user's preferences, and thus, any recommender or technique which makes use of temporal features is not guaranteed to succeed. Here, we can observe that TimeSimple predictor obtains strong correlations for all the recommenders except for the Random

Predictor	Random	CB	ItemPop	kNN	pLSA
Average Count	0.027	0.138	0.069	-0.013	0.191
Count	0.046	0.118	-0.058	0.131	0.139
Mean	-0.079	-0.361	0.054	-0.110	-0.376
Standard deviation	-0.050	-0.158	0.082	-0.132	-0.177
Autocorrelation	0.004	0.139	-0.066	-0.105	0.100
TimeSimple Clarity	-0.091	-0.342	0.093	-0.317	-0.354
ItemTime Clarity	0.037	0.078	0.038	0.258	0.064
ItemPriorTime Clarity	0.057	0.154	0.189	0.307	0.154
Frequency-based Clarity	-0.049	-0.410	-0.221	-0.291	-0.376
ItemSimple Clarity	0.027	0.047	-0.107	0.221	0.029

Table 6.14. Pearson's correlation between log-based predictors and P@10 for different recommenders using 1R methodology (Last.fm temporal dataset).

Predictor	Random	CB	ItemPop	kNN	pLSA
Average Count	-0.023	-0.068	-0.170	-0.018	-0.087
Count	-0.012	-0.236	-0.242	-0.086	-0.198
Mean	0.036	0.182	0.100	0.047	0.118
Standard deviation	-0.009	0.089	0.079	0.092	0.082
Autocorrelation	0.045	-0.069	-0.089	-0.012	-0.055
TimeSimple Clarity	0.031	0.274	0.314	0.169	0.240
ItemTime Clarity	0.021	-0.145	0.004	0.025	-0.053
ItemPriorTime Clarity	0.011	-0.057	0.176	0.145	0.083
Frequency-based Clarity	0.025	0.018	-0.287	-0.182	-0.220
ItemSimple Clarity	0.020	-0.247	-0.163	-0.068	-0.186

Table 6.15. Pearson's correlation between log-based predictors and P@10 for different recommenders using 1R methodology (Last.fm five-fold dataset).

technique. Like before, ItemPriorTime has a high correlation with the ItemPop recommender. In contrast with the previous situation, the ItemSimple clarity obtains strong but negative correlations for the personalised recommenders. Besides, the frequency-based clarity has negative correlations for all the recommenders except CB, a consistent situation with the results obtained with the temporal split.

Hence, we may conclude that **log-based and time-aware predictors successfully predict the performance of the recommendation algorithms**, although in some situations the sign of the prediction is negative. Moreover, frequency-based, ItemSimple, and TimeSimple clarity obtain consistently strong correlations both in a temporal split and in a random split of the data, evidencing their predictive power.

6.5.4 User predictors using social-based preference data

In this section we study the correlation between the predictors described in Section 6.3 and several recommenders using the two versions of the CAMRa dataset: social and collaborative. In this case, we also consider social filtering recommenders in order to analyse whether these predictors are sensitive to the source of information used by the recommender, and thus, whether they obtain stronger correlations with social filtering recommenders. Besides, one clarity-based predictor (ItemSimple) and the baseline rating predictors presented in Section 6.4.1 are also included in the analysis for comparison purposes. Additionally, for the HITS and PageRank graph metrics in this experiment we use the implementation developed in the JUNG library (O'Madadhain et al., 2003).

Table 6.16 shows correlation values obtained when using the AR methodology in the social version of the dataset. Here, we can observe that most of the correlation values obtained for the social predictors are negative, representing that the lower the predictor output, the better the performance, which may seem a little counter-intuitive, at least for the social filtering recommenders (Personal and PureSocial).

Among the social-based predictors, degree and two-hop neighbourhood size obtain better correlations than the rest.

A similar situation is presented in Table 6.17, where the collaborative-social version of the dataset is used. Again, most of the correlations with the social-based predictors are negative, and degree and two-hop neighbourhood size obtain higher correlations (in absolute value). Interestingly, in this situation strong correlations are obtained with the user-based recommender (kNN), in particular with degree and the average neighbour degree predictors. Nonetheless, these correlations are lower than those obtained for the ItemSimple predictor with the collaborative filtering recommenders. At the same time, this predictor always obtains worse correlations (in absolute value) than the social-based predictors for the social filtering recommenders, as expected.

Additionally, note that the number of points used in the correlation computation is different in each version of the dataset, namely: in the collaborative-social version

Predictor	Random	ItemPop	kNN	pLSA	Personal	PureSocial
Count (training)	0.032	0.122	0.113	0.031	0.062	0.111
Count (test)	0.158	0.252	0.382	0.167	0.235	0.174
Mean	-0.066	0.033	-0.012	0.023	-0.057	-0.051
Standard deviation	0.034	0.054	-0.020	0.115	0.128	0.183
Avg neighbour degree	-0.062	-0.089	-0.013	0.011	-0.074	-0.106
Betweenness centrality	-0.031	-0.016	0.027	-0.038	-0.012	-0.079
Clustering coefficient	0.049	-0.084	-0.023	0.048	-0.027	-0.035
Degree	-0.038	-0.046	0.015	-0.059	-0.147	-0.133
Ego components size	-0.058	0.005	0.004	-0.046	-0.056	-0.020
HITS	-0.021	-0.043	0.005	0.061	0.038	0.000
PageRank	-0.022	-0.025	-0.023	-0.039	-0.102	-0.037
Two-hop neighbourhood	-0.080	-0.082	0.004	-0.054	-0.123	-0.136
ItemSimple Clarity	0.030	0.157	0.130	0.050	0.072	0.126

Table 6.16. Pearson's correlation between social-based predictors and P@10 for different recommenders using AR methodology (CAMRa Social).

Predictor	Random	ItemPop	kNN	pLSA	Personal	PureSocial
Count (training)	0.012	0.098	0.203	0.107	0.058	0.111
Count (test)	0.096	0.207	0.389	0.179	0.232	0.170
Mean	-0.067	0.000	-0.126	-0.024	-0.051	-0.050
Standard deviation	0.082	0.014	-0.029	0.016	0.129	0.182
Avg neighbour degree	0.071	-0.008	0.152	0.046	-0.073	-0.104
Betweenness centrality	-0.007	-0.008	0.010	-0.005	-0.012	-0.078
Clustering coefficient	0.006	-0.022	0.152	0.076	-0.032	-0.035
Degree	0.032	0.018	0.164	0.006	-0.143	-0.134
Ego components size	0.026	0.044	0.133	0.002	-0.053	-0.022
HITS	-0.011	-0.034	-0.001	0.061	0.038	0.001
PageRank	-0.002	0.021	0.118	0.014	-0.099	-0.040
Two-hop neighbourhood	0.059	-0.015	0.130	0.012	-0.121	-0.135
ItemSimple Clarity	0.010	0.120	0.211	0.129	0.070	0.126

Table 6.17. Pearson's correlation between social-based predictors and P@10 for different recommenders using AR methodology (CAMRa Collaborative).

the number of users contained in the test set is twice the number available in the social version (see Appendix A.1.3), which means that significant correlations can be achieved with lower values (as described in Chapter 5).

In the results described above, we can observe how, like in the previous sections, the size of the user profile in test (predictor count in test) obtains significant correlations. This trend, however, is almost neutralised in the collaborative-social dataset with respect to the Random recommender. Thus, as before, we would attempt to use the 1R methodology with each dataset in order to obtain unbiased correlations towards users with more ratings in test. However, due to the lack of coverage of Personal and PureSocial recommenders, this methodology do not obtain sensible results (for instance, the value of precision at 10 is almost invariably 0.10, that is, the maximum possible value when only one relevant document – as assumed in the 1R methodology – is retrieved in the top 10, mainly because the recommender is not able to retrieve most of the not relevant items). This lack of coverage is natural for these recommenders since they can only suggest items rated by users in the active user's social network (see Appendix A.2 for details on the implementation of the algorithms).

In conclusion, most of the social performance predictors proposed obtain significant correlations, however, **correlations with the social filtering methods are not so strong as we would expect**. Nonetheless, the **ItemSimple clarity does obtain significant correlations** with respect to most of the recommenders, highlighting the importance and validity of this predictor even when the main input of some recommenders (social network) is so different to that of the predictor (ratings).

6.5.5 Discussion

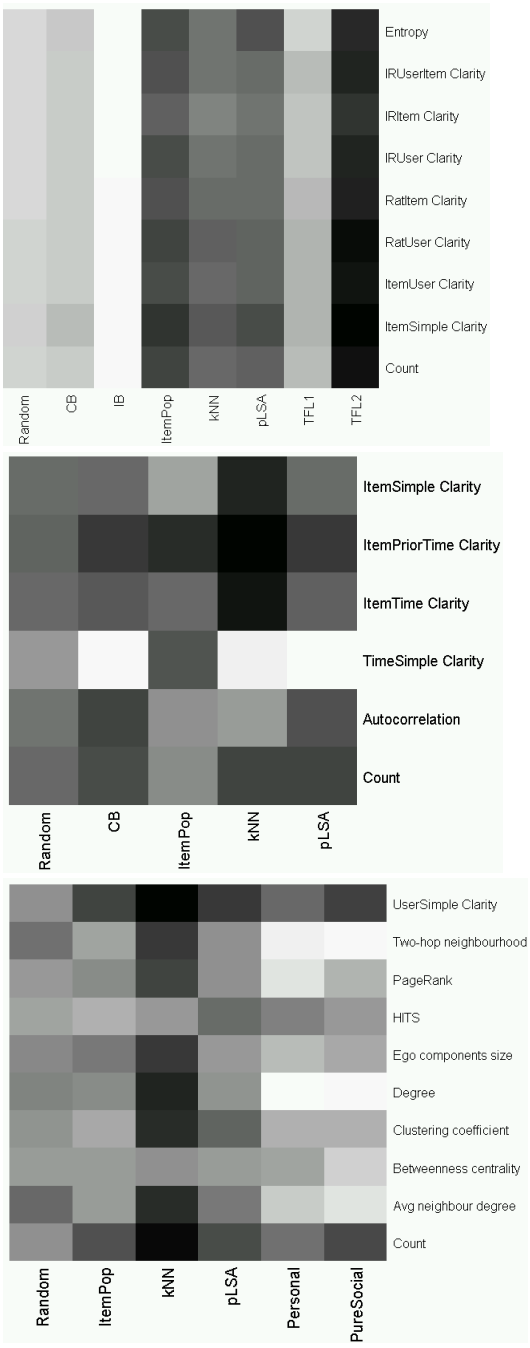
The reported experiments confirm that it is possible to predict a recommender's performance and obtain strong correlations in this regard. The results show that, in general, the proposed predictors (mostly based on Kullback-Leibler divergences over different language models and other concepts from Social Graphs and Information Theory such as entropy) obtain significant correlations in the three spaces considered: ratings, logs, and social networks. More importantly, these correlations are stronger than those obtained by more simple predictors, such as the profile size of a user, the standard deviation of her ratings, and the user's performance using a validation split. Specifically, for each recommendation input considered we have observed the following:

- Clarity-based predictors are very powerful for rating-based preferences, in particular, the ItemSimple, IRUser, and IRUserItem clarity predictors obtain strong correlations for most of the recommendation methods.

- The use of the item space as a contextual variable shows strong correlation values when the AR methodology is used, but these correlations decrease when we use unbiased methodologies, which may indicate that this new dimension is in fact capturing the item popularity and, thus, when the popularity bias is neutralised such predictors show less predictive power. We find a similar situation with the item clarity and the user space used as the contextual dimension.
- Temporal and log-based versions of the clarity predictor show higher prediction power than the rest of predictors.
- Social-based predictors are not the ones with the strongest correlation regarding the social filtering recommenders in this experiment, but the correlation found is significative and they could serve as a complement to other predictors based on a different input such as the rating-based.
- The ItemSimple clarity predictor consistently obtains strong correlation values in most of the datasets where we have analysed it. This is an evidence of the theoretical power of the user clarity to capture the uncertainty in user's tastes, even when the recommender's input is different (social filtering recommenders) or when we apply some transformation to the data (frequency-based clarity with transformation from implicit to explicit).
- As described in the Appendix A.4, most of the correlations presented in this chapter are stable when other evaluation metrics and correlation coefficients are used.

In the Recommender Systems field there are, however, additional problems due to subtle differences with respect to the common settings and experimental assumptions in Information Retrieval. Since we aim to predict the performance of a recommender, we have to be sure that we are using an unbiased performance metric, and its subsequent evaluation methodology. As we analysed in Chapter 4 there are at least two biases in the evaluation of recommender systems which may distort the results: data sparsity and item popularity. Thus, in this chapter we have computed correlations between the output of the predictors and the evaluation metrics using different evaluation methodologies, in order to analyse how sensitive the different proposed predictors are to these biases. Interestingly, although the correlations may change drastically when different evaluation methodologies are considered, most of the performance predictors still obtain good correlations. In particular, this result evidences that our proposed predictor are not so prone to the analysed biases like other simple predictors.

Finally, in Figure 6.3 we summarise the correlations found for the proposed predictors in each dimension – ratings, logs, and social. We have selected the most representative evaluation methodology (AR for rating and social data, and 1R for log



Correlation between rating-based user predictors and recommenders using AR methodology in MovieLens (Table 6.7).

Correlation between log-based user predictors and recommenders using 1R methodology in Last.fm (Table 6.14).

Correlation between social-based user predictors and recommenders using AR methodology in CAMRa collaborative (Table 6.17).

Figure 6.3. Heatmap of the correlation values between a subset of predictors and recommenders, using the most representative methodologies for the three considered spaces.

data) and a subset of the evaluated predictors and recommenders from each experiment, where the same information presented in Table 6.7, Table 6.14, and Table 6.17 (except for the average and median correlation values) is depicted in a more visual form. In particular, we may observe that predictors in MovieLens seem to be more redundant since the correlations are too similar.

From the figure we may also observe that in Last.fm and CAMRa datasets such redundancy is much lower and the predictors are quite different. Moreover, the first column and row (from the bottom) represent the recommender and predictor baselines, which serve as references from where the correlations should be analysed. In the three cases we can observe that most of the predictors obtain larger (darker) values than the count predictor. In the first case (rating-based predictors), however, it is clear that the correlation depends more on the recommender and less on the actual predictor.

6.6 Conclusions

We have proposed adaptations of query performance techniques from ad-hoc Information Retrieval to define performance predictors in Recommender Systems. Taking inspiration in the query predictor known as query clarity, we have defined and elaborated in the Recommender Systems domain several predictive models according to different formulations and assumptions. Furthermore, we propose performance predictors from theories and models of Information Theory, Social Graph Theory, and Information Retrieval based on three types of preference data: rating-based, log-based, and social-based.

We find several effective schemes with a high predictive power for recommender systems performance. We have proposed different ways for the adaptation of the query clarity predictor to recommender systems depending on the equivalences between the involved spaces. The clarity formulation is powerful because of its theoretical soundness, which is suitable to different domain-oriented adaptations. Hence, for rating-based preferences we use different expansions which take into account the rating values and the items rated by the user. For log-based preferences we exploit the co-occurrences of the items in the user profile and, more importantly, the temporal dimension, which allows for more principled functions such as the temporal Kullback-Leibler divergence or the user's autocorrelation. Finally, for social-based preferences we exploit the user's social network and different graph metrics are used apart from the user clarity based on the ratings. The results, as summarised in the previous section, are in general positive and provide evidences that the proposed functions are able to indeed predict the performance of user or items in recommender systems.

Furthermore, by analysising the behaviour of trivial predictors (such as the count of ratings in training and test) we have been able to uncover noisy biases or sensitivity to irrelevant variables in the way performance is measured. Irrelevant and uninteresting in the sense that it is not clear that the variations due to these variables really reflect actual differences in quality. As a result, we have used unbiased evaluation

methodologies where non trivial predictors still obtain positive results with respect to performance correlation.

As a side-effect, our study introduces an interesting revision of the gray sheep user concept. A simplistic interpretation of the gray sheep intuition would suggest that users with a too unusual behavior are a difficult target for recommendations. It appears however in our study that, on the contrary, users who somewhat distinguish themselves from the main trends in the community are easier to give well-performing recommendations. This suggests that perhaps the right characterisation of a gray sheep user might be one who has scarce overlap with other users. On the other hand, the fact that a clear user distinguishes herself from the aggregate trends does not mean that she does not have a sufficiently strong neighbourhood of similar users. In particular, this seems to indicate that users who follow mainstream trends are more difficult to be suggested successful items by a recommender system (at least, by a personalised one). In Information Retrieval, one can observe a similar trend: more ambiguous (mixture of topics) queries perform worse than higher-coherence queries (Cronen-Townsend et al., 2002).

In the future we plan to explore further performance predictors. Specifically, we are interested in incorporating explicit recommender dependence into the predictors, so as to better exploit the information managed by the recommender, allowing to the predictor a smoother adaptation to the recommender performance, and increasing the final correlation between them. Additionally, we are also interested in exploring alternative item-based predictors apart from those defined in this chapter, and, eventually, using other information sources such as log-based preference data and even the social network of the users who rated a particular item.

Part IV

Applications

It is through science that we prove, but
through intuition that we discover.
Jules Henri Poincaré

Chapter 7

Dynamic recommender ensembles

Hybrid recommender systems – and recommender ensembles as a particular case – have become a very popular strategy for making recommendations, since they help alleviate most of the shortcomings of the individual recommenders combined. They have, however, specific problems such as the need of deciding which information sources should be exploited, which recommenders should exploit each of these sources, and how the combination of recommenders should be configured.

In this chapter we propose a framework to decide how dynamic hybridisation should be balanced, by estimating its expected improvements on individual recommendations. Furthermore, we provide some requirements to decide when to build such hybridisation. Within the spectrum of hybrid recommendation approaches, we focus on those that linearly combine the output from several recommenders, and use different weights for generating a particular aggregation of the individual recommendations. In the standard approach, these weights are typically fixed regardless of the user for which recommendations are produced, or the recommended items. In this context we investigate the use of performance predictors to assign those weights dynamically depending on the target user or item. We evaluate our approach using the predictors proposed in the previous chapter. The results obtained show that the generated dynamic ensembles are capable of outperforming their static counterparts. Furthermore, they also show that dynamic ensembles can be improved if predictors with stronger predictive power (higher correlation values as observed in the previous chapter) are used.

In Section 7.1 we present and formulate the research problem of recommendation hybridisation. Next, in Section 7.2 we describe our proposed performance prediction framework for dynamic hybrid recommendation. Section 7.3 describes the experiments conducted and provide an overall discussion of the obtained results. Finally, in Section 7.4 some conclusions are given.

7.1 Problem statement

As described in Chapter 2, hybrid recommenders are built by the combination of different recommendation methods. In the simplest and typical case, hybrid recommendations are produced by weighting and summing the utility values output by some recommenders, forming a so called recommender ensemble where an arbitrary number of algorithms of different kinds (content-based, user-based collaborative filtering, item-based collaborative filtering, social-based, demographics-based, etc.) can be combined.

Researchers in Machine Learning have known for long that the combination of classifiers usually achieves better results than each method separately, which is also true in Recommender Systems – the Netflix prize has been a paradigmatic example of this, where all the top classified teams used large recommender ensembles. We focus on weighted hybrid approaches, as an option that begets a simple and general formulation of the dynamic balance of the combined methods R_k by just setting the weights λ_k of each method in the hybrid combination. This approach can be expressed as follows:

$$\tilde{r}(u,i) = \sum_k \lambda_k * \tilde{r}_{R_k}(u,i) \text{ s.t. } \sum_k \lambda_k = 1 \tag{7.1}$$

In this chapter we investigate whether the performance predictors proposed in the previous chapter – where we have already found degrees of correlation between the ambiguity (clarity) of the user's preferences and the accuracy of the system's recommendations – can be useful for hybridisation. Specifically, we aim to use these predictors to build **dynamic hybrid recommenders** in such a way that the weight λ_k depends not only on the recommender but also on the current user u, or potentially other variables such as the item i or other available context information. We propose to specify such weights according to the ambiguity of the user's preferences or item's patterns, that is, we aim to use the performance predictors defined in the Chapter 6 to estimate those weights.

In the next section we propose a framework to perform dynamic hybrid recommendation where we use recommendation performance predictors and we analyse different requirements related to the adaptation of such predictors to produce weights in a hybrid recommender combination. After that, three different experiments are presented, where the predictors proposed in Chapter 6 are used as dynamic weights in the combination.

7.2 A performance prediction framework for ensemble recommendation

Let us simplify Equation (7.1) to the case where only two recommenders $R1$ and $R2$ are used. In this situation, only one weighting factor λ is needed (because of the constraint for the weights to sum to one) and we would have the following formulation:

$$\tilde{r}(u,i) = \lambda * \tilde{r}_{R1}(u,i) + (1-\lambda) * \tilde{r}_{R2}(u,i) \tag{7.2}$$

In this case, since the λ weight is the same for every user u and item i we refer to such a recommender as a *static hybrid*. However, a single value of the combination parameter λ is not generally the optimal for each (user, item) pair. Therefore, instead of Equation (7.2), we may want to consider:

$$\tilde{r}(u,i) = \gamma_{R1}(u,i) * \tilde{r}_{R1}(u,i) + \gamma_{R2}(u,i) * \tilde{r}_{R2}(u,i) \tag{7.3}$$

where γ_R is the combination parameter which may depend on the current user, item, or both, and probably also depending on the recommender R. In this case we refer to such method as a *dynamic hybrid*.

A suitable assignment of the $\gamma(u,i)$ parameters is a difficult task. In our approach, however, we propose to use the performance prediction methodology developed in the previous chapter, whenever the predictors show some correlation with the performance of a recommender. In this way, since we have some evidence that the performance predictors are able to estimate in advance the performance of a user in a user or item basis, we can use such estimations to weight accordingly the ratings predicted for a given user and item pair by each recommender.

In this context, it is not granted in general to obtain improvements whenever a performance predictor is used in a dynamic ensemble. We have to devise a set of conditions in which such predictors may be used; moreover, the ensemble problem has to be well defined, which is not always true as we shall show. Hence, we define a framework for dynamic hybrid recommendation based on recommendation performance predictors, characterised by some prerequisites, a specific normalisation strategy, and a weighting distribution among recommenders. In this framework, the weights γ_R are obtained by transformations of the values obtained by a performance predictor, in a similar way as the work presented in (Yom-Tov et al., 2005b) on rank aggregation in Information Retrieval, but in the context of Recommender Systems.

7.2.1 Requirements

A first requirement to use a performance predictor for weighting the recommenders of an ensemble, is that it should correlate positively with the performance of not all

but some of such recommenders, or with the performance of all the recommenders but to different degrees. If a performance predictor correlates positively with all the recommenders in an ensemble to a similar extent, it does not provide a discriminative criteria to weight the recommenders any differently.

A predictor should be used to assign weights to those recommenders of the ensemble with which it correlates for performance. These assignments also alter the weights of the uncorrelated recommenders, since the weights of all the recommenders in the ensemble need to sum to 1. However, this should not affect the overall performance contribution of these recommenders, as the resulting weight should correspond randomly with their performance (hence the unpredicted recommenders' weight can be expected to change for good as much as for bad, whereas the weight of predicted recommenders should change more often for good).

Figure 7.1 shows which correlations can be considered valid according to the statements presented above, for an ensemble with two recommenders R1 and R2. The horizontal axis depicts the correlation with respect R1 and the vertical axis with R2. Hence, the dotted area represents those situations where a predictor's correlation for R1 is higher than for R2, and thus, the predictor should weight R1. Analogously, the striped area represents the candidate situations where the predictor should weight R2. Furthermore, when correlations with R1 and R2 are too similar (diagonal) no weighting assignment is preferred, and thus, if a predictor lies in the white area it should be used for weighting neither R1 nor R2 for the reasons described above.

Another requirement is that a recommender should not have an always superior or always inferior performance to those of the rest of the ensemble's recommenders. Otherwise the problem is distorted by the fact that the best weight is the one that gets closest to 0 for the recommenders that systematically perform worse (or 1 for the best), regardless of how excellent or terribly bad is the applied strategy, or the predictive power of the approach, since a biased predictor (either towards 0 or 1, depending on which recommender (the worst or the best) such predictor is weighting) would obtain very good results. This issue is recognised in (van Setten, 2005) where the author presents the situation where all recommenders produce item suggestions that are all too low or all too high with respect to the true user's preferences, and then the recommender ensemble is less accurate than the best individual recommender. In summary, underperforming recommenders are useless in an ensemble to begin with, or equivalently, the over performing one(s) should be used alone, and thus, there is no true weighting problem to solve.

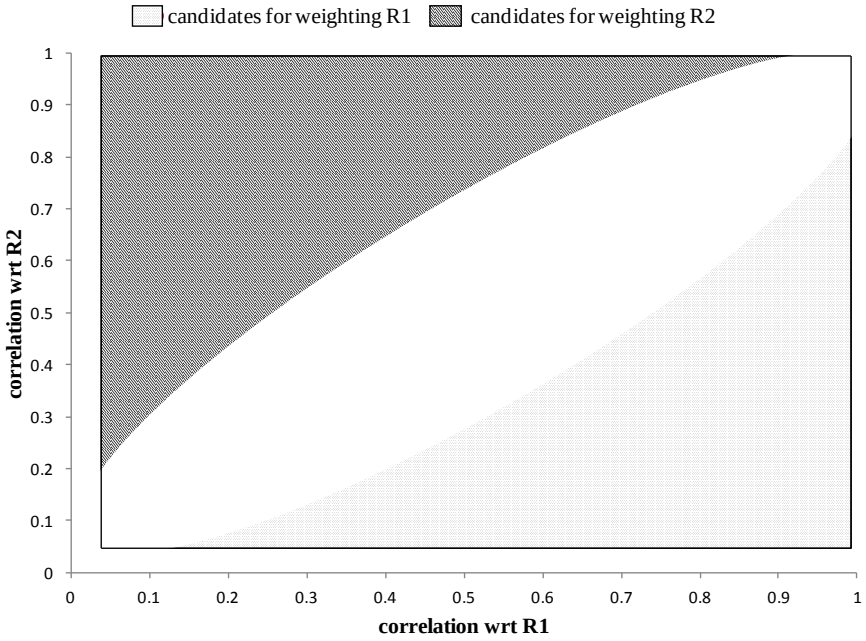


Figure 7.1. Valid predictor correlation regions for a recommender ensemble of size 2.

7.2.2 Predictor normalisation

The output of a predictor is required to correlate with the performance of a recommender, but it is not necessarily by itself a good value for weighting the recommender in an ensemble, as already pointed out in (Hauff et al., 2009). In order to generate appropriate weights, the predictor output should be transformed by a monotonic function into values on a comparable scale, such as simply $[0,1]$. We shall call this transformation "normalisation."

In this context, different transformations can be applied. Mapping the minimum value to 0 and the maximum to 1 is the simplest transformation, also known as *min-max* score normalisation (Renda and Straccia, 2003). Another common approach is to map (named *rank-sim* by Renda and Straccia, 2003) the predictor scores onto evenly distributed points in the $[0,1]$, preserving their order. Min-max preserves the original predictor score distribution, while rank-sim maps it onto a uniform distribution. There is no obvious a priori reason to decide which case is preferable, to preserve the original distribution, or to equalise it somehow, and in fact more complex normalisation techniques could be used, like the one proposed in (Fernández et al., 2006b).

7.2.3 Weight distribution among recommenders

Once the predictor output has been normalised, it still needs a final adjustment to ensure, among other things, that the sum of the weights assigned to the ensemble's

recommenders is 1. How this step is done depends, mainly, on how many recommenders are weighted by predictors, more specifically on whether all or only some of the combined recommenders are treated by performance predictors. Hence, we consider two options for the distribution of the weights among the recommenders:

a) Only some of the recommenders in the ensemble are given dynamic weights. The rest of the recommenders receive the same weight, ensuring the weights of the ensemble's recommenders sum up to 1. This can be done in different ways:
   - Assigning a weight of 0.5 to the unpredicted recommenders, and dividing all weights by the total sum. This strategy is named as *fixed weight* or **FW**.
   - Assigning the dynamic weights to the corresponding recommenders, if we assume that their sum is ≤ 1, then we divide 1 minus the sum of dynamic coefficients equally among the unpredicted recommenders. We denote this strategy as *one minus* or **OM**. If the sum is greater than 1, we have to divide by the total sum and normalise it by the total number of predictors.

b) All recommenders are weighted using a specific predictor per recommender. This is not easy to grant in general, as there may not be predictors for all the recommenders combined. In case this option is taken, the weights can be simply normalised by the sum of weights.

Furthermore, if the output of each recommender has a different range, it would be necessary to apply an additional normalisation step to the recommender scores. The most usual strategies are the ones described in the previous section: score or rank normalisation (Renda and Straccia, 2003).

7.3 Experimental results

We next report experiments assessing the usefulness of the proposed predictors for adjusting the weights of a recommender ensemble, once their predictive power has been confirmed against the recommenders' actual performance, as reported in the previous chapter. We identify the combinations of recommenders that meet the conditions stated in the previous section for the dynamic combination problem to make sense and select the performance predictors to be applied based on their observed correlation with the performance of the recommenders (as reported in Section 6.5), and the requirements proposed in this chapter, i.e., that one recommender in the ensemble should have a positive correlation with the predictor, and the other should have an opposite or near neutral correlation. Then, we compare dynamic against static ensembles.

Among the different ways to set up static ensembles of two recommenders we take as baselines a) the best performing one in test, and b) the best theoretical static one without prior information, i.e., one with $\lambda = 0.5$. Intuitively, an even weighting

is the optimum over the – theoretical – set of all recommender ensembles: if say $\lambda_B = 0.3$ was the best weight for the combination of two recommenders R1+R2, then $\lambda = 0.3$ should be fairly bad for the permutation R2+R1 ($\lambda = 1 - 0.3 = 0.7$ being best). If we assume that performance loss is convex with respect to $|\lambda - \lambda_B|$ – it can be seen that otherwise the hybrid may underperform its constituents –, then $\lambda = 0.5$ is the best compromise for R1+R2 and R2+R1. Since the set of all possible ensembles includes all the permutations of the combined recommenders, $\lambda = 0.5$ is the best (theoretical) overall weight.

We also take as "skylines" (upper bound baselines) an oracle performance predictor consisting of the performance of the recommender itself. We shall refer to this method as 'perfect correlation', where the true performance of both recommenders is used as a weight for hybridisation (hence, such predictor would have a correlation of 1.0 with the recommender's performance), whereas we shall refer to it as 'PC-OM' and 'PC-FW' when the performance of only one recommender is used (the same recommender being weighted by the predictors) along with the one minus or the fixed weight strategy for weight distribution (see Section 7.2.3). In all cases we apply a rank normalisation technique on the recommenders' scores.

In the subsequent sections we present three experiments conducted to evaluate the proposed performance predictors. In the first experiment we use the rating-based predictors and test both user- and item-based performance predictors presented in Section 6.2.1. We use the MovieLens dataset, and compare the results with four of the evaluation methodologies presented in Chapter 4, i.e., AR, 1R, P1R, and U1R. In the second experiment we use predictors based on log data. We evaluate the predictors presented in Section 6.2.2 on the two versions of the Last.fm dataset using the 1R methodology. Finally, in the third experiment we test the social-based predictors presented in Section 6.3 on the CAMRa dataset and the AR methodology.

7.3.1 Dynamic recommender ensembles on rating data

As a first instantiation of our framework for building dynamic recommender ensembles described in Section 7.2, we first have to identify the recommenders to combine, that is: one of the recommenders should have a positive correlation with the predictor, while the other should have an opposite or near neutral correlation; besides, they should not perform very differently.

According to the correlation results presented in Section 6.5.1, we identify the pairs of recommenders presented in Table 7.1 as combinations meeting the conditions stated above. The first three ensembles are combinations of a collaborative filtering with a content-based recommendation method. The last ensemble combines a user-based collaborative filtering method with a non-personalised method, and the rest of the ensembles are combinations of two collaborative filtering methods. Al-

	R1	R2
HRU1	TFL1	CB
HRU2	TFL2	CB
HRU3	kNN	CB
HRU4	kNN	IB
HRU5	kNN	pLSA
HRU6	kNN	ItemPop

Table 7.1. Selected recommenders for building dynamic ensemble using user performance predictors that exploit rating-based information (MovieLens dataset).

though some of these combinations have not been typical in the recommender systems literature, in our study they serve as a proof of concept to check whether the proposed dynamic recommender ensemble framework is useful in general or not. We refer the reader to Appendix A.2 for more details about the implementation of the recommenders.

The first two rows of Table 7.2, Table 7.3, Table 7.4, and Table 7.5 show the P@10 values for each of the combined recommenders obtained using the AR, 1R, U1R, and P1R methodologies, respectively. In Appendix A.5.1 we report results with other evaluation metrics. Note that, as mentioned in Chapter 4, in the AR methodology the absolute values are not meaningful since they depend on the amount of relevant information in test; on the other hand, for the 1R related methodologies (i.e., 1R, U1R, and P1R) the precision at 10 metric has an upper bound on 0.1, since there is only one relevant item in each ranking.

In these tables we may observe that among the six considered ensembles, there are cases where the first recommender (with respect to which the performance is predicted) performs better, worse, or similarly to the second recommender. This situation changes accross methodologies and provides for a comparison of the resulting effects when the stated requirements are not met. Analogously, the predictors' correlations may change depending on the evaluation methodology followed, as observed in Section 6.5.1. Specifically, the recommenders presented in Table 7.1 where chosen according to the correlation results obtained for the AR methodology, and we may observe that some of the conditions stated above do not hold for some of the selected cases, for instance, correlation between most of the predictors and kNN recommender is negligible in the 1R, U1R, and P1R methodologies, in contrast with the results found for the AR methodology.

In the tables we may also observe that the best static ensemble is different depending on the evaluation methodology and the combined recommenders. The performance values of the best static ensembles, on the other hand, show an interesting situation that does depend on the specific considered ensemble, namely, whether the (best) static ensembles outperform or not both recommenders. For the AR methodology (Table 7.2), in the case of HRU1, HRU3, HRU5, and HRU6, the best static

	HRU1	HRU2	HRU3	HRU4	HRU5	HRU6
R1 (λ=1.0)	0.0024	0.0696	0.0307	0.0307	0.0307	0.0307
R2 (λ=0.0)	0.0163	0.0163	0.0163	0.0001	0.1454	0.0897
Baseline (λ=0.5)	0.0106	0.0473	0.0363	0.0008	0.1142	0.0808
Best static (best λ)	0.0180 (0.1)	0.0668 (0.9)	0.0392 (0.9)	0.0078 (0.9)	0.1475 (0.1)	0.0937 (0.1)
Perfect correlation	0.0189	0.0732	0.0401	0.0311	0.1469	0.0980
PC-OM	0.0176	0.0721	0.0434	0.0091	0.1489	0.0958
PC-FW	0.0177	0.0541	0.0379	0.0025	0.1478	0.0958
Entropy-OM	**0.0110**▼△	**0.0685**△▲	**0.0388**▽▲	**0.0069**▼▲	0.1126▼▽	0.0791▼▼
ItemSimple-OM	**0.0170**▼▲	**0.0685**△▲	**0.0390**▽▲	**0.0072**▽▲	**0.1496**▲▲	**0.0919**▼▲
ItemUser-OM	**0.0172**▼▲	**0.0680**△▲	**0.0386**▽▲	**0.0068**▼▲	**0.1513**▲▲	**0.0924**▼▲
RatUser-OM	**0.0177**▽▲	**0.0687**△▲	**0.0393**△▲	**0.0072**▽▲	**0.1535**▲▲	**0.0931**▽▲
RatItem-OM	**0.0178**▽▲	**0.0674**△▲	**0.0389**▽▲	**0.0066**▼▲	**0.1542**▲▲	**0.0928**▽▲
IRUser-OM	**0.0169**▼▲	**0.0668**▲▲	**0.0387**▽▲	**0.0066**▼▲	**0.1487**▲▲	**0.0922**▼▲
IRItem-OM	**0.0172**▼▲	**0.0655**▽▲	**0.0378**▼▲	**0.0061**▼▲	**0.1500**▲▲	**0.0918**▼▲
IRUserItem-OM	**0.0170**▼▲	**0.0665**▽▲	**0.0388**▽▲	**0.0066**▼▲	**0.1498**▲▲	**0.0916**▼▲
Entropy-FW	**0.0111**▼▲	**0.0528**▼▲	**0.0369**▼△	**0.0027**▼▲	**0.1156**▼▲	0.0807▼▽
ItemSimple-FW	**0.0156**▼▲	**0.0529**▼▲	**0.0369**▼△	**0.0027**▼▲	**0.1433**▼▲	**0.0908**▼▲
ItemUser-FW	**0.0166**▼▲	**0.0529**▼▲	**0.0368**▼△	**0.0028**▼▲	**0.1468**▽▲	**0.0915**▼▲
RatUser-FW	**0.0170**▼▲	**0.0528**▼▲	**0.0370**▼△	**0.0028**▼▲	**0.1498**▲▲	**0.0919**▼▲
RatItem-FW	**0.0170**▼▲	**0.0529**▼▲	**0.0369**▼△	**0.0027**▼▲	**0.1499**▲▲	**0.0918**▼▲
IRUser-FW	**0.0161**▼▲	**0.0526**▼▲	**0.0371**▼△	**0.0029**▼▲	**0.1420**▼▲	**0.0912**▼▲
IRItem-FW	**0.0163**▼▲	**0.0525**▼▲	**0.0367**▼△	**0.0027**▼▲	**0.1459**▽▲	**0.0909**▼▲
IRUserItem-FW	**0.0164**▼▲	**0.0527**▼▲	**0.0372**▼△	**0.0028**▼▲	**0.1452**▽▲	**0.0908**▼▲

Table 7.2. Dynamic ensemble performance values (P@10) using AR methodology and user predictors (MovieLens dataset). Improvements over the baseline are in bold, the best result for each column is underlined. The value α of each dynamic hybrid is marked with α_y^x, where x and y indicate, respectively, statistical difference with respect to the best static (upper, x) and with respect to the baseline (lower, y). Moreover, ▲ and △ indicate, respectively, significant and non-significant improvements over the corresponding recommender. A similar convention with ▼ and ▽ indicates values below the recommender performance. Statistical significance is established by paired Wilcoxon $p < 0.05$ in all cases.

outperforms both recommenders, but this is not observed for HRU2 nor for HRU4. In the latter scenarios, thus, it seems hybridisation would not be so useful for combination.

Additionally, regarding the normalisation of the predictor's output we evaluate two normalisation techniques: rank and score normalisation. Since there is no prior information about which normalisation technique would provide better results, we test both, and report the best results in each situation, which are usually achieved by the rank-sim normalisation technique. Finally, the weigh strategy is also included as a parameter of the experiments. Since we only have a predictor for one of the recommenders in the ensemble (denoted as R1), as we explained in Section 7.2.3, we may weight the unpredicted recommender as one minus the predictor value (OM), or as 0.5 and then divide the weights of the two recommenders by the sum of weights (FW).

	HRU1	HRU2	HRU3	HRU4	HRU5	HRU6
R1 (λ=1.0)	0.0221	0.0690	0.0437	0.0437	0.0437	0.0437
R2 (λ=0.0)	0.0221	0.0221	0.0221	0.0074	0.0836	0.0649
Baseline (λ=0.5)	0.0338	0.0536	0.0469	0.0327	0.0749	0.0658
Best static	0.0338	0.0720	0.0514	0.0455	0.0856	0.0696
(best λ)	(0.4)	(0.9)	(0.8)	(0.9)	(0.1)	(0.2)
Perfect correlation	0.0370	0.0715	0.0553	0.0458	0.0840	0.0723
PC-OM	0.0358	0.0683	0.0507	0.0353	0.0811	0.0709
PC-FW	0.0343	0.0592	0.0482	0.0344	0.0803	0.0699
Entropy-OM	0.0332▼▼	**0.0662**▼▲	**0.0472**▼△	**0.0382**▼▲	0.0709▼▼	0.0626▼▼
ItemSimple-OM	0.0304▼▼	**0.0666**▼▲	**0.0473**▼▲	**0.0384**▼▲	**0.0844**▼▲	**0.0681**▼▲
ItemUser-OM	0.0305▼▼	**0.0660**▼▲	**0.0471**▼△	**0.0381**▼▲	**0.0847**▼▲	**0.0680**▼▲
RatUser-OM	0.0307▼▼	**0.0666**▼▲	**0.0478**▼▲	**0.0386**▼▲	**0.0850**▼▲	**0.0680**▼▲
RatItem-OM	0.0305▼▼	**0.0663**▼▲	**0.0475**▼▲	**0.0385**▼▲	**0.0849**▼▲	**0.0678**▼▲
IRUser-OM	0.0304▼▼	**0.0655**▼▲	**0.0470**▼△	**0.0381**▼▲	**0.0839**▼▲	**0.0675**▼▲
IRItem-OM	0.0298▼▼	**0.0644**▼▲	0.0457▼▼	**0.0370**▼▲	**0.0839**▼▲	**0.0671**▼▲
IRUserItem-OM	0.0305▼▼	**0.0655**▼▲	**0.0471**▼△	**0.0381**▼▲	**0.0841**▼▲	**0.0674**▼▲
Entropy-FW	**0.0339**△△	**0.0594**▼▲	**0.0472**▼△	**0.0356**▼▲	0.0686▼▼	0.0650▼▼
ItemSimple-FW	0.0321▼▼	**0.0596**▼▲	**0.0473**▼▲	**0.0358**▼▲	**0.0837**▼▲	**0.0684**▼▲
ItemUser-FW	0.0320▼▼	**0.0594**▼▲	**0.0471**▼△	**0.0356**▼▲	**0.0843**▼▲	**0.0683**▼▲
RatUser-FW	0.0321▼▼	**0.0596**▼▲	**0.0475**▼▲	**0.0359**▼▲	**0.0848**▼▲	**0.0684**▼▲
RatItem-FW	0.0321▼▼	**0.0595**▼▲	**0.0473**▼▲	**0.0358**▼▲	**0.0847**▼▲	**0.0684**▼▲
IRUser-FW	0.0320▼▼	**0.0592**▼▲	**0.0471**▼△	**0.0356**▼▲	**0.0834**▼▲	**0.0680**▼▲
IRItem-FW	0.0318▼▼	**0.0588**▼▲	0.0465▼▼	**0.0349**▼▲	**0.0835**▼▲	**0.0674**▼▲
IRUserItem-FW	0.0320▼▼	**0.0592**▼▲	**0.0471**▼△	**0.0356**▼▲	**0.0837**▼▲	**0.0678**▼▲

Table 7.3. Dynamic ensemble performance values (P@10) using 1R methodology and user predictors (MovieLens dataset).

Table 7.2 shows the results obtained following the AR methodology. We may observe how, except in three cases, dynamic ensembles outperform the baseline. Interestingly, for HRU5, the best performing method is not the one obtained with the 'perfect correlation' approach, as we may expect, but with our dynamic ensembles based on the user clarity performance predictors. This is due to the fact that the corresponding predictor for the first recommender (P@10 values for kNN) also has a strong correlation with the performance of the second recommender (pLSA), and thus, it does not satisfy the requirement that the correlation values should not be too similar for both recommenders.

Table 7.3 shows the results obtained with the 1R methodology. Note that in this case the correlations were consistently lower than those obtained with the AR methodology. In particular, this is emphasised in the results of the dynamic ensemble HRU1, which do not outperform the baseline for almost any predictor. This can be explained with the results reported in Table 6.9, where the TFL1 recommender obtains a near-zero correlation, and thus, the correlation requirement of our framework is not satisfied. Specifically, this fact highlights the importance of the strength in the correlation between the predictor and the recommender performance, as stated in Section 7.2.1. Furthermore, we may observe in the table that for two combinations

	HRU1	HRU2	HRU3	HRU4	HRU5	HRU6
R1 (λ=1.0)	0.0294	0.0524	0.0381	0.0381	0.0381	0.0381
R2 (λ=0.0)	0.0223	0.0223	0.0223	0.0068	0.0718	0.0406
Baseline (λ=0.5)	0.0345	0.0440	0.0396	0.0283	0.0639	0.0493
Best static (best λ)	0.0351 (0.6)	0.0536 (0.9)	0.0424 (0.7)	0.0384 (0.9)	0.0732 (0.1)	0.0493 (0.5)
Perfect correlation	0.0389	0.0552	0.0493	0.0396	0.0742	0.0559
PC-OM	0.0373	0.0485	0.0471	0.0332	0.0732	0.0548
PC-FW	0.0355	0.0459	0.0429	0.0307	0.0722	0.0535
Entropy-OM	0.0345	**0.0518**	**0.0404**	**0.0337**	0.0615	0.0471
ItemSimple-OM	0.0333	**0.0519**	**0.0403**	**0.0339**	**0.0723**	0.0444
ItemUser-OM	0.0334	**0.0517**	**0.0403**	**0.0336**	**0.0726**	0.0438
RatUser-OM	0.0335	**0.0521**	**0.0410**	**0.0341**	**0.0728**	0.0435
RatItem-OM	0.0334	**0.0516**	**0.0406**	**0.0341**	**0.0726**	0.0434
IRUser-OM	0.0333	**0.0511**	**0.0401**	**0.0336**	**0.0718**	0.0440
IRItem-OM	0.0326	**0.0504**	0.0388	**0.0325**	**0.0714**	0.0430
IRUserItem-OM	0.0334	**0.0511**	**0.0401**	**0.0336**	**0.0719**	0.0437
Entropy-FW	**0.0347**	**0.0472**	**0.0402**	**0.0308**	0.0636	0.0486
ItemSimple-FW	0.0342	**0.0473**	**0.0402**	**0.0309**	**0.0720**	0.0467
ItemUser-FW	0.0342	**0.0471**	**0.0401**	**0.0308**	**0.0724**	0.0467
RatUser-FW	0.0343	**0.0474**	**0.0405**	**0.0310**	**0.0727**	0.0469
RatItem-FW	0.0342	**0.0472**	**0.0403**	**0.0309**	**0.0725**	0.0469
IRUser-FW	0.0341	**0.0470**	**0.0401**	**0.0308**	**0.0714**	0.0469
IRItem-FW	0.0338	**0.0467**	0.0393	**0.0302**	**0.0712**	0.0464
IRUserItem-FW	0.0341	**0.0471**	**0.0401**	**0.0308**	**0.0716**	0.0469

Table 7.4. Dynamic ensemble performance values (P@10) using the U1R methodology and user predictors (MovieLens dataset)

(HRU2 and HRU5) the best performance results are not obtained by dynamic approaches, but by the best static approaches in contrast with what we found for the AR methodology. This situation is different to the one obtained when we evaluate using MAP@10 (see Appendix A.4.1), where the best results are always obtained by dynamic ensembles.

Table 7.4 and Table 7.5 show the performance values obtained with the unbiased methodologies proposed in Chapter 4, that is, U1R and P1R. Following the U1R methodology (Table 7.4) we obtain similar results to those obtained in the 1R methodology except for HRU6. In contrast, with the P1R methodology (Table 7.5) our framework does not show improvements over any baseline. We may see that the 'perfect correlation' methods are able to obtain better, although very close, values than those of the best static ensemble. This means that there is room for improvement in this methodology, and that the performance of the dynamic recommender ensembles could be improved if better performance predictors were found.

	HRU1	HRU2	HRU3	HRU4	HRU5	HRU6
R1 (λ=1.0)	0.0203	0.0348	0.0265	0.0265	0.0265	0.0265
R2 (λ=0.0)	0.0197	0.0197	0.0197	0.0208	0.0604	0.0282
Baseline (λ=0.5)	0.0470	0.0579	0.0539	0.0269	0.0763	0.0560
Best static	0.0470	0.0593	0.0541	0.0278	0.0796	0.0560
(best λ)	(0.5)	(0.6)	(0.6)	(0.7)	(0.4)	(0.5)
Perfect correlation	0.0464	0.0579	0.0546	0.0314	0.0767	0.0564
PC-OM	0.0425	0.0554	0.0528	0.0296	0.0746	0.0537
PC-FW	0.0429	0.0542	0.0504	0.0282	0.0764	0.0522
Entropy-OM	0.0431▼▼	0.0564▼▼	0.0502▼▼	0.0261▼▼	0.0698▼▼	0.0521▼▼
ItemSimple-OM	0.0358▼▼	0.0509▼▼	0.0429▼▼	0.0261▼▼	0.0689▼▼	0.0441▼▼
ItemUser-OM	0.0361▼▼	0.0512▼▼	0.0431▼▼	0.0261▼▼	0.0675▼▼	0.0444▼▼
RatUser-OM	0.0362▼▼	0.0514▼▼	0.0436▼▼	0.0263▼▼	0.0663▼▼	0.0446▼▼
RatItem-OM	0.0361▼▼	0.0511▼▼	0.0432▼▼	0.0262▼▼	0.0661▼▼	0.0444▼▼
IRUser-OM	0.0365▼▼	0.0513▼▼	0.0435▼▼	0.0263▼▼	0.0687▼▼	0.0447▼▼
IRItem-OM	0.0357▼▼	0.0504▼▼	0.0421▼▼	0.0257▼▼	0.0669▼▼	0.0439▼▼
IRUserItem-OM	0.0365▼▼	0.0513▼▼	0.0434▼▼	0.0263▼▼	0.0675▼▼	0.0447▼▼
Entropy-FW	0.0457▼▼	0.0577▼▼	0.0524▼▼	0.0265▼▼	0.0745▼▼	0.0546▼▼
ItemSimple-FW	0.0410▼▼	0.0540▼▼	0.0475▼▼	0.0266▼▼	0.0720▼▼	0.0498▼▼
ItemUser-FW	0.0409▼▼	0.0538▼▼	0.0473▼▼	0.0265▼▼	0.0706▼▼	0.0497▼▼
RatUser-FW	0.0410▼▼	0.0540▼▼	0.0477▼▼	0.0267▼▼	0.0691▼▼	0.0499▼▼
RatItem-FW	0.0411▼▼	0.0541▼▼	0.0476▼▼	0.0266▼▼	0.0688▼▼	0.0499▼▼
IRUser-FW	0.0410▼▼	0.0538▼▼	0.0474▼▼	0.0266▼▼	0.0721▼▼	0.0496▼▼
IRItem-FW	0.0406▼▼	0.0534▼▼	0.0467▼▼	0.0263▼▼	0.0699▼▼	0.0491▼▼
IRUserItem-FW	0.0409▼▼	0.0538▼▼	0.0474▼▼	0.0266▼▼	0.0706▼▼	0.0496▼▼

Table 7.5. Dynamic ensemble performance values (P@10) using the P1R methodology and user predictors (MovieLens dataset).

In summary, the **results show that our methods significantly outperform static ensembles for different recommender combinations in most of the evaluation methodologies**. Moreover, in most cases our methods also achieve the best results for each ensemble, let aside the performance of the oracle performance prediction (perfect correlation) and best static approaches, which use groundtruth (test) information, differently to the clarity- and entropy-based performance predictors.

Nevertheless, we observe that in those cases where the dynamic ensembles do not perform better than the static ensembles, the best static approaches use values of λ close to 0.5. We hypothesise that our framework may be biased towards favouring those ensembles whose recommender combination is highly unbalanced. Interestingly, although the predictors only weight one of the recommenders (not always the better performing one) a dynamic ensemble is usually able to find the optimal combination in the unbalanced cases. In particular, this could help to answer why our dynamic ensembles underperform static approaches for the U1R and P1R methodologies, since the best static in these cases seem to be often very close to 0.5.

	R1	R2
HRI1	pLSA	CB
HRI2	pLSA	kNN
HRI3	ItemPop	CB
HRI4	ItemPop	kNN

Table 7.6. Selected recommenders for building dynamic ensembles using item predictors that exploit rating data (MovieLens dataset).

Using item-based predictors

As we noted in Section 6.5.2, item-based predictors could also be valuable since they also obtain high correlations with respect to item perfomance. Table 7.6 shows the selected recommenders that satisfy the correlation requirements with item predictors. Table 7.7, Table 7.8, and Table 7.9 show the results obtained when these recommender combinations are evaluated and compared against dynamic versions (using our proposed item predictors), and using the 1R, U1R, and uuU1R methodologies. In this case, ensemble predictions are computed by means of Equation (7.3) with values $\gamma(u,i)$ only depending on the current item, that is, $\gamma(i)$.

When measuring the performance of dynamic ensembles that use item-based performance predictors, we do not compute the perfect correlation predictors because we do not have a standard metric for item performance. Apart from that, the

	HRI1	HRI2	HRI3	HRI4
R1 (λ=1.0)	0.0836	0.0836	0.0649	0.0649
R2 (λ=0.0)	0.0221	0.0437	0.0221	0.0437
Baseline (λ=0.5)	0.0909	0.0924	0.0886	0.0907
Best static (best λ)	0.0909 (0.5)	0.0924 (0.5)	0.0886 (0.5)	0.0907 (0.5)
Entropy-OM	0.0708▼▼	0.0858▼▼	0.0684▼▼	0.0831▼▼
UserSimple-OM	0.0761▼▼	0.0905▼▼	0.0723▼▼	0.0837▼▼
UserItem-OM	0.0776▼▼	0.0903▼▼	0.0749▼▼	0.0843▼▼
RatItem-OM	0.0751▼▼	0.0893▼▼	0.0712▼▼	0.0824▼▼
RatUser-OM	0.0759▼▼	0.0892▼▼	0.0674▼▼	0.0789▼▼
URItem-OM	0.0776▼▼	0.0911▼▼	0.0797▼▼	0.0885▼▼
URUser-OM	0.0781▼▼	0.0906▼▼	0.0721▼▼	0.0820▼▼
URItemUser-OM	0.0777▼▼	0.0909▼▼	0.0777▼▼	0.0869▼▼
Entropy-FW	0.0798▼▼	0.0923▼▼	0.0771▼▼	0.0895▼▼
UserSimple-FW	**0.0946**▲▲	**0.0979**▲▲	**0.0916**▲▲	**0.0949**▲▲
UserItem-FW	**<u>0.0949</u>**▲▲	**0.0980**▲▲	**0.0920**▲▲	**0.0950**▲▲
RatItem-FW	**0.0944**▲▲	**0.0979**▲▲	**0.0913**▲▲	**0.0948**▲▲
RatUser-FW	**0.0946**▲▲	**0.0978**▲▲	**0.0908**▲▲	**0.0942**▲▲
URItem-FW	**0.0940**▲▲	**<u>0.0981</u>**▲▲	**<u>0.0923</u>**▲▲	**<u>0.0958</u>**▲▲
URUser-FW	**0.0946**▲▲	**0.0978**▲▲	**0.0912**▲▲	**0.0945**▲▲
URItemUser-FW	**0.0944**▲▲	**0.0980**▲▲	**0.0921**▲▲	**0.0954**▲▲

Table 7.7. Dynamic ensemble performance values (P@10) using <u>1R methodology</u> with item predictors (MovieLens dataset).

rest of the experimental settings is the same as those described above for dynamic hybrids with user-based performance predictors.

Table 7.7 shows the results obtained by using item-based predictors and the 1R methodology. We may observe that if the predictors are weighted using the FW strategy, dynamic ensembles outperform static combinations in every situation, except for the Entropy predictor. It is interesting to note that, differently to user-based predictors, the dynamic ensembles are able to outperform the best static ensemble even when they are close to the baseline with $\lambda = 0.5$. The reader may compare Table 7.4 and Table 7.7 to observe these differences.

In Table 7.8, where the methodology U1R is used, a very similar situation occurs, although not all dynamic ensembles outperform the static approach with the FW strategy. Specifically, the dynamic hybrid weighted by the URItem clarity predictor clearly obtains better performance than the rest of the dynamic and static ensembles, in particular the HRI3 and HRI4 combinations.

Finally, the performance results found for the uuU1R methodology are presented in Table 7.9, in which the test ratings – i.e., the users – are uniformly distributed over the items, items previously uniformly distributed in the test (like in the U1R methodology). In this experiment, the performance of the dynamic ensemble is much better than in the previous experiments, since **all the rating-based item predictors (except for the Entropy predictor) outperform the static baseline no matter the weighting strategy in three out of four recommender combinations**.

	HRI1	HRI2	HRI3	HRI4
R1 (λ=1.0)	0.0718	0.0718	0.0406	0.0406
R2 (λ=0.0)	0.0223	0.0381	0.0223	0.0381
Baseline (λ=0.5)	0.0764	0.0812	0.0630	0.0689
Best static (best λ)	0.0764 (0.5)	0.0812 (0.5)	0.0630 (0.5)	0.0689 (0.5)
Entropy-OM	0.0571▼▼	0.0652▼▼	0.0435▼▼	0.0508▼▼
UserSimple-OM	0.0657▼▼	0.0716▼▼	0.0399▼▼	0.0450▼▼
UserItem-OM	0.0671▼▼	0.0721▼▼	0.0425▼▼	0.0462▼▼
RatItem-OM	0.0645▼▼	0.0699▼▼	0.0392▼▼	0.0435▼▼
RatUser-OM	0.0620▼▼	0.0671▼▼	0.0335▼▼	0.0382▼▼
URItem-OM	0.0705▼▼	0.0757▼▼	0.0496▼▼	0.0532▼▼
URUser-OM	0.0650▼▼	0.0699▼▼	0.0372▼▼	0.0414▼▼
URItemUser-OM	0.0690▼▼	0.0741▼▼	0.0462▼▼	0.0500▼▼
Entropy-FW	0.0668▼▼	0.0757▼▼	0.0518▼▼	0.0595▼▼
UserSimple-FW	**0.0840**▲▲	**0.0886**▲▲	0.0601▼▼	0.0658▼▼
UserItem-FW	**0.0844**▲▲	**0.0887**▲▲	0.0609▼▼	0.0663▼▼
RatItem-FW	**0.0839**▲▲	**0.0883**▲▲	0.0598▼▼	0.0653▼▼
RatUser-FW	**0.0831**▲▲	**0.0876**▲▲	0.0573▼▼	0.0630▼▼
URItem-FW	**0.0851**▲▲	**0.0897**▲▲	**0.0642**▲▲	**0.0698**▲▲
URUser-FW	**0.0836**▲▲	**0.0881**▲▲	0.0585▼▼	0.0642▼▼
URItemUser-FW	**0.0848**▲▲	**0.0893**▲▲	0.0625▼▼	0.0680▼▼

Table 7.8. Dynamic ensemble performance values (P@10) using U1R methodology with item predictors (MovieLens dataset).

	HRI1	HRI2	HRI3	HRI4
R1 (λ=1.0)	0.0536	0.0536	0.0225	0.0225
R2 (λ=0.0)	0.0198	0.0275	0.0198	0.0275
Baseline (λ=0.5)	0.0374	0.0440	0.0239	0.0256
Best static (best λ)	0.0491 (0.9)	0.0502 (0.9)	0.0239 (0.6)	0.0271 (0.2)
Entropy-OM	0.0324▼▼	0.0385▼▼	0.0236▽▽	**0.0280**▲▲
UserSimple-OM	**0.0510**△△	**0.0548**▲▲	0.0237▽▽	**0.0282**▲▲
UserItem-OM	**0.0514**▲▲	**0.0547**▲▲	0.0236▽▽	**0.0280**▲▲
RatItem-OM	**0.0516**▲▲	**0.0547**▲▲	0.0237▽▽	**0.0281**▲▲
RatUser-OM	**0.0523**▲▲	**0.0551**▲▲	0.0237▽▽	**0.0282**▲▲
URItem-OM	**0.0498**▲▲	**0.0536**▲▲	0.0234▼▼	**0.0280**▲▲
URUser-OM	**0.0518**▲▲	**0.0551**▲▲	0.0234▼▼	**0.0279**▲▲
URItemUser-OM	**0.0505**▲▲	**0.0542**▲▲	0.0235▽▼	**0.0280**▲▲
Entropy-FW	0.0344▼▼	0.0410▼▼	**0.0241**△△	**0.0275**▲▲
UserSimple-FW	**0.0435**△△	**0.0503**△▲	**0.0244**▲▲	**0.0276**▲▲
UserItem-FW	**0.0435**▼▲	**0.0501**△▲	**0.0245**▲▲	**0.0275**▲▲
RatItem-FW	**0.0436**▼▲	**0.0504**△▲	**0.0244**▲▲	**0.0275**▲▲
RatUser-FW	**0.0440**▼▲	**0.0509**▲▲	**0.0245**▲▲	**0.0276**▲▲
URItem-FW	**0.0429**▼▲	**0.0494**▲▲	**0.0244**▲▲	**0.0273**△▲
URUser-FW	**0.0438**▼▲	**0.0506**△▲	**0.0245**▲▲	**0.0274**△▲
URItemUser-FW	**0.0432**▼▲	**0.0498**△▲	**0.0245**▲▲	**0.0274**△▲

Table 7.9. Dynamic ensemble performance values (P@10) using uuU1R methodology with item predictors (MovieLens dataset).

In the other combination (HRI3) the best strategy is FW, the same as with the other evaluation methodologies.

7.3.2 Dynamic recommender ensembles on log data

In this section we present experiments in which log-based predictors are used to dynamically weight an ensemble's recommenders. As with rating-based information, in this case we first have to select suitable recommenders to combine according to the requirements established in our framework. Hence, we choose the combinations HL1, HL2 and HL3 presented in Table 7.10, where, as before, the performance predictors weight the recommender denoted as R1.

The Last.fm dataset contains timestamped log-based information. As noted in Chapter 4, for efficiency reasons, we only use the 1R methodology in this dataset. Table 7.11 shows the results obtained with a temporal split of the data, and Table 7.12 shows the results obtained with a random split (five-fold) of the data.

	R1	R2
HL1	kNN	CB
HL2	kNN	ItemPop
HL3	pLSA	kNN

Table 7.10. Selected recommenders for building dynamic ensembles using performance predictors that exploit log-based information (Last.fm dataset).

	HL1	HL2	HL3
R1 (λ=1.0)	0.0603	0.0603	0.0926
R2 (λ=0.0)	0.0916	0.0797	0.0603
Baseline (λ=0.5)	0.0852	0.0755	0.0820
Best static	0.0914	0.0812	0.0925
(best λ)	(0.2)	(0.1)	(0.9)
Perfect correlation	0.0890	0.0783	0.0863
PC-OM	0.0869	0.0771	0.0851
PC-FW	0.0849	0.0751	0.0826
ItemSimple-OM	**0.0904**▼▲	**0.0804**▼▲	**0.0901**▼▲
Autocorrelation-OM	0.0815▼▼	0.0722▼▼	0.0781▼▼
TimeSimple-OM	**0.0905**▼▲	**0.0789**▼▲	**0.0898**▼▲
ItemTime-OM	**0.0906**▼▲	**0.0804**▼▲	**0.0902**▼▲
ItemPriorTime-OM	**0.0885**▼▲	**0.0778**▼▲	**0.0863**▼▲
ItemSimple-FW	**0.0903**▼▲	**0.0802**▼▲	**0.0891**▼▲
Autocorrelation-FW	0.0842▼▲	0.0746▼▼	0.0809▼▼
TimeSimple-FW	**0.0901**▼▲	**0.0785**▼▲	**0.0884**▼▲
ItemTime-FW	**0.0904**▼▲	**0.0800**▼▲	**0.0891**▼▲
ItemPriorTime-FW	**0.0883**▼▲	**0.0775**▼▲	**0.0855**▼▲

Table 7.11. Dynamic ensemble performance values (P@10) using the 1R methodology with the log-based user predictors (Last.fm, temporal split).

	HL1	HL2	HL3
R1 (λ=1.0)	0.0204	0.0204	0.0836
R2 (λ=0.0)	0.0828	0.0767	0.0204
Baseline (λ=0.5)	0.0764	0.0643	0.0704
Best static	0.0818	0.0767	0.0837
(best λ)	(0.2)	(0.1)	(0.9)
Perfect correlation	0.0818	0.0760	0.0829
PC-OM	0.0816	0.0755	0.0823
PC-FW	0.0815	0.0745	0.0811
ItemSimple-OM	**0.0799**▼▲	**0.0730**▼▲	**0.0771**▼▲
Autocorrelation-OM	0.0717▼▼	0.0596▼▼	0.0686▼▼
TimeSimple-OM	**0.0814**▼▲	**0.0762**▼▲	0.0518▼▼
ItemTime-OM	**0.0806**▼▲	**0.0734**▼▲	**0.0761**▼▲
ItemPriorTime-OM	**0.0770**▼▲	**0.0658**▼▲	**0.0743**▼▲
ItemSimple-FW	**0.0804**▼▲	**0.0726**▼▲	**0.0739**▼▲
Autocorrelation-FW	0.0756▼▼	0.0631▼▼	0.0697▼▼
TimeSimple-FW	**0.0814**▼▲	**0.0753**▼▲	0.0579▼▼
ItemTime-FW	**0.0808**▼▲	**0.0728**▼▲	**0.0732**▼▲
ItemPriorTime-FW	**0.0783**▼▲	**0.0671**▼▲	**0.0719**▼▲

Table 7.12. Dynamic ensemble performance values (P@10) using the 1R methodology with log-based user predictors (Last.fm, five-fold random split).

We can see that the results of both tables are analogous. **The dynamic ensembles weighted by the log-based performance predictors outperform the baseline static ensemble in all cases, except with the Autocorrelation predictor**. This result is consistent with the correlations presented in Table 6.14 and Table 6.15, where autocorrelation obtained the lowest (absolute) correlation value for the kNN recommender on both versions of the dataset. Regarding the pLSA recommender (in the combination HL3), the Autocorrelation and TimeSimple predictors obtain com-

	R1	R2
HS1	Personal	pLSA
HS2	Personal	kNN
HS3	PureSocial	pLSA
HS4	PureSocial	kNN

Table 7.13. Selected recommenders for building dynamic ensembles using social-based user predictors (CAMRa dataset).

parable correlations with the combined recommenders, yet the performance of the corresponding dynamic ensembles is very different, thus suggesting that, although we have found a dependence between the predictors' power in terms of correlation, and their effectiveness in weighting hybrids, this is not a strict necessary condition to obtain improvements over the static ensembles.

The best performance values were achieved either by single recommenders or by the best static ensembles. When the best results are obtained by single recommenders emphasises the fact that no hybridisation is required for that combination (like in HL1 and HL3 for the temporal split, and HL1 and HL2 for the random split). In the other case, when the best results are achieved by the best static ensembles, it may restrict the usefulness of our approach, although our proposed dynamic ensembles significantly outperform the baseline static ensembles for some predictors such as TimeSimple and ItemSimple. We have to recall that the best static ensembles are in fact optimised using the test set, which is clearly not a fair comparison. The results of the perfect correlation ensembles in the random split are always better than those obtained by the performance predictors, confirming that predictors with stronger correlations should obtain better performance results when used for dynamic ensembles.

7.3.3 Dynamic recommender ensembles on social data

In the third experiment we exploit the social information available in the CAMRa dataset to combine collaborative and social filtering recommenders using social-based performance predictors. Table 7.13 shows the recommender combinations selected based on the correlations obtained in Section 6.5.4. Here, we present 4 ensembles where the two social filtering recommenders, Personal and PureSocial, are combined with two collaborative filtering recommenders, pLSA and kNN. We saw in Section 6.5.4 that most of the social-based predictors obtained higher correlations with the social filtering recommenders, and lower or negligible correlations with the collaborative filtering recommenders, at least for the social version of the dataset (Table 6.16). The situation for the collaborative-social version was not so clear, but for the sake of coherence, we use the same set of ensembles in both versions of the dataset.

	HS1	HS2	HS3	HS4
R1 (λ=1.0)	0.1732	0.1732	0.1760	0.1760
R2 (λ=0.0)	0.1110	0.0473	0.1110	0.0473
Baseline (λ=0.5)	0.1813	0.1821	0.2006	0.1929
Best static (best λ)	0.1842 (0.7)	0.1899 (0.8)	0.2012 (0.4)	0.1952 (0.6)
Perfect correlation	0.2018	0.1929	0.2089	0.1979
PC-OM	0.1872	0.1875	0.2048	0.1946
PC-FW	0.1863	0.1869	0.2042	0.1994
AvgNeighDeg-OM	0.1795▽▽	**0.1896**▽△	0.1973▽▽	0.1804▼▼
BetCentrality-OM	0.1744▼▽	0.1804▼▽	0.1833▼▼	0.1777▼▼
ClustCoeff-OM	0.1786▽▽	0.1786▼▽	0.1836▼▼	0.1753▼▼
Degree-OM	0.1738▼▽	**0.1839**▽△	0.1976▽▽	0.1765▼▼
EgoCompSize-OM	0.1756▼▽	**0.1833**▽△	0.1967▽▽	0.1827▼▼
HITS-OM	0.1774▽▽	**0.1911**△▲	0.1813▼▼	0.1798▼▼
PageRank-OM	0.1762▼▽	**0.1842**▽△	0.1917▼▽	0.1801▼▼
TwoHopNeigh-OM	0.1756▼▽	**0.1851**▽△	0.1964▽▽	0.1777▼▼
AvgNeighDeg-FW	0.1807▽▽	**0.1896**▽▲	0.2003▽▽	0.1914▽▽
BetCentrality-FW	0.1801▽▽	**0.1872**▽△	**0.2024**△△	**0.1929**▽△
ClustCoeff-FW	0.1804▽▽	**0.1875**▽▲	0.2003▽▽	0.1890▼▽
Degree-FW	0.1798▽▽	**0.1887**▽▲	0.2000▽▽	**0.1929**▽△
EgoCompSize-FW	0.1789▼▽	**0.1896**▽▲	**0.2009**▽△	**0.1938**▽△
HITS-FW	0.1801▽▽	**0.1902**△▲	0.1997▽▽	0.1926▽▽
PageRank-FW	0.1810▽▽	**0.1875**▽△	0.2003▽▽	0.1923▽▽
TwoHopNeigh-FW	0.1801▽▽	**0.1905**△▲	0.2000▽▽	0.1926▽▽

Table 7.14. Dynamic ensemble performance values (P@10) using the AR methodology with social-based user predictors (CAMRa, social dataset).

As we mentioned in Section 6.5.4, due to the lack of coverage of the social filtering recommenders, the only methodology that provides sensible results is the AR methodology. In this section we present the results obtained using this methodology on the two available versions of the CAMRa dataset: social and collaborative-social.

Table 7.14 shows the results obtained on the social version of the CAMRa dataset. We see that only for one out of the four recommender combinations, the dynamic ensembles consistently outperform the baseline static ensemble. However, it is interesting to note that the best value is always achieved by the perfect correlation ensemble, which means that further improvements could be possible if we were able to find predictors with stronger correlations.

In the collaborative-social version of the dataset (Table 7.15) the results are similar, except that now for HS2, the best result is obtained by the best static ensemble. Moreover, a larger number of dynamic ensembles outperform the baseline static ensemble HS3, whereas at least one dynamic ensemble outperforms the baseline HS1, which is a better result than the one shown in the previous Table 7.14. We hypothesise this is because on this version of the dataset the individual recommenders display a more similar performance to each other (compare the differences between R1 and R2 in Table 7.14 and Table 7.15).

Furthermore, some of the correlations obtained for the CAMRa collaborative dataset are more discriminative between the combined recommenders, in the sense that, for instance, the correlations between the two-hop neighbourhood predictor and the Personal recommender were -0.123 and -0.121 in the social and collaborative-social datasets, respectively. However, the correlations between the two-hop neighbourhood predictor and kNN were 0.004 and 0.130, that is, in the second dataset the relative distance in correlation between these two recommenders is larger, according to the correlation with respect to the predictor. This change in the correlations may explain the fact that in Table 7.15 some of the dynamic ensembles outperform the perfect correlation ensemble, which does not take the relative correlation into account with respect to each individual recommender, as noted in 7.3.1.

In general, **the HITS predictor obtains the best results among the dynamic ensembles for some of the tested combinations. Other predictors such as the betweenness centrality and the ego components size produce more competitive ensembles in the social version of the dataset**, whereas the degree and the average neighbour degree preditors provide better results for more than one combination in the CAMRa collaborative dataset.

	HS1	HS2	HS3	HS4
R1 (λ=1.0)	0.1066	0.1066	0.1072	0.1072
R2 (λ=0.0)	0.1007	0.0226	0.1007	0.0226
Baseline (λ=0.5)	0.1509	0.1142	0.1599	0.1219
Best static	0.1524	0.1200	0.1632	0.1219
(best λ)	(0.4)	(0.7)	(0.3)	(0.5)
Perfect correlation	0.1608	0.1188	0.1640	0.1237
PC-OM	0.1202	0.1164	0.1254	0.1199
PC-FW	0.1189	0.1143	0.1263	0.1219
AvgNeighDeg-OM	0.1489▽▽	**0.1195**▽△	0.1599▽▽	0.1131▼▼
BetCentrality-OM	0.1443▼▼	0.1132▽▽	0.1487▼▼	0.1114▼▼
ClustCoeff-OM	0.1465▼▽	0.1123▽▽	0.1483▼▼	0.1108▼▼
Degree-OM	0.1472▽▽	**0.1154**▽△	**0.1614**▽△	0.1107▼▼
EgoCompSize-OM	0.1461▽▽	**0.1158**▽△	0.1596▽▽	0.1140▼▼
HITS-OM	0.1485▽▽	**0.1200**▲	0.1467▼▼	0.1134▼▼
PageRank-OM	0.1471▼▽	**0.1167**▽△	0.1579▽▽	0.1123▼▼
TwoHopNeigh-OM	0.1478▽▽	**0.1171**▽△	0.1585▼▽	0.1118▼▼
AvgNeighDeg-FW	**0.1518**▽△	**0.1191**▽▲	**0.1623**▽△	0.1204▼▼
BetCentrality-FW	0.1491▼▽	**0.1180**▽▲	0.1577▼▽	0.1213▼▼
ClustCoeff-FW	0.1500▽▽	**0.1182**▽▲	0.1566▼▼	0.1189▼▼
Degree-FW	0.1489▽▽	**0.1191**▽▲	**0.1627**▽△	0.1208▼▼
EgoCompSize-FW	0.1489▽▽	**0.1193**▽▲	**0.1618**▽△	0.1210▼▼
HITS-FW	0.1482▼▼	**0.1195**▽▲	0.1564▼▼	0.1202▼▼
PageRank-FW	0.1491▼▽	**0.1186**▽▲	**0.1610**▽△	0.1211▼▼
TwoHopNeigh-FW	0.1500▽▽	**0.1195**▽▲	**0.1619**▽△	0.1211▼▼

Table 7.15. Dynamic ensemble performance values (P@10) using the AR methodology with social-based user predictors (CAMRa, collaborative dataset).

7.3.4 Discussion

The analysis of the results presented in this chapter shows that ensembles can indeed benefit from a dynamic weighting of their recommenders. In particular, we have seen that when these weights come from performance predictors, which previously had shown significant correlation with the performance of individual recommenders, the resulting dynamic ensemble tends to outperform static combinations of the recommenders. In this context, in order to obtain successful hybridisations, we have to take several variables into account, which correspond to three stages proposed in our framework: the correlation between the predictor and the combined recommenders, the relative performance of such recommenders, the strategy to normalise the predictor's values, and the weight distribution among recommenders.

The relative performance of the recommenders has proven to be decisive, since in some cases, hybridisation does not make sense to begin with, when the difference in performance between the recommenders is significant and systematic, and thus, dynamic ensembles cannot obtain the best performance result, although they may outperform static ensembles. Performance prediction normalisation and weight distribution, on the other hand, do make a difference in the results. Although no explicit results are presented in this work regarding different normalisation approaches, previously conducted experiments showed us that score normalisation produce worse results than rank normalisation. Finally, the weight distribution strategy is not as critical as other stages of our framework, but helps to obtain much better results, specifically, when the one minus strategy (OM) is used.

The obtained results have also shown that more complex formalisations and probability models do not necessarily lead to better results, with respect to the adaptation and definition of the user and item clarity performance predictors. In this adaptation, various configurations were available, and we experimented with further extensions of different language models for the same clarity model, using rating and log-based information. Additionally, several graph-based metrics were tested, where the concept of the user's strength in a social network is modelled in different ways.

We find that different formulations for the user-based performance clarity predictor consistently obtain the best results in different situations for rating-based preference information. We also experimented with item-based predictors, and found that the UserItem, URItem, and RatUser predictors were noticeably better than the rest of the formulations. When log-based information is exploited, the ItemTime and TimeSimple predictors obtained better results than other predictors not based on the clarity concept, such as the Autocorrelation function. Moreover, regarding the social-based ensembles, the HITS, two-hop neighbourhood, and average neighbour degree approaches clearly outperform the ensemble weighted by the rest of the predictors and, in most of the cases, also outperform the baseline static ensemble.

These results are, in general, consistent with the correlation values between the predictors' output values and the recommenders' performance values. Figure 7.2 shows a summary of the results presented in this and previous chapters, where the difference in correlation is plotted against the gain (or loss) in performance with respect to the baseline. For this figure, the best and worst dynamic ensembles were selected from Table 7.2, Table 7.11 and Table 7.15. In the figure we may observe the trend that the larger the difference in correlation, the better the improvement over the baseline, which is in concordance with the requirement that both correlations should not be very similar. These results provide some insights in order to understand which features may help configure well performing dynamic recommender ensembles, where performance predictors have emerged as a clear useful characteristic.

7.4 Conclusions

In this chapter we have explored how the performance of a recommender ensemble can be improved by dynamically assigning the weights of its recommenders, by analysing the performance correlation between the values of a performance predictor and the performance of an individual recommender. In this way, we have proposed a dynamic hybrid framework that let decide when and how dynamic hybridisation should be done.

Drawing from the performance predictors proposed in the previous chapter, we have conducted several experiments in order to assess whether recommender en-

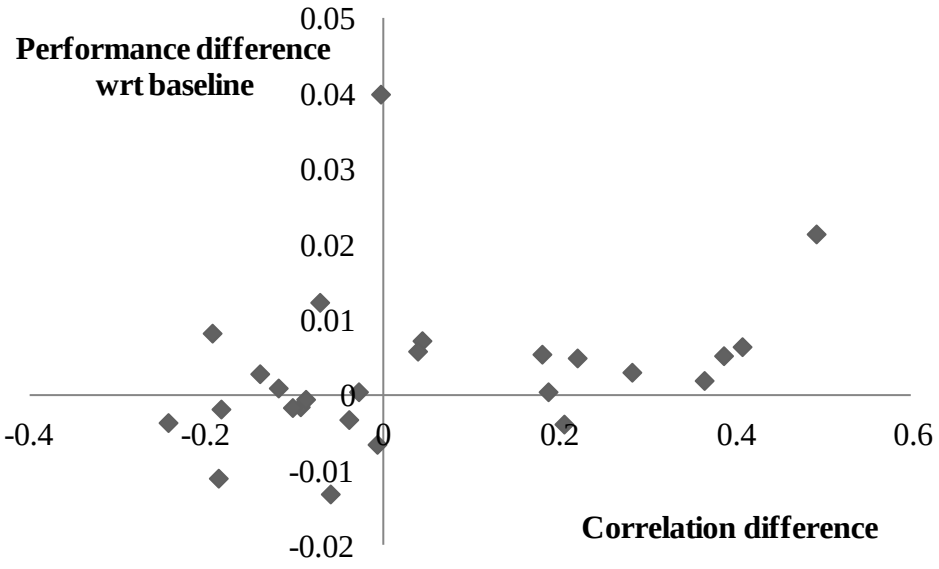


Figure 7.2. For each best and worst dynamic ensemble in Table 7.2, Table 7.11 and Table 7.15, this graph plots the difference in correlation between each predictor and a recommender against the difference in performance between the ensemble and the baseline.

sembles can benefit from dynamic weights according to such predictors. The results obtained in our experiments indicate that a strong correlation with performance tends to correspond with enhancements in ensembles by using the predictor for weight adjustment. The dynamic ensembles usually outperformed the baseline static ensemble for different recommender combinations, supporting their effectiveness in different situations.

In future work we aim to evaluate our framework with more than two recommenders in an ensemble, and more than one performance predictor, eventually, one for each recommender. We also plan to test different normalisation strategies of the predictor's values, where several assumptions about the ideal weight distribution can be verified, such as whether the user's rating distribution or the recommender's output are beneficial for the final performance of the ensemble. Moreover, Machine Learning approaches could also be used to learn the best weights in a user (or item) basis. Despite being more time consuming, these techniques may also achieve good results in terms of performance of the dynamic ensemble, although they are usually more prone to overfit the learned weights.

Chapter 8

Neighbour selection and weighting in user-based collaborative filtering

User-based recommender systems suggest interesting items to a user relying on similar-minded people called neighbours. The selection and weighting of the input from these neighbours characterise different variants of the approach. Thus, for instance, while standard user-based collaborative filtering strategies select neighbours based on user similarities, trust-aware recommendation algorithms rely on other aspects indicative of user trustworthiness and reliability.

In this chapter we restate the user-based recommendation problem, generalising it in terms of performance prediction techniques. We investigate how to adopt this generalisation to define a unified framework where we conduct an objective analysis of the effectiveness (predictive power) of neighbour scoring functions. We evaluate our approach with several state-of-the-art and novel neighbour scoring functions on two publicly available datasets. The notion of performance takes here a different nuance from previous chapters. More precisely, we consider the notion of neighbour performance, for which we propose several measures and new predictors. In an empirical comparison involving four neighbour quality metrics and thirteen performance predictors, we find a strong predictive power for some of the predictors with respect to certain metrics. This result is then validated by checking the final performance of recommendation strategies where predictors are used for selecting and/or weighting user neighbours. As a result, we are able to anticipate which predictors will perform better in neighbour scoring powered versions of a user-based collaborative filtering algorithm.

In Sections 8.1 and 8.2 we present a unified formulation and the proposed framework for neighbour selection and weighting in user-based recommendation, and in Section 8.3 we describe how the different neighbour scoring functions proposed in the literature fit into the framework. Finally, in Section 8.4 we present an experimental evaluation of the framework, and in Section 8.5 we provide conclusions.

8.1 Problem statement

We focus on user-based collaborative filtering algorithms, one type of memory-based approaches that explicitly seek people – commonly called **neighbours** – having preferences (and/or other characteristics of interest) in common with the target user, and use such preferences to predict item ratings for the user. User-based algorithms are built on the principle that a particular user's rating records are not equally useful to all other users as input to provide them with item suggestions (Herlocker et al., 2002). Therefore, as stated in Chapter 2, central aspects to these algorithms are a) how to identify which neighbours form the best basis to generate item recommendations for the target user, and b) how to properly make use of the information provided by them. Once the target user's neighbours are selected, the more similar a neighbour is to the user, the more her preferences are taken into account as input to produce recommendations.

A common user-based recommendation approach consists of predicting the relevance of an item for the target user by a linear combination of her neighbours' ratings, which are weighted by the similarity between the target user and her neighbours, as presented in Equation (2.3). For the sake of clarity, and since we shall later elaborate from it, we reproduce here the above equation:

$$\tilde{r}(u,i) = \bar{r}(u) + C \sum_{v \in N_k(u,i)} sim(u,v)\big(r(v,i) - \bar{r}(v)\big) \tag{8.1}$$

User similarity has been the central criterion for neighbour selection in most of the user-based collaborative filtering approaches (Desrosiers and Karypis, 2011). Nonetheless, recently it has been suggested that additional factors could have a valuable role to play on this point. For instance, two users with a high similarity value may no longer be reliable predictors for each other at some point because of a divergence of tastes over time (O'Donovan and Smyth, 2005). Thus, in the context of user-based collaborative filtering, more complex methods have been proposed in order to effectively select and weight useful neighbours (O'Donovan and Smyth, 2005; Desrosiers and Karypis, 2011). In this context a particularly relevant dimension relates the above additional factors with the general concept of trust (trustworthiness, reputation) on a user's contribution to the computation of recommendations. Hence, a number of trust-aware recommender systems have been proposed in the last decade (Hwang and Chen, 2007; O'Donovan and Smyth, 2005; Golbeck, 2009).

Most of these systems focus on the improvement of accuracy metrics, such as the Mean Average Error, by defining different heuristic trust functions, which, in most cases, are applied either as additional weighting factors in the neighbourhood-based formulation, or as a component of the neighbour selection criteria. The way trust is measured is considerably diverse in the literature. In fact, the notion of trust

has embraced a wide scope of neighbour aspects, spanning from personal trust on the neighbour's faithfulness, to trust on her competence, confidence in the correctness of the input data, or the effectiveness of the recommendation resulting from the neighbour's data. More specifically, in trust-aware recommender systems, a trust model is defined and, typically, introduced into the Resnick's equation (Equation (8.1)) either as an additional weight or as a filter for the potential user's neighbours. Moreover, depending on the nature of their input, different types of trust-aware recommendation approaches can be distinguished: rating-based approaches, and social-based approaches (using a trust network).

One of the first works that proposed *rating-based trust metrics* between users is (O'Donovan and Smyth, 2005). In that work O'Donovan and Smyth propose to modify how the "recommendation partners" (neighbours) are weighted and selected in the user-based collaborative filtering formula. They argue that the trustworthiness of a particular neighbour should be taken into account in the computed recommendation score by looking at how reliable her past recommendations were. Trust values are computed by measuring the amount of correct recommendations in which a user has participated as a neighbour, and then they are used for weighting the influence (along with computing the similarity), and selecting the target user's neighbours. Weng et al. (2006) propose an asymmetric trust metric based on the expectation of other users' competence in providing recommendations to reduce the uncertainty in predicting new ratings. The metric is used in the standard collaborative filtering formula instead of the similarity value. Two additional metrics are defined in (Kwon et al., 2009) based on the similarity between the ratings of a neighbour and the ratings from the community. Finally, Hwang and Chen (2007) define two trust metrics (local and global) by averaging the prediction error of co-rated items between a user and a potential neighbour.

Social-based trust metrics make use of explicit trust networks of users, built upon friendship relations (Massa and Avesani, 2004; Massa and Bhattacharjee, 2004) and explicit trust scores between individuals in a system (Ma et al., 2009; Walter et al., 2009). These metrics and, to some extent, their inherent meanings, are different with respect to rating-based metrics. Nonetheless, Ziegler and Lausen (2004) conduct a thorough analysis that shows empirical correlations between trust and user similarities, suggesting that users tend to create social connections with people who have similar preferences. Once such a correlation is proved, techniques based on social-based trust can be applicable. Golbeck and Hendler (2006) propose a metric called TidalTrust to infer trust relationships by using recursive search. Inferred trust values are used for every user who has rated a particular item in order to select only those users with high trust values. Then, a weighted average between past ratings and inferred trust values provides the predicted ratings. Massa and Avesani (2007b) ex-

periment with local (MoleTrust) and global (PageRank) trust metrics, showing that trust-based recommenders are very valuable for cold start users.

The research presented here seeks to provide an algorithmic generalisation for a significant variety of notions, computational definitions, and roles of trust in neighbour selection. Specifically, we aim to provide a theoretical framework for neighbour selection and weighting in which trust metrics can be defined and evaluated in terms of improvements on a final recommender's performance. We cast the rating prediction task – typically based, as described above, on the aggregation of neighbour preferences – into a framework for dynamic combination of inputs, from a performance prediction perspective, borrowing from the methodology for this area in the Information Retrieval field. The application of this perspective is not trivial, and requires a definition of what the performance of a neighbour means in this context. Hence, restated the problem in these terms, we propose to adapt and exploit techniques and methodologies developed in Information Retrieval for predicting query performance; in our case the target user's neighbours are equivalent to the queries, and our goal is to predict which of these neighbours will perform better for the target user.

Furthermore, since our framework provides an objective measure of the neighbour scoring function efficiency, we would be able to obtain a better understanding of the whole recommendation process. For instance, if the results obtained when a particular function is introduced in a recommender are not consistent with the (already observed) objective performance measures, it would mean that the chosen strategy is not the most appropriate, suggesting to experiment with further strategies, providing such a function has already shown some predictive power.

Therefore, the main contribution of our framework is that it provides a formal setting for the evaluation of neighbour selection and weighting functions, while, at the same time, enables to discriminate whether recommendation performance improvements are achieved by the neighbour scoring functions, or by the way these functions are used in the recommendation computation. Besides, our framework provides an unification of state-of-the-art trust-based recommendation approaches, where trust metrics are casted as neighbour performance predictors. As a result, in this chapter, we shall propose four neighbour quality metrics and thirteen performance predictors, defined upon a specific neighbour (user-based), a neighbour and the current user (user-user), or a neighbour and the current item (user-item). We shall generalise the different strategies proposed in the literature to introduce trust into collaborative filtering. Moreover, thanks to the proposed formulation, we will define and evaluate new strategies.

8.2 A performance prediction framework for neighbour scoring

8.2.1 Unifying neighbour selection and weighting in user-based Recommender Systems

From the observation that most of the methods for neighbour selection and weighting are elaborated upon the standard Resnick's scheme (Equation (8.1)), we propose a unified formulation as follows. Let us suppose, for the sake of generality, that we have a neighbour scoring function $s(u, v, i)$ that may depend on the target user u, a neighbour v, and a target item i. This function outputs a higher value whenever the user, neighbour, item, or a combination of them, is more trustworthy (in the case of trust models), or is expected to perform better as a neighbour according to the information available in the system, such as other ratings and external information, like a social network. Using this function we generalise Equation (8.1) to:

$$\tilde{r}(u,i) = \bar{r}(u) + C \sum_{v \in f^{neigh}(u,i;k;s)} f^{agg}\big(s(u,v,i), sim(u,v)\big)\big(r(v,i) - \bar{r}(v)\big) \quad (8.2)$$

where the function f^{neigh} denotes the selection of the set of neighbours, and f^{agg} is an aggregation function combining the output of s and the user similarity into a single weight value. In this way, we integrate the neighbour scoring function s into the Resnick's formula in order to: a) select the neighbours to be considered, instead of or in addition to the most similar users (via function f^{neigh}), and b) provide a general weighting scheme by introducing an aggregation function f^{agg} between the actual neighbour score and the similarity between the target user and her neighbours. Note that it is not required that s is bounded, since a constant C would normalise the output rating value. The function s is thus a core component in the generalisation of the user-based collaborative filtering techniques. It may embody similarity in itself (in such case f^{agg} may just return its first input argument), but sim and f^{agg} are left to simplify the connection with the original similarity-only formulation, and to suit particular cases where s applies other principles distinct to similarity.

The aggregation function f^{agg} can take different definitions, some examples of which can be found in the literature. For instance, O'Donovan and Smyth (2005) initially propose to use the arithmetic mean of the neighbour score (x) and the similarity (y; henceforth denoted as f_1^{agg}), and end up using the harmonic mean (f_2^{agg}) because of its better robustness to large differences in the inputs. In (Bellogín and Castells, 2010), on the other hand, we use the product function (f_3^{agg}). Moreover, Hwang and Chen (2007) propose to directly use the neighbour score as the weight

given to neighbours, that is, they use the projection function $f_4^{agg}(x,y) = x$. Obviously, the original Resnick's formulation can be expressed as the symmetric projection function $f_0^{agg}(x,y) = y$.

The neighbourhood selection embodied in function f^{neigh} also generalises Resnick's approach – the latter corresponds to the particular case $f_0^{neigh}(u,i;k;s) = N_k(u,i)$, where the neighbour scoring function is ignored, and only similarity is used. The general form admits different instantiations. In (Golbeck and Hendler, 2006) only the users with the highest trust values are selected as neighbours. In (O'Donovan and Smyth, 2005), on the other hand, those users whose trust values exceed a certain threshold are taken into consideration. This threshold is empirically defined as the mean across all the obtained values for each pair of users. The latter strategy can be formulated as follows:

$$f_1^{neigh}(u,i;k;s) = \{v \in N_k(u,i): s(u,v,i) > \tau\};\ \tau = \frac{1}{|\{(u,v,i)\}|}\sum_{(u,v,i)} s(u,v,i)$$

There are, nonetheless, some considerations to take into account when using specific combinations of neighbour weighting and neighbour selection functions. First, if f_4^{agg} is used together with f_0^{neigh} – only considering the most similar users in the neighbourhood –, then less reliable users (with low f_4^{agg}) who are very similar to the current user would be penalised, and more reliable neighbours but less similar to the current user are ignored, since they do not belong to the neighbourhood. Second, when using f_0^{agg} together with f_1^{neigh}, neighbours are weighted by their similarities with the target user. These similarities, however, could be very low, and thus, non-similar but reliable neighbours would be penalised. Finally, if f_4^{agg} is used with f_1^{neigh}, the similarity weight will not be considered at any point in the recommendation process.

Some of these configurations may deserve further investigation, and are considered in Section 8.4, along with other combinations not listed here.

8.2.2 Neighbour selection and weighting as a performance prediction problem

Neighbour scoring and selection can be seen as a task of predicting the effectiveness of neighbours as input for collaborative recommendations. In this section we elaborate and adapt the performance prediction framework presented in Chapter 5 to the problem of neighbour selection and weighting.

The same as performance prediction in Information Retrieval, which has been used to optimise rank aggregation (Yom-Tov et al., 2005a), in our proposed framework each user's neighbour can be considered as a retrieval subsystem (or criterion)

whose output is combined to form a final system's output (the recommendations) to the user.

For user-based collaborative filtering algorithms, the estimation $\tilde{r}(u,i)$ of the preference of the target user u for a particular item i can be formulated as an aggregation function of the ratings of some other users $\hat{V}$:

$$\tilde{r}(u,i) \propto \text{aggr}_{v \in \hat{V}}\big(sim(u,v);\ r(v,i); \bar{r}(u); \bar{r}(v)\big) \tag{8.3}$$

where $\hat{V}$ denotes the selected neighbours for a particular user u according to function f^{neigh} (see Equation (8.2)). As observed in (Adomavicius and Tuzhilin, 2005), different aggregation functions can be defined, but the most typical one is the weighted average function presented in the previous section.

In the previous function the term $\tilde{r}(u,i)$ can be seen as a retrieval function that aggregates the outputs of several utility subfunctions $r(v,i) - \bar{r}(v)$, each corresponding to a recommendation obtained from a neighbour of the target user. The combination of utility values is defined as a linear combination (translated by $\bar{r}(u)$) of the neighbours' ratings, weighted by their similarity $sim(u,v)$ with the target user. Hence, the computation of utility values in user-based filtering is equivalent to a typical rank aggregation model of Information Retrieval, where the aggregated results may be enhanced by predicting the performance of the combined recommendation outputs. In fact, the similarity value can be seen as a prediction of how useful a neighbour's advice is expected to be for the target user, which has proved to be a quite effective approach. The question is whether other performance factors beyond user similarity can be considered in a way that further enhancements can be drawn, as research on user trust awareness has attempted to prove in the last years.

The Information Retrieval performance prediction view provides a methodological approach, which we propose to adapt to the neighbour selection problem. The approach provides a principled path to drive the formulation, development and evaluation of effective neighbour selection and weighting techniques, as we shall see. In the proposed view, the selection/weighting problem is expressed as an issue of neighbour performance, as an additional factor (besides user similarity) to automatically tune the neighbours' contribution to the recommendations, according to the expected goodness of their advice. As summarised in Section 5.1, there are three core concepts in the performance prediction problem as addressed in the Information Retrieval literature: performance predictor, retrieval quality assessment, and predictor quality assessment. Since we are dealing with the prediction of which users may perform better as neighbours, the above three concepts can respectively be translated into *neighbour performance predictor*, *neighbour quality*, and *neighbour predictor quality*. For the sake of simplicity, let us assume we can define a performance predictor as a function that receives as input a user profile u (in general, it could receive other users or items as well), the set of items $\mathcal{I}_u$ rated by that user, and the collection S of ratings and

items (or any other user preference and item description information) available in the system. Then, following the notation given used in Chapter 5, we define a neighbour performance prediction function as:

$$\hat{\mu}(u) \leftarrow \gamma(u, \mathcal{I}_u, S). \tag{8.4}$$

The function γ can be defined in different ways, for instance, by taking into account the rating distribution of each user, the number of ratings available in the system, and the (implicit or explicit) relations made by that user with the rest of the community. Essentially, the neighbour performance predictor is intended to estimate the true neighbour quality metric, denoted as $\mu(u)$, which is typically measured using groundtruth information about whether the neighbour's influence is positive. The application of this perspective is not trivial, and requires, in particular, a definition of what the performance of a neighbour means in this context – where no standard metric for neighbour performance is yet available in the literature.

Once the estimated neighbour performance prediction values $\hat{\mu}(u_n)$ are computed for all users, the quality of the prediction can be measured as presented in Section 5.4.2, that is, either by measuring the correlation between the estimations and the real values $\mu(u_n)$, or by using classification accuracy metrics such as the F-measure. Since in this case we are interested in providing a ranking of users, this relates more with the traditional query performance task, and not with query difficulty (see Section 5.4.1), where the latter metrics are used. In other words, the neighbour predictor quality metric is defined as the following correlation:

$$q(\gamma) = \text{corr}([\hat{\mu}(u_1), \cdots, \hat{\mu}(u_n)], [\mu(u_1), \cdots, \mu(u_n)]). \tag{8.5}$$

Similarly to the situation in Information Retrieval, this correlation provides an assessment of the prediction accuracy (Carmel and Yom-Tov, 2010); the higher its (absolute) value, the higher the predictive power of γ. Moreover, the sign of $q(\gamma)$ represents whether the two involved variables – neighbour prediction and neighbour quality – are directly or inversely correlated.

Besides validating any proposed predictor by checking the correlation between predicted outcomes and objective metrics, we may further test the effectiveness of the defined predictors by introducing and testing a dynamic variant of user-based collaborative filtering. In this variant, the weights of neighbours are dynamically adjusted based on their expected effectiveness, along with the decision of which users belong to each neighbourhood, as in the general formulation presented in Equation (8.2). We propose to define the neighbour scoring function $s(u, v, i)$ based on the values computed from each neighbour performance predictors.

Hence, the basic idea of the framework presented here is to formally treat the neighbour selection and weighting in memory-based recommendation as a performance prediction problem. The performance prediction framework provides a principle basis to analyse whether the predictors are capturing some valuable, measurable

characteristic known to be useful for prediction, independently from their latter use in a recommendation strategy. Furthermore, if a neighbour scoring function with strong predictive power is introduced into the recommendation process and the performance is not improved, then, new ways of introducing such predictor into the rating estimation should be tested (either for selection or weighting), since we have some confidence that this function captures interesting user's characteristics, valuable for recommendation.

8.3 Neighbour quality metrics and performance predictors

The performance prediction research methodology requires a means to compare the predicted performance with the observed performance. This comparison is typically conducted in terms of some one-dimensional functional values, where the performance is assessed by some specific metric and the prediction can be translated to a certain numeric value. This value quantifies the expected degree of effectiveness, providing, thus, a relative magnitude.

Whereas in the context of performance prediction in IR, standard metrics of system effectiveness in response to a query are used for this purpose, in the case of predicting the performance of a neighbour for recommendation we would require to use metrics that measure how effective a neighbour is. In this section we propose several neighbour quality metrics and performance predictors which we shall evaluate in Section 8.4.

8.3.1 Neighbour quality metrics

The purpose of effectiveness predictors in our framework is to assess how useful specific neighbour profiles are as a basis for predicting ratings for the target user. Each predictor has to be contrasted to a measure of how "good" the neighbour's contribution is to the global community of users in the system. In contrast with query performance prediction, where a well established array of metrics are used to quantify query performance, to the best of our knowledge, in the literature there is not an equivalent function for neighbours used in user-based collaborative filtering. We therefore need to introduce and propose some sound candidate metrics.

Ideally, in the proposed framework, a quality metric should take the same arguments as the predictor, and thus, if we have, for instance, a user-item predictor, we should also be able to define a quality metric that depends on users and items. In general, we shall focus on user-based predictors, but it would be possible to explore item-based alternatives. Furthermore, we shall consider metrics taking neighbours as single input, independently from which neighbourhood is involved (i.e., independ-

ently from the target user), and which item is recommended. At the end of this section, nonetheless, we shall introduce a neighbour quality metric suitable for the user-user scenario, where both the target user and neighbour are taken into account.

Now, we propose three different neighbour quality metrics. The first two metrics had a different intended use by their authors, but we found they could be useful to evaluate how good a user is as a neighbour. The third metric was proposed by us in (Bellogín and Castells, 2010), where the problem of neighbour performance was explicitly addressed.

Rafter et al. (2009) propose two metrics in order to examine whether the neighbours have any influence in the recommendation accuracy. Both metrics are based on the comparison between true ratings and a neighbour's estimation of the ratings, as a way to measure the direction of the neighbour estimation and the average absolute magnitude of the shift produced by this estimation. Thus, the larger the neighbour's influence, the better her performance, according to our definition of a "good" neighbour. In this context we use those metrics as follows:

$$\mu_1 = \mu(v) = \frac{1}{|T_v|}\sum_{i \in T_v} \frac{1}{|N_k^{-1}(v;i)|} \sum_{w \in N_k^{-1}(v;i)} |r(w,i) - r(v,i)|$$

$$\mu_2 = \mu(v) = \frac{1}{|T_u|}\sum_{i \in T_v} \frac{1}{|N_k^{-1}(u;i)|} \sum_{w \in N_k^{-1}(v;i)} \delta\big([sgn(r(w,i) - \bar{r}(v)) = sgn(r(v,i) - \bar{r}(v))]; 1\big)$$

where δ is a binary function whose output is 1 if its arguments are true, and 0 otherwise. Metric μ_1 represents the **absolute error deviation** of a particular user, and μ_2 is the **sign of error deviation**. Note that $N_k^{-1}(v;i)$ denotes an inverse neighbourhood, which represents those users for whom v is a neighbour, and T_v denotes the items rated by user v in the test set. We can observe how each of these metrics represents a different method to measure how accurate the user v is as a neighbour.

In (Bellogín and Castells, 2010) we proposed a metric named **neighbour goodness**, which is defined as the difference in performance of the recommender system when including vs. excluding the user (i.e., her ratings) from the dataset. For instance, based on the mean average error standard metric, neighbour goodness can be instantiated as:

$$\mu_3 = \mu(v) = \frac{1}{|R_{\mathcal{U}\setminus\{v\}}|} \sum_{w \in \mathcal{U}\setminus\{v\}} \left[CE_{\mathcal{U}\setminus\{v\}}(w) - CE_{\mathcal{U}}(w)\right]$$

$$CE_X(v) = \sum_{i \in \mathcal{I}, r(v,i) \neq \emptyset} |\tilde{r}_X(v,i) - r(v,i)|$$

where $\tilde{r}_X(v,i)$ represents the predicted rating computed using only the data in X. This metric quantifies how much a user affects (contributes to or detracts from) the

total amount of mean average error of the system, since it is computed in the same way as that metric, but leaving out the user of interest – in the first term, the user is completely omitted; in the second term, the user is only involved as a neighbour. In this way we measure how a user contributes to the rest of users, or put informally, how better or worse the "world" is in the sense of how well recommendations work with and without the user. Hence, if the error increases when the user is removed from the dataset, it is considered as a good neighbour.

Based on the same idea of the previous metric, we propose a user-user quality metric that measures how one particular user affects to the error of another user when acting as her neighbour:

$$\mu_4 = \mu(u, v) = CE_{U\setminus\{v\}}(u) - CE_U(u)$$

We call this metric **user-neighbour goodness**. It quantifies the difference in user u's error when neighbour v is not in the system against the error when such neighbour is present, that is, it measures how much each neighbour contributes to reduce the error of a particular user.

8.3.2 Neighbour performance predictors

Having formulated neighbour selection in memory-based recommendation as a task of neighbour effectiveness prediction, and having proposed effectiveness metrics to compare against, the core of an approach to this problem is the definition of effectiveness predictors. For this purpose, similarity functions and trust models such as those mentioned in Section 8.1 can be directly used, since in trust-aware recommendation, trust metrics aim at measuring how reliable a neighbour is when introduced in the recommendation process (O'Donovan and Smyth, 2005). Interestingly, some of them only depend on one user (**global trust metrics**), and others depend on a user and an item or another user (**local trust metrics**). Furthermore, other authors have proposed different indicators for selecting good neighbours, mainly based on the overlap between the user and her neighbour, without considering the concept of trust.

We thus distinguish three types of neighbour performance predictors: **user predictors** – equivalent to the global trust metrics –, **user-item predictors**, and **user-user predictors** – equivalent to the local trust metrics. Note that, although trust metrics could now be interpreted as neighbour performance predictors, the proposed performance prediction framework let us to provide an inherent value to these metrics (identified as performance predictors), independently from whether they improve a recommender's performance when used for selecting or weighting in the specific collaborative filtering algorithm. This is due to the fact that it is possible to empirically check the quality of the prediction by analysing their correlation with respect to the neighbour performance metric, prior to the integration in any collabora-

tive filtering method. Thus, each predictor would obtain an explicit score that represents its predictive power, related to our *a priori* confidence on whether such predictor is capturing the neighbour's reliability or trustworthiness.

In the following we propose an array of neighbour effectiveness prediction methods, by adapting and integrating trust functions from the literature into our framework, and we also propose novel prediction functions.

User Predictors

User predictors are performance predictors that only depend on the target neighbour. When that neighbour is predicted to perform well, her assigned weight in the user-based collaborative filtering formulation is high.

One of the first user trust metrics proposed in the literature is the **profile-level trust** (O'Donovan and Smyth, 2005), which is defined as the percentage of correct recommendations in which a user has participated as a neighbour. If we denote the set of recommendations in which a user has been involved as

$$\text{RecSet}(u) = \{(v, i): u \in N_k(v; i)\},$$

then the predictor is defined as follows:

$$\gamma_1(u, v, i) = \gamma(v) = \frac{|\text{CorrectSet}(v)|}{|\text{RecSet}(v)|},$$

where the definition of correct recommendations depends on a threshold ϵ:

$$\text{CorrectSet}(u) = \{(c_k, i_k) \in \text{RecSet}(u): \text{Correct}(i_k, u, c_k; 1)\}$$

$$\text{Correct}(i, u, v; \lambda) = \delta(|r(u, i) - r(v, i)| \leq \epsilon; \lambda),$$

$\delta(a; b)$ being a binary function like before whose output is a value b if the predicate a is true, and 0 otherwise. That is, the recommendations considered as correct are those in which the user was involved as a neighbour, and her ratings were close (up to a distance of ϵ) to the actual ratings.

A similar trust metric, called **expertise trust**, is presented in (Kwon et al., 2009), where the concept of 'correct recommendation' is also used. In that work Kwon and colleagues introduce a compensation value for situations in which few raters are available. Specifically, the correct recommendation function only outputs a value of 1 when there are enough raters for a particular item (more than 10 in the paper). Otherwise, an attenuation factor is introduced by dividing the number of raters by 10, in the same way as significance weighting is introduced in Pearson's correlation in (Herlocker et al., 2002). More formally, the predictor is defined as:

$$\gamma_2(u, v, i) = \gamma(v) = \frac{1}{\sum_{j \in I_v} \sum_{w \in \mathcal{U}_i} 1} \sum_{j \in \mathcal{I}_v} \sum_{w \in \mathcal{U}_i} \text{Correct}\big(j, v, w; \lambda(j)\big)$$

where $\lambda(j)$ is 1 when item j has more than 10 raters, and $\mathcal{U}_i$ denotes the users who rated item i. In the same paper the authors propose another trust metric called **trustworthiness**, which is equivalent to the absolute value of the similarity between the target user's ratings and the average ratings given by the community (denoted as $\bar{R}$). The authors introduce the significance weighting factor β as in (Herlocker et al., 2002), in a way that $\beta(v)$ is 1 when user v has more than 50 ratings; otherwise, β is computed as the user's ratings divided by 50. Once the β factor is computed, the predictor is defined as follows:

$$\gamma_3(u,v,i) = \gamma(v) = \beta(v) \times \left| \frac{\sum_{j \in \mathcal{I}_v} \big(r(v,j) - \bar{r}(v)\big)(\bar{r}(j) - \bar{R})}{\sqrt{\sum_{j \in \mathcal{I}_v} \big(r(v,j) - \bar{r}(v)\big)^2 \sum_{j \in \mathcal{I}_v} (\bar{r}(j) - \bar{R})^2}} \right|$$

Hwang and Chen (2007) present a global trust metric, which we call **global trust deviation**, defined as an average of local (user-to-user) trust deviations. This metric makes use of the predicted rating for a user–item pair by using only one user as neighbour:

$$\tilde{r}(u,i) \sim \tilde{r}(u,i;v) = \bar{r}(u) + \big(r(v,i) - \bar{r}(v)\big)$$

where user v is the considered neighbour. The predictor is then computed by averaging the prediction error of co-rated items between each user, and normalising the error according to the rating range R_r (e.g. in a typical 1 to 5 rating scale, $R_r = 4$):

$$\gamma_4(u,v,i) = \gamma(v) = \frac{1}{|N_k(v)|} \sum_{w \in N(v)} \left(\frac{1}{|\mathcal{I}_v \cap \mathcal{I}_w|} \sum_{j \in \mathcal{I}_v \cap \mathcal{I}_w} \left[1 - \frac{|\tilde{r}(v,j;w) - r(v,j)|}{R_r} \right] \right).$$

Finally, a performance predictor inspired by the clarity score defined for query performance (Cronen-Townsend et al., 2002) was proposed in (Bellogín and Castells, 2010), considering its adaptation to predict neighbour performance in collaborative filtering. In the same way query clarity captures the lack of ambiguity in a query, **user clarity** is expected to capture the lack of ambiguity in a user's preferences. Thus, the amount of uncertainty involved in a user's profile is assumed to be a good predictor of her performance; and the larger the following value, the lower the uncertainty and the higher the expected performance:

$$\gamma_5(u,v,i) = \gamma(v) = KLD(v \,\|\mathcal{U} \setminus \{u\}) = \sum_{w \in \mathcal{U} \setminus \{v\}} p(w|v)\, log_2 \frac{p(w|v)}{p(w)}$$

The probabilistic models defined in that work are based on smoothing estimations and conditional probabilities over users and items. Specifically, a uniform distribution is assumed for users and items, whereas the user-user probability is defined by an expansion through items as follows:

$$p(v|u) = \sum_{i \in \mathcal{I}_u} p(v|i)p(i|u).$$

Conditional probabilities are linearly smoothed with the user's probabilities and the maximum likelihood estimators, which finally depend on the rating given by the user towards an item; i.e., $p_{ml}(i|u) \propto r(u,i)$.

It is interesting to note that this predictor (and the probability model in which is grounded) does not correspond with any of the adaptations of the clarity score proposed in Chapter 6, since relations between users are not considered in any of the rating-based probability models presented.

In addition to the integration of the above methods in the role of neighbour effectiveness predictors in our framework, we propose two novel predictors based on well known quantities measured over the probability models of (Bellogín and Castells, 2010): the entropy and the mutual information. Entropy, as an information-theoretic magnitude, measures the uncertainty associated with a probability distribution (Cover and Thomas, 1991). Borrowing the definition of user entropy from Chapter 6, we hypothesise that the uncertainty in the system's knowledge about a user's preferences may be a relevant signal in the effectiveness of a user as a potential neighbour, which could be captured by the entropy of the item distribution as follows:

$$\gamma_7(u,v,i) = \gamma(v) = -H(\mathcal{I}_v) = \sum_{j \in \mathcal{I}_v} p(j|v)\, log_2\, p(j|v).$$

Note that uncertainty, measured in this way, can be due to the system's knowledge about the user's tastes, or may come from the user herself (e.g. some users may have strong preferences, while others may be more undecided), and both causes may similarly affect the neighbour effectiveness. In either case the predictor can be interpreted as the lack of ambiguity in a user profile.

The second information-theoretic magnitude we propose to use over the probability models presented above is the mutual information. To be precise, the mutual information is a quantity computed between two random variables that measure the mutual dependence of the variables, or, in other terms, the reduction in uncertainty about one variable provided some knowledge about the other (Cover and Thomas, 1991). Here, we propose to adapt this concept, and compute the **mutual information** between the neighbour and the rest of the community in order to assess the uncertainty involved in the neighbour's preferences. For this purpose, instead of computing the mutual information over all the events in the sample space for both variables (users), we fix one of them (for the current neighbour), and move along the other dimension:

$$\gamma_6(u,v,i) = \gamma(v) = MI(v; \mathcal{U} \setminus \{u\}) = \sum_{w \in \mathcal{U} \setminus \{u\}} p(w|v)\, log_2 \frac{p(w|v)}{p(v)p(w)}.$$

User-Item Predictors

User-item predictors consist of performance predictors that depend on a user-item pair. More specifically, they are defined upon the active neighbour and the target item. This type of predictor is more difficult to apply because of its higher vulnerability to data sparsity. In a bi-dimensional user-item input space less observations can be associated to each input data point, whereby the confidence on the predictor outcome is lower, as it can be biased to outliers or unusual users or items.

A local trust metric based on the target user and item is proposed in (O'Donovan and Smyth, 2005). This metric is called **item-level trust**, and aims to discriminate reliable neighbours depending on the current item, since the same user may be more trustworthy for predicting ratings for certain items than for others. The formulation of this predictor can be seen as a particularisation of γ_1, but constraining the recommendation set only to the pairs in which the current item is involved:

$$\gamma_8(u,v,i) = \gamma(v,i) = \frac{|\{(c_k,i_k) \in \text{CorrectSet}(v): i_k = i\}|}{|\{(c_k,i_k) \in \text{RecSet}(v): i_k = i\}|}.$$

User-User Predictors

The user-user predictors take as inputs two users: the active user and the current neighbour. User-user predictors based on local trust metrics have been studied further than user-item predictors in the literature, since the former are able to represent how much a user can be trusted by another, and let for different interpretations of the relation between users. These metrics have been often researched in the scope of social networks, and the users' explicit links in this context (Ziegler and Lausen, 2004; Massa and Avesani, 2007a), along with several trust metrics based on ratings, as we shall show below. In this way, although social-based metrics could be smoothly integrated in our framework, here we focus on a complementary view on trust where predictors are defined based on ratings. We leave other type of predictors as future work.

A first simple neighbour reliability criterion one may consider is the amount of common experience with the target user, that is, the amount of information upon which the two users can be compared. If we define "user experience" as the set of items the user has interacted with, we may define a predictor embodying this principle as:

$$\gamma_9(u,v,i) = \gamma(u,v) = |\mathcal{I}_u \cap \mathcal{I}_v|.$$

We shall refer to this predictor as **user overlap**. This predictor will serve as a basis for subsequent predictors, since most of them will depend on the items rated by both users. For instance, it has a clear use in assessing the reliability of the inter-user similarity assessments, which has been applied in the literature under a more practi-

cal, ad-hoc manner. Specifically, Herlocker et al. (2002) proposed the introduction of a weight on the similarity function, where the latter is devalued when it has been based on a small number of co-rated items. We may formulate **Herlocker's significance weighting** predictor as follows:

$$\gamma_{10}(u,v,i) = \gamma(u,v) = \frac{|\mathcal{I}_u \cap \mathcal{I}_v|}{n_H} \text{ if } |\mathcal{I}_u \cap \mathcal{I}_v| < n_H; 1 \text{ otherwise,}$$

where n_H is the minimum number of co-rated items that two users should have in common in order to avoid similarity penalisation. A value of $n_H = 50$ was proved empirically to work effectively.

A variation of the previous scheme was proposed in (McLaughlin and Herlocker, 2004), to which we shall refer as **McLaughlin's significance weighting**:

$$\gamma_{11}(u,v,i) = \gamma(u,v) = \frac{\max(|\mathcal{I}_u \cap \mathcal{I}_v|, n_{Mc})}{n_{Mc}}.$$

This predictor is aimed to be equivalent to the Herlocker's significance weighting (γ_{10}) formulation when $n_{Mc} = n_H$. However, we note that γ_{10} and γ_{11} represent different concepts, and are not fully equivalent. For instance, as noted in (Ma et al., 2007), γ_{11} may return values larger than 1 when $|\mathcal{I}_u \cap \mathcal{I}_v| > n_{Mc}$, while γ_{10}, by definition, always returns a value in the $(0,1]$ interval.

Alternatively, the following variant can be drawn from (Ma et al., 2007), which is just a more compact reformulation of γ_{10}:

$$\gamma_{12}(u,v,i) = \gamma(u,v) = \frac{\min(|\mathcal{I}_u \cap \mathcal{I}_v|, n_M)}{n_M}.$$

A more elaborated predictor was proposed in (Weng et al., 2006). The rationale behind such predictor is to consider two situations depending whether or not user u takes into account the recommendation made by neighbour v. In this sense trustworthiness is defined as the reduction in the proportion of incorrect predictions of going from the latter situation to the former. The definition of this predictor, denoted as **user's trustworthiness**, is the following:

$$\gamma_{13}(u,v,i) = \gamma(u,v) = \frac{1}{|R|^2 - \sum_x n(u,v;x,\cdot)^2}\left[|R| \sum_x \sum_y \frac{n(u,v;x,y)^2}{n(u,v;\cdot,y)} - \sum_x n(u,v;x,\cdot)^2\right]$$

In this formulation $|R|$ represents the number of allowed rating values in the system (e.g. in a 1 to 5 rating scale, $|R| = 5$), the function $n(u,v;x,y)$ represents the number of co-rated items on which v's ratings have the value y while u's ratings are x, that is, $n(u,v;x,y) = |\{(u,\cdot,x)\} \cap \{(v,\cdot,y)\}|$ when each rating tuple is represented as (a,b,c), given a user a, an item b, and a rating value c. In the same way, $n(u,v;x,\cdot) = \sum_y n(u,v;x,y)$ represents all the co-rated items between u and v

rated with any rating value by user v, and, analogously, $n(u,v;\cdot,y)=\sum_x n(u,v;x,y)$. In this case, the assumed hypothesis is that trust is one's expectation of other's competence in reducing its uncertainty in predicting new ratings.

Finally, a user-user predictor can be defined based on the global trust deviation predictor defined above (γ_4). In fact, Hwang and Chen (2007) define **trust deviation** by ignoring the average along users as follows:

$$\gamma_{14}(u,v,i)=\gamma(u,v)=\frac{1}{|\mathcal{I}_u\cap\mathcal{I}_v|}\sum_{j\in\mathcal{I}_u\cap\mathcal{I}_v}\left[1-\frac{|\tilde{r}(u,j;v)-r(u,j)|}{R_r}\right]$$

This predictor identifies effective neighbours mainly based on how many trustworthy (understood as "accurate") recommendations a user has received from another.

8.4 Experimental results

In this section we report experiments in which the proposed neighbour effectiveness prediction framework is tested. First, we check the existing correlations between the user-based predictors defined in Section 8.3.2 and the neighbour performance metrics proposed in Section 8.3.1, as a direct test of their predictive power. For the user-item predictors we cannot analyse their correlation because we have no neighbour performance metric depending on both the target user and an item available.

Moreover, we test the usefulness of the predictors to enhance the final performance of memory-based algorithms, by using the predictors' values in the selection and weighting of neighbours, that is, by taking the predictors as the scoring function in Equation (8.2).

Our experiments were conducted on two versions of the MovieLens dataset, namely the 100K and 1M versions, described in Section 3.4.1 and Appendix A.1. For the user-based collaborative filtering method, we used Pearson's correlation as the similarity measure between users, and a varying neighbourhood size (k), which is a parameter with respect to which the results were examined.

8.4.1 Correlation analysis

We analyse the correlation between neighbour quality metrics and neighbour performance predictors in terms of the Pearson and Spearman's correlation metrics. Correlation provides a measure of the predictive power of the neighbour effectiveness prediction approaches: the higher the (absolute) correlation value, the better the predictor estimates the positive neighbour effect on the recommendation accuracy. The sign of the correlation coefficient represents whether the two involved variables – neighbour quality metric and neighbour performance predictor – are directly or inversely correlated.

	Absolute error deviation μ_1 (-)	Neighbour goodness μ_3 (+)	Sign of error μ_2 (+)
Clarity	-0.21	+0.17	+0.14
Entropy	-0.18	+0.18	+0.12
Expertise	-0.62	+0.03	+0.25
Global Trust Deviation	-0.35	-0.01	+0.08
Mutual Information	-0.20	+0.17	+0.12
Profile Level Trust	+0.62	-0.04*	-0.24
Trustworthiness	-0.21	+0.03	+0.20

Table 8.1. Pearson's correlation between the proposed neighbour quality metrics and neighbour performance predictors in the MovieLens 100K dataset. Next to the metric name, an indication about the sign of the metric – direct(+) or inverse(-) – is included. Not significant values for a p-value of 0.05 are denoted with an asterisk (*).

	Absolute error deviation μ_1 (-)	Neighbour goodness μ_3 (+)	Sign of error μ_2 (+)
Clarity	-0.30	+0.16	+0.21
Entropy	-0.22	+0.17	+0.15
Expertise	-0.65	+0.02	+0.30
Global trust deviation	-0.38	-0.03	+0.11
Mutual Information	-0.25	+0.16	+0.17
Profile Level Trust	+0.65	-0.02	-0.30
Trustworthiness	-0.24	+0.03	+0.25

Table 8.2. Spearman's correlation between quality metrics and performance predictors in the MovieLens 100K dataset.

Table 8.1 and Table 8.2 show the correlation values obtained on the MovieLens 100K dataset for the user-based predictors. We associate a sign to each quality metric indicating whether the metric is direct (denoted as '+') or inverse (denoted with '-'), according to the expected sign of the correlation with the predictor, i.e., a metric is direct if the higher its value, the better the true neighbour performance. We can observe that the Spearman's correlation values are consistent, but slightly higher than Pearson's, thus evidencing a non-linear relationship between the quality metrics and the performance predictors.

The absolute error deviation (μ_1) metric presents higher values when the neighbour's prediction is less accurate, being thus an inverse neighbour metric. The other two metrics, sign of error (μ_2) and neighbour goodness (μ_3), are, by definition, direct neighbour metrics, since the former indicates how many times a recommendation from the neighbour has been made in the right direction, whereas the latter represents the change in error between excluding a particular user in the neighbourhood or including her, and thus, the larger this error, the "better" neighbour this user.

	Absolute error deviation μ_1 (-)	Neighbour goodness μ_3 (+)	Sign of error μ_2 (+)
Clarity	-0.14	+0.40	+0.02
Entropy	-0.07	+0.39	-0.08
Expertise	-0.95	-0.06	+0.70
Global Trust Deviation	-0.55	-0.24	+0.36
Mutual Information	-0.17	+0.30	+0.13
Profile Level Trust	+0.83	+0.04	-0.55
Trustworthiness	-0.27	+0.03	+0.36

Table 8.3. Pearson's correlation between quality metrics and performance predictors in the MovieLens 1M dataset. All the values are significant for a p-value of 0.05.

	Absolute error deviation μ_1 (-)	Neighbour goodness μ_3 (+)	Sign of error μ_2 (+)
Clarity	-0.16	+0.35	+0.04
Entropy	-0.03	+0.37	-0.10
Expertise	-0.94	-0.09	+0.69
Global trust deviation	-0.54	-0.25	+0.39
Mutual information	-0.16	+0.31	+0.04
Profile level trust	+0.94	+0.09	-0.69
Trustworthiness	-0.25	+0.02	+0.37

Table 8.4. Spearman's correlation between quality metrics and predictors in the MovieLens 1M dataset.

We can observe in Table 8.1 that, except for some of the predictors that obtain very low absolute values (< 0.10), the four quality metrics are consistent with each other. This consistency is evidenced by the way the predictors correlate with the different metrics: some of the predictors obtain the correct correlations in every situation, that is, positive correlation with direct metrics and negative correlation with the inverse metric (like the clarity predictor), while other predictors obtain opposite values for all the metrics, that is, positive correlations with the inverse metric and negative correlations with direct metrics (such as the profile level trust predictor).

Also in Table 8.1 and Table 8.2 we see that each metric captures a different notion of neighbour quality because they show different correlation values with respect to the predictors. In this way, although consistent correlation results are obtained for direct and inverse metrics, each of them is actually detecting a different nuance of how a neighbour should behave in order to perform well.

Table 8.3 and Table 8.4 show the correlation values obtained on the Movie-Lens 1M dataset. We can observe that the trend in correlation is very similar to the behavior observed on the 100K dataset, and thus, similar conclusions can be drawn from it. There are, however, some changes in the absolute values of the correlation scores for some combinations of performance predictor and quality metric. For instance,

the clarity predictor and the neighbour goodness metric obtain larger values in this dataset, while the correlation between entropy and absolute error deviation is smaller.

It is important to note that the number of points used to compute the correlation values is different in the two datasets; there are less than 1,000 points in MovieLens 100K (with 943 users), and more than 6,000 points in MovieLens 1M dataset. This difference affects the significance of the correlation results, as already described in Section 5.4.2, where we observed how the confidence test for a Pearson's (and Spearman's) correlation depends on the size of the sample, and thus, the significance of a correlation value may change for different sample sizes.

In our experiments, for MovieLens 100K, the correlations are significant for a p-value of 0.05 when $r > 0.05$, and in the 1M dataset when $r > 0.02$. Hence, in Table 8.1, there is only one non-significant correlation value (denoted with an asterisk), whereas in Table 8.3, all the results are statistically significant.

Analysing in more detail the reported results for both datasets, we observe that the profile level trust predictor consistently obtains direct correlation values with inverse metrics, whereas inverse correlation values are obtained with direct metrics. This predictor seems to give higher scores to neighbours with larger deviations in their accuracy error, which would result on bad performance prediction because these values are not in the same direction than the performance metrics. The expertise and global trust deviation predictor obtain strong inverse correlations with the absolute error deviation metric, although their correlations with respect to the neighbour goodness metric are negligible, especially for the first predictor, in both datasets. At the other end of the spectrum, the clarity, entropy, and mutual information predictors obtain strong correlation values with the neighbour goodness, and moderate correlations with the rest of metrics, which make these predictors good candidates for successful neighbour performance predictors. Finally, the trustworthiness predictor obtains a significant amount of correlation with respect to the absolute error deviation and sign of error metrics, although its correlation with respect to the neighbour goodness is very low. This predictor thus seems to be useful on estimating how accurate the neighbour may be in terms of the error in a user basis, but probably not as a global metric.

Table 8.5 shows the correlations obtained for user-user neighbour predictors and the proposed user-neighbour clarity metric. Due to the high dimensionality of the vectors involved in this computation, we have considered only those users that have at least one item in common. Despite this fact, correlations are almost negligible, except for the McLaughlin's significance weighting predictor and the Spearman's coefficient, which evidences a non-linear relation between this predictor and the metric. In the next section we shall show that this function is one of the best performing predictors among the evaluated neighbour scoring functions. This result confirms the usefulness of the proposed neighbour performance metric since it is able to discrimi-

	Movielens 100K		Movielens 1M	
	Pearson	Spearman	Pearson	Spearman
Herlocker	0.02	0.03	0.01	0.02
McLaughlin	0.01	0.12	0.01	0.11
Trust Deviation	0.01	0.01	0.01	0.01
User Overlap	0.02	0.03	0.02	0.02
User's Trustworthiness	-0.02	-0.02	-0.01	-0.01

Table 8.5. Correlation between the user-neighbour goodness and user-user predictors in the two datasets evaluated.

nate which neighbour performance predictors are able to capture interesting properties between the user and her neighbours.

In summary, we have observed that most of the performance predictors agree with respect to the different performance metrics, and in general, the correlations computed between neighbour quality metrics and neighbour performance predictors are statistically significant.

8.4.2 Performance analysis

The results reported in the previous section show that some of the studied predictors have the ability to capture neighbour performance, and because of that we hypothesise that they could be used to improve the accuracy of a recommendation model. This hypothesis, nonetheless, has to be checked since the metric against which we measure the neighbour goodness is not the same as the final recommendation performance metric we aim to optimise. With the experiments we report next we aim to confirm the usefulness of the proposed predictors, the validity of the proposed metrics as useful references to assess the power of the predictive methods, and the usefulness of the overall framework as a unified approach to enhance neighbourhood-based collaborative filtering.

In order to achieve this we test the integration of the neighbour predictors into a neighbour selection and weighting scheme for user-based collaborative filtering, as described in Section 8.2.1. Besides testing the effectiveness of the predictors, this experiment provides for observing to what extent the correlations obtained in the previous section correspond with improvements in the final performance of those predictors.

We provide recommendation accuracy and precision results on the MovieLens 1M dataset. Those obtained on the MovieLens 100K dataset are not reported here since they had similar trends. Figure 8.1 and Figure 8.2 show the Root Mean Square Error (RMSE) of the Resnick's collaborative filtering adaptation proposed in Equation (8.2) when used for different neighbour selection and weighting approaches. The curves at the top of the figures represent the values obtained when neighbour per-

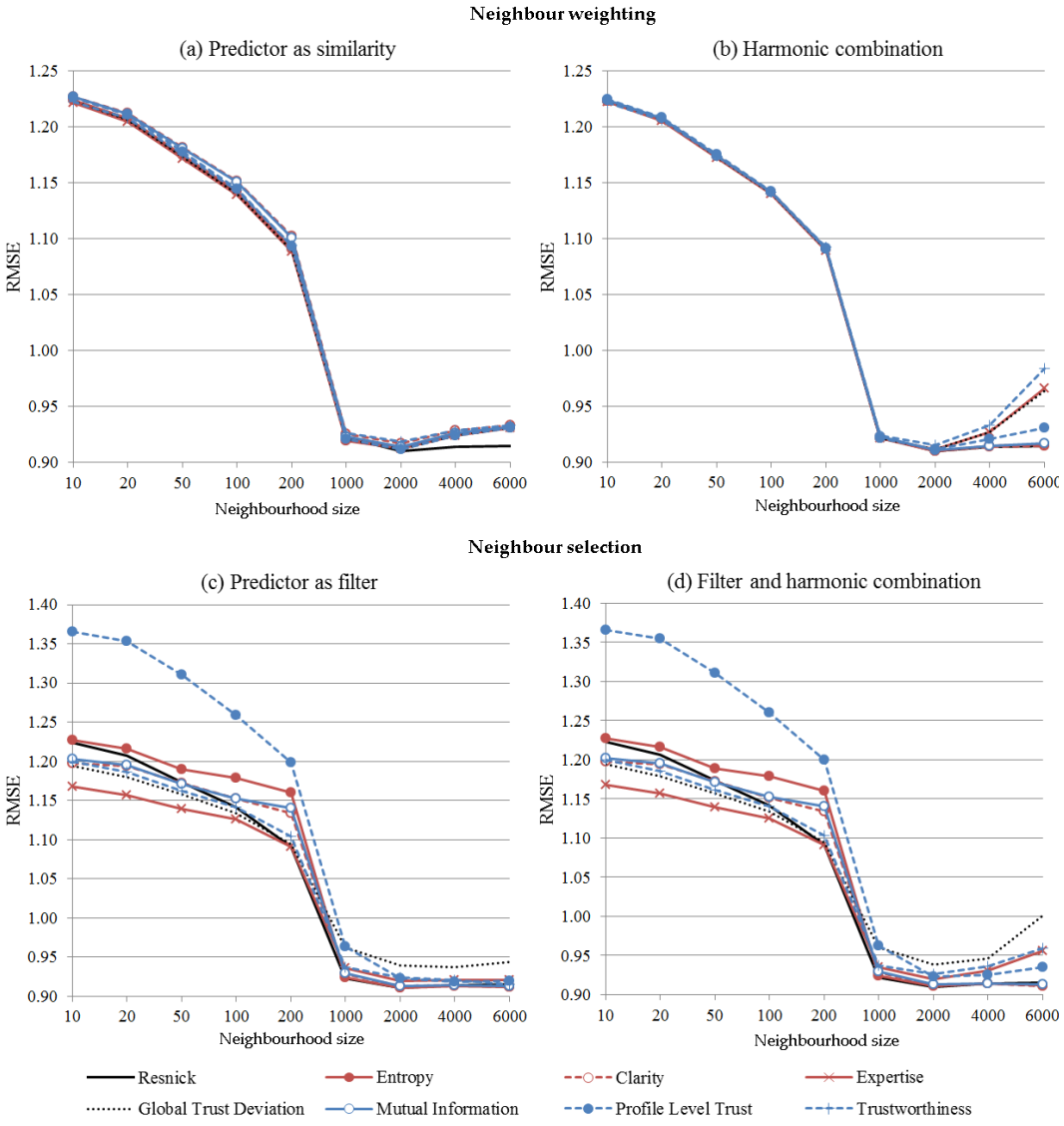


Figure 8.1. Performance comparison for user-based predictors and different neighbourhood sizes.

formance predictors are used for neighbour weighting, that is, when the standard neighbour selection strategy is used ($f^{neigh} = f_0^{neigh}$ in Equation (8.2)). Note that since the lines represent errors, the lower these values, the better the performance. Besides, Figure 8.3 presents the results found with the precision at 10 (P@10) ranking metric of a subset of the proposed methods, where in this case the higher the values, the better the performance.

A different aggregation function is used in each approach, depending on whether the harmonic mean between the predictor score and the similarity value (function $f^{agg} = f_2^{agg}$, on the right), or the projection function ($f^{agg} = f_4^{agg}$, on the left) are used, in the latter case in order to ignore the similarity. The curves at the bottom

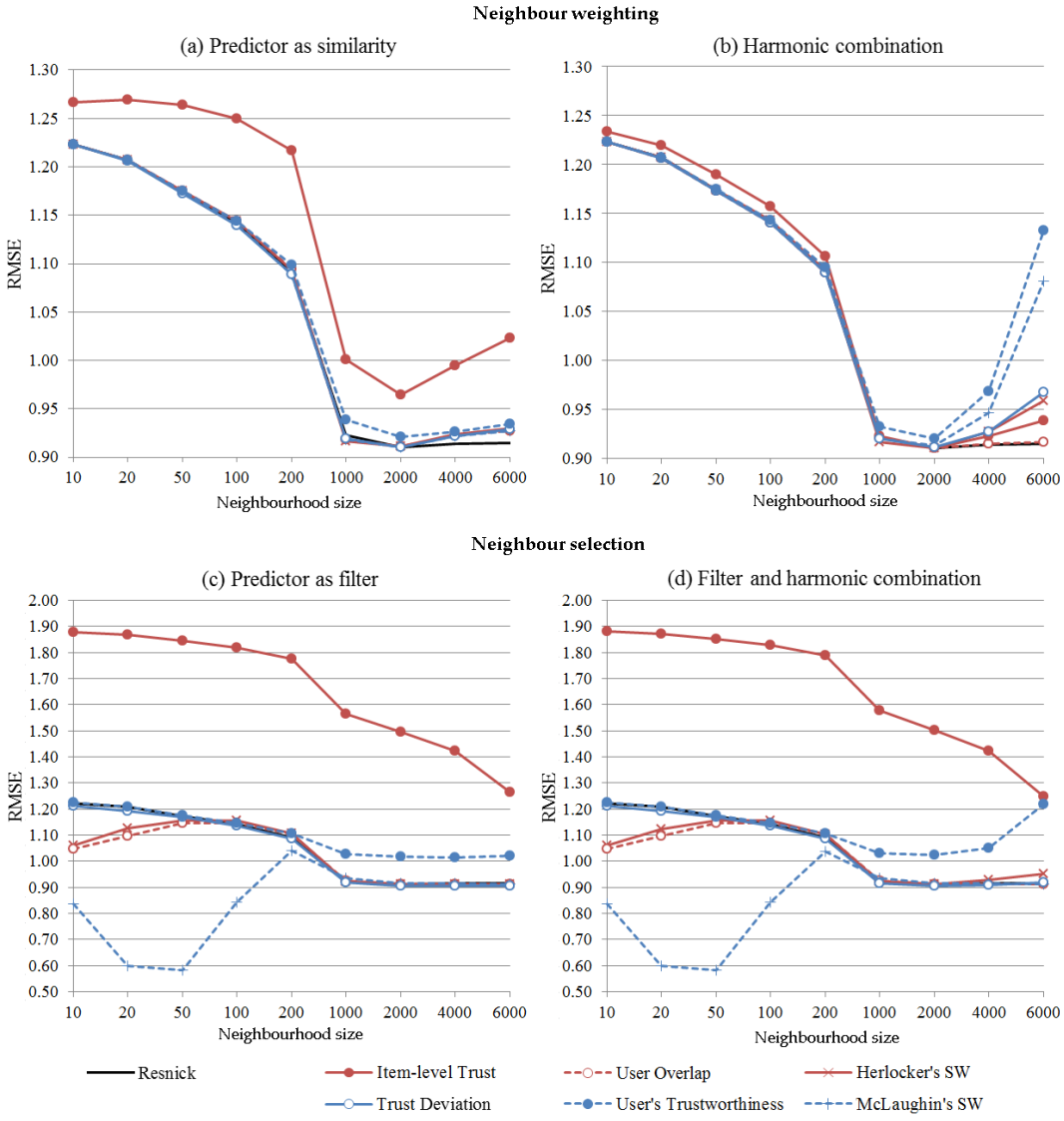


Figure 8.2. Performance comparison using user-item and user-user predictors for different neighbourhood sizes.

of the figures show the neighbour selection approach ($f^{neigh} = f_1^{neigh}$ in Equation (8.2)) along with the same neighbour weighting functions described above (i.e., f_2^{agg} on the right and f_4^{agg} on the left). The rest of the aggregation functions, such as average (f_1^{agg}) and product (f_3^{agg}), were also evaluated for neighbour selection and weighting, but provided results equivalent to those of the harmonic mean. For this reason, they have been omitted in the figures to avoid cluttering them. We believe this equivalence may be due to the normalisation factor included in the collaborative filtering formulation, since it would cancel out the weights obtained by the harmonic, average, and product functions in the same way.

	RMSE
Resnick	1.174
Clarity	1.181
Entropy	1.175
Expertise	1.171
Global Trust Deviation	1.173
Mutual Information	1.180
Profile Level Trust	1.177
Trustworthiness	1.175

	RMSE
Resnick	1.174
Herlocker	1.175
Item-level Trust	1.264
McLaughlin	1.174
Trust Deviation	1.173
User Overlap	1.175
User's Trustworthiness	1.175

Table 8.6. Detail of the accuracy of baseline vs. recommendation using neighbour weighting; here, performance predictors are used as similarity scores (50 neighbours).

	RMSE
Resnick	1.174
Clarity	1.172
Entropy	1.189
Expertise	1.139
Global Trust Deviation	1.158
Mutual Information	1.171
Profile Level Trust	1.310
Trustworthiness	1.162

	RMSE
Resnick	1.174
Herlocker	1.156
Item-level Trust	1.843
McLaughlin	0.581
Trust Deviation	1.168
User Overlap	1.146
User's Trustworthiness	1.174

Table 8.7. Detail of the accuracy of baseline vs recommendation using neighbour selection; here, performance predictors are used for filtering (50 neighbours).

Figure 8.1 shows the accuracy results when only user-based neighbour predictors are evaluated. We observe that, independently from the neighbourhood size, using performance predictors as similarity scores does not lead to large differences with respect to the baseline. These results are compatible with those presented in (Weng et al., 2006), where the improvement in RMSE is not very high (ΔMAE < 0.05 in that work). For the sake of clarity, in Table 8.6 and Table 8.7 we show the error values for a horizontal cut of the left curves; specifically, when the neighbourhood size is 50. We can observe that some predictors do improve Resnick's accuracy. Regarding the use of the harmonic mean as aggregation function (curves on the right), similar results are obtained except for very large neighbourhood sizes, for which some of the performance predictors produce worse results than the baseline, probably due to the amount of noise created by considering too many neighbours.

The curves at the bottom of the figures represent the accuracy results for neighbour selection strategies. In this case some of the predictors lead to worse performance than the baseline, particularly the profile level trust (γ_1). This situation is consistent with the correlations observed in the previous section, since this predictor obtained inverse correlations with the different metrics, i.e., direct correlation values

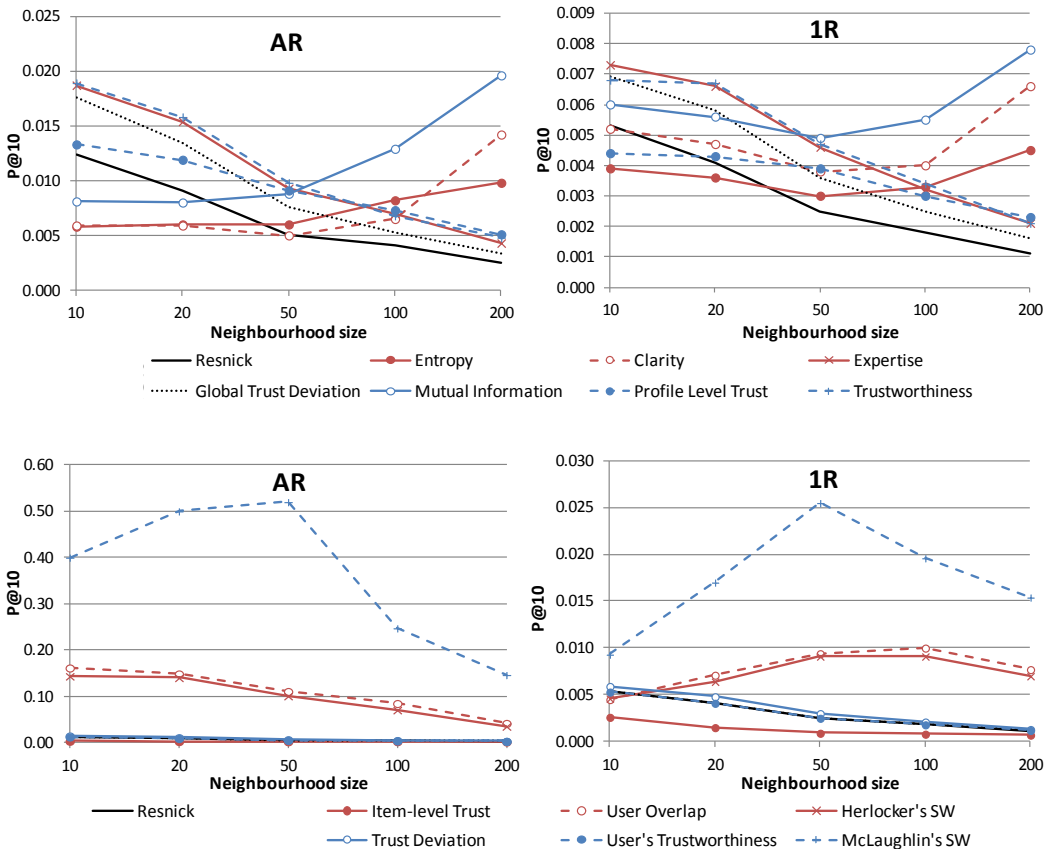


Figure 8.3. Performance comparison using ranking-based metrics for both user and user-user neighbour predictors using the AR and 1R evaluation methodologies.

with inverse metrics, and inverse values with direct metrics. Moreover, as predicted by the correlation analysis, trustworthiness (γ_3), mutual information (γ_6), and clarity (γ_5) result in some of the best performing recommenders (with strong correlations), as shown in the figures and in Table 8.7, along with expertise (γ_2) and global trust deviation (γ_4), which obtained more moderated correlation values.

In Figure 8.2 we can see how user-item and user-user neighbour predictors affect the performance of collaborative filtering recommenders. The curves in the top show that most of the predictors obtain a similar performance to that of the baseline, except for the item-level trust (γ_8), the performance of which is much worse than Resnick's. Table 8.6 shows the specific error values for these recommenders. It is interesting to note that the performance of this predictor is drastically improved when using the harmonic mean as the aggregation function (shown on the right side of the figure). Similarly to user-based neighbour predictors (Figure 8.1), some of the user-item and user-user predictors decrease their accuracy with large neighbourhoods; in this case, user's trustworthiness (γ_{13}) and McLaughlin's significance weighting (γ_{12}) are the more representative examples.

A different conclusion results when neighbour selection is analysed (curves at the bottom). Two of the predictors are characterised by a much better (McLaughlin's significance weighting, γ_{12}) or worse (item-level trust, γ_8) final performance, independently from the weighting aggregation function. Table 8.7 shows the specific error values obtained for each of these predictors. It is interesting how the McLaughlin's predictor, despite its inability to boost good neighbours (see top figures), seems to be very useful for neighbour selection. This effect, nonetheless, is attenuated when the neighbourhood increases, since in that situation, selection methods have to deal with too many users in each neighbourhood. We believe the reason why this predictor is very good for neighbour selection is because it gives higher scores to those neighbours that have more items in common with the target user, and thus the confidence in the computation of the similarity values between the neighbour and the target user is higher. It is worth noting that, to the best of our knowledge, this function has never been used for neighbour selection, since its original motivation was to penalise the similarity value whenever it has been based on a small number of co-rated items. However, by plugging this function into our framework, and measuring its predictive power for user-neighbour performance, a novel application naturally emerges and provides very good results.

Finally, in Figure 8.3 we can observe that a similar trend is found with P@10 for both user-based predictors (top curves), and user-item and user-user predictors (bottom curves). In the figure we only present the results of the neighbour selection and weighting approaches for less than 200 neighbours, since the results of the rest of the approaches and neighbourhoods are very similar. It is worth noting that the two methodologies evaluated – AR and 1R – agree on the order of the best and worst performing dynamic approaches, although as already observed in the previous chapter, the absolute performance values obtained with each methodology may be very different – e.g. the maximum P@10 value with 1R is 0.1, which is reached by several recommendation methods with the AR methodology. More interestingly, these results show consistency between the performance of some dynamic approaches using error- and ranking-based metrics, since the best and worst predictors according to RMSE and P@10 are the same; McLaughlin's significance weighting and item-level trust, respectively. Moreover, the entropy and clarity user-based predictors show worse performance in small neighbourhoods, but outperform the baseline significantly in larger neighbourhoods, something different to what we observed in the previous experiment with error-based metrics.

In summary, we have been able to validate both the proposed user-user neighbour performance metrics, and the different evaluated user-user neighbour performance predictors. We have obtained positive results when this type of predictors has been introduced and compared against the baseline in the different aggregation strategies and configurations, and these results are consistent with the correlations

obtained between the predictors and the performance metrics. In particular, McLaughlin's significance weighting obtains an improvement up to 55% in both accuracy (i.e., error decrease) and precision (i.e., precision improvement) when this predictor is used to select the neighbours which will further contribute to the rating prediction. Besides, the (Spearman's) correlation for this predictor is positive and strong, in contrast to the values obtained for the rest of user-user predictors, which did not improve the accuracy of the baseline. In this context, a possible drawback of the conducted analysis is that we have not been able to define neighbour performance metrics based on user-item pairs, and thus the user-item neighbour performance predictors are out of the scope of the developed correlation analysis. Nevertheless, the obtained results showed that the only user-item neighbour performance predictor defined here – the item-level trust – is not able to outperform the baseline recommender. We believe this fact, which is in contradiction with what was reported in (O'Donovan and Smyth, 2005), may be caused by the different variables taking place in our evaluation, such as the dataset (MovieLens 1M instead of MovieLens 100K), the neighbourhood size (not specified in the original paper), and the several aggregation functions and combinations used across our experiments.

8.4.3 Discussion

The reported experiment results provide empiric evidence of the usefulness of the proposed framework, and the specific proposed predictors, as an effective approach to enhance the accuracy of memory-based collaborative filtering. As described in the preceding sections, the methodology comprises two steps, one in which the predictive power of neighbour predictors is assessed, and one in which the predictors are introduced in the collaborative filtering scheme to enhance the effectiveness of the latter. Our experiments confirm a strong correlation for some of the predictors – both user predictors and user-user predictors –, and this has been found to correspond with final accuracy enhancements in the recommendation strategy: the predictors that obtain strong direct correlations with the performance metrics are the best performing dynamic strategies; the profile level trust predictor, which obtains inverse correlation values with respect to the neighbour performance metrics, is the worst performing dynamic strategy.

In light of these results, it could be further investigated whether the actual correlation values between neighbour performance predictors and neighbour performance metrics could be used to infer how each predictor should be incorporated into a memory-based collaborative filtering method as a neighbour scoring function, since there is no obvious link between the ranking of the best performing scoring functions and the strength of their corresponding correlations. As a starting point, only the sign of the correlation could be considered, using either the raw neighbour predictor score (for positive correlations) or its inverse (for negative values). Then, this

rationale could be further elaborated and evaluated in order to check whether the performance improvements are consistent.

Research on finding functions with strong correlation power with respect to neighbour performance metrics could be an interesting area by itself, since it could have different final applications. We have experimented here with variations in neighbour selection and weighting for user-based collaborative filtering, but those predictors (functions) could also be used, for instance, for active learning (Elahi, 2011), or for providing more meaningful explanations (Marx et al., 2010), depending or based on the predicted performance of a particular user's neighbours.

8.5 Conclusions

We have shown in this chapter that performance prediction does not only serve to aggregate entire recommender systems, but also to aggregate subcomponents of recommender algorithms – in this case, neighbour related terms in collaborative filtering. We propose a theoretical framework for neighbour selection and weighting in user-based recommender systems, which is based on a performance prediction approach drawn from the query performance methodology of the Information Retrieval field. By viewing the neighbourhood-based collaborative filtering rating prediction task as a case of dynamic output aggregation, our approach places user-based collaborative filtering in a more general frame, linking to the principles underlying the formation of ensemble recommenders, and rank aggregation in Information Retrieval. By doing so, it is possible to draw concepts and techniques from these areas, and vice versa. Our study thus provides a comparison of different state-of-the-art rating-based trust metrics and other neighbour scoring techniques, interpreted as neighbour performance predictors, and evaluated under this new angle. The framework lets an objective analysis of the predictive power of several neighbour scoring functions, integrating different notions of neighbour performance into a unified view. Thus, the proposed methodology discriminates which neighbour scoring functions are more effective in predicting the goodness of a neighbour, and thus identifies which weighting functions are more effective in a user-based collaborative filtering algorithm.

Drawing from different state-of-the-art neighbour scoring functions – cast as user, user-user, and user-item neighbour performance predictors –, we have reported several experiments in order to, first, check the predictive power of these functions, and second, validate them by comparing the final performance of neighbour-scoring powered memory-based strategies with that of the standard collaborative filtering algorithm. We also evaluate different ways to introduce these functions in the rating prediction formulation, namely for neighbour weighting, neighbour selection, and combinations thereof. In this context, methods where neighbour scoring functions

were integrated outperform the baseline for different values of neighbourhood size and predictor type.

We have also proposed several neighbour performance metrics that capture different notions of neighbour quality. The evaluated performance predictors show consistent correlations with respect to these metrics, and some of them present particularly strong correlations. Interestingly, a correspondence is confirmed between the correlation analysis and the final performance results, in the sense that the correlation values obtained between neighbour performance predictors and neighbour performance metrics anticipate which predictors will perform better when introduced in a memory-based collaborative filtering algorithm.

This research opens up the possibility to several research lines for the integration of other types of predictors and trust metrics into our framework. For instance, performance predictors defined upon social data, such as those defined in Chapter 6 based on user's trust network, could be smoothly integrated into our framework and analysed in the future. Furthermore, alternative neighbour performance metrics may be defined to check the predictive power of user-user and user-item predictors. These metrics may help better understand which characteristics of the neighbour performance such predictors are capturing, although based on a smaller amount of information since in rating-based systems users only rate items once. In particular, our framework would allow for different interpretations of the user's performance, by modelling different neighbour performance metrics, which may be oriented to accuracy (using error metrics as in this chapter), ranking precision, or even alternative metrics such as diversity, coverage and serendipity (Shani and Gunawardana, 2011). Additionally, other predictors based on item information could be defined similar to those proposed in (Weng et al., 2006; Ma et al., 2007), and easily incorporated into our framework using item-based algorithms instead of user-based.

Part V

Conclusions

Not everything that can be counted counts,
and not everything that counts can be counted.
Albert Einstein

Chapter 9

Conclusions and future work

In this thesis we have investigated how to measure and predict the performance of recommender systems. We have analysed and proposed an array of methods based on the adaptation of performance predictors from Information Retrieval – mainly the query clarity predictor, which captures the ambiguity of a query with respect to a given document collection. We have defined several language models according to various probability spaces to capture different aspects of the users and items involved in recommendation tasks. In this context, we have proposed and evaluated novel approaches drawing from Information Theory and Social Graph Theory for different recommender input spaces, using information-theoretic properties of the user's preferences and graph metrics such as PageRank over the user's social network.

Moreover, since we aimed to predict the performance of a particular recommender system, we required a clear recommender evaluation methodology against which performance predictions can be constrasted. Hence, in this thesis we addressed the evaluation methodology as part of the problem, where we have identified statistical biases in the recommendation evaluation – namely the sparsity and popularity biases – which may distort the performance assessments, and therefore may confound the apparent power of performance prediction methods. We have analysed in depth the effect of such biases, and have proposed two experimental designs that are able to neutralise the popularity bias: a percentile-based approach and a uniform-test approach. The systematic analysis of the evaluation methodologies and the new proposed variants have enabled a more complete and precise assessment of the effectiveness of our performance prediction methods.

On the other hand, we have exploited the proposed performance prediction methods in two applications where they are used to dynamically weight different components of a recommender system, namely the dynamic adjustment of weighted hybrid recommendations, and the dynamic weighting of neighbours' preferences in user-based collaborative filtering. Through a series of empirical experiments on several datasets and experimental designs, we have found a correspondence between the predictive power of our performance predictors and performance enhancements in the two tested applications.

In this chapter we present the main conclusions obtained in our research work. In Section 9.1 we provide a summary and a discussion of our contributions, and in Section 9.2 we provide research directions that could be addressed in future work.

9.1 Summary and discussion of contributions

In the next subsections we summarise and discuss the main contributions of this thesis, addressing the research goals stated in Chapter 1. These contributions are organised according to the three main objectives addressed. First, we analysed how to properly evaluate recommender systems in order to obtain unbiased measurements of a recommender system's performance. Second, we proposed performance predictors that aim to estimate the performance of a recommendation method. And third, we used our performance predictors to dynamically combine components of a recommender system.

9.1.1 Analysis of the definition and evaluation of performance in recommender systems

We have analysed different experimental designs existing in the literature about recommender systems, oriented in particular to ranking-based evaluation, and have shown that **assumptions and conditions underlying the Cranfield paradigm are not granted in usual recommendation settings**. Specifically, we have detected statistical confounders (biases) that arise in applying that paradigm to the evaluation of recommender systems. We have shown that the specific value of the evaluation metric has a use for comparative purposes, but has no particular absolute meaning by itself. We have shown that precision decreases linearly with the sparsity of relevant items (**sparsity bias**) in the AR evaluation methodology, whereas it does not suffer from such bias in the 1R approach.

We have also observed that a non-personalised recommender based on item popularity obtains high performance values, and have shown and analysed in detail how this is due to a **popularity bias** in the experimental methodology. To address these issues, we have proposed **novel experimental approaches that effectively neutralise the popularity bias**.

9.1.2 Definitions and adaptations of performance predictors for recommender systems

We have defined and elaborated **performance predictors in the context of recommendation**, usually taking the user as the object of the prediction, but also considering items as an alternative prediction input. Specifically, we have adapted the query performance predictor known as *query clarity* by taking different assumptions and formulations into several variations of **user clarity** predictors. We have also used information theoretical related concepts such as entropy, graph metrics like centrality, PageRank, and HITS, and other domain-specific, heuristic approaches. We have

defined these predictors upon three input spaces of user preferences: **ratings, logs, and social networks**. On ratings and logs we have defined several language models and vocabulary spaces in such a way that our adaptations of clarity would capture different aspects of the user in a unified formulation for both input spaces. Within the same framework, we have introduced the temporal dimension on log-based preference data, drawing and elaborating time-based performance predictors proposed in prior work in the IR field for ad-hoc search.

Additionally, we have defined **item-based predictors** when rating-based preferences are used, which aim to estimate the performance of the items under consideration (to be more precise, the performance of a recommender system in suggesting those items). Here, the main problem is how to define the true performance metric that the predictor is aimed to estimate, since the items are not the main input of the recommendation process. For this reason, we have developed novel methodologies where the performance of an item can be measured, also considering possible biases arising from heavy raters that may distort the results just for statistical reasons.

We have assessed the predictive accuracy of our methods by computing the correlation between estimated and true performance, following standard practice in the IR performance prediction literature. In doing so, we used the unbiased methodologies analysed throughout the thesis to **compare how the predictors behave when the sparsity and popularity biases have been neutralised**. We have found strong correlation values confirming that our approaches result in a **significant predictive power**.

9.1.3 Dynamic weighting in recommender ensembles

Prevalent in the Recommender Systems literature we find combination of recommenders into the so-called recommender ensembles, which are a special type of hybrid recommendation methods where several recommenders are combined, and which are currently very common in the field as represented by current competitions (Bennett and Lanning, 2007; Dror et al., 2012). Collaborative Filtering, one of the major techniques used among the array of available recommendation strategies, can also be seen as a combination of several utility subfunctions, each corresponding to one neighbour (in user-based CF). In the same way performance prediction in Information Retrieval has been used to optimise rank aggregation, we have investigated the use of recommendation performance predictors to dynamically aggregate the output of recommenders and neighbours.

We have defined a **dynamic hybrid framework** where recommender ensembles can benefit from dynamic weights according to performance predictors with which strong correlations have been found. Our results indicate that high correlation with performance tends to correspond with enhancements in dynamic hybrid recommenders. Additionally, dynamic ensembles of recommenders usually outperform

baseline static ensembles for different recommender combinations and the three types of performance predictors investigated.

On the other hand, we have also proposed a **framework for neighbour selection and weighting** in user-based recommender systems. We have defined neighbour performance predictors and metrics by adapting and integrating some of the methods from the trust-aware recommendation literature. Our framework unifies several notions of neighbour performance under the same view, and provides an objective analysis of the predictive power of different neighbour scoring functions. Once the predictive power of these neighbour predictors was confirmed, we used them to weight the information coming from each neighbour in a dynamic fashion, by means of different strategies that combine similarity values and neighbours' weights. Our experiments confirm a correspondence between the correlation analysis and the final performance results, in the sense that the correlation values obtained between neighbour performance predictors and neighbour performance metrics anticipate which predictors will perform better when introduced into the user-based collaborative filtering algorithm.

9.2 Future work

Performance prediction in recommendation is an interesting research topic also from a business perspective, since one could decide when to deliver certain item recommendations to a user, avoiding lowering the user's confidence on the relevance of the recommendations. In this sense, performance predictions of potential recommendations may give control to the service provider; a control that could be used in various ways, such as recommendation combination methods more general than those addressed in this thesis. Regardless of the plausible applications for industry, and beyond the achievements presented throughout the thesis, we envision the following potential future research lines.

The evaluation of recommender systems still is an object of active research in the field, where several questions need more attention, such as the gap between offline and online experiments, and the missing not at random assumption. Nonetheless, in this thesis we have focused our research on aspects related to the prediction of performance, which requires a deeper understanding of the evaluation methodologies used. In this way, we could **extend our analysis of evaluation methodologies to other ranking metrics** such as those based on two rankings (NDPM, and Spearman's and Kendall's correlations) and those adapted from Machine Learning (e.g. AUC). In this way, we may find that one of these metrics is not influenced by any of the confounders described in Chapter 4, or that none of the design alternatives proposed are able to neutralise these effects. As an example of the interest of this topic, recently in (Pradel et al., 2012) the authors analysed the popularity effects over the

AUC metric and found that considering missing data as a form of negative feedback during training may improve performance, although it may also favour popularity-based recommenders over personalised recommendation methods.

Additionally, it would be beneficial for our research to be able to validate the usefulness of the unbiased measurement of performance with **online evaluations**. This would be valuable for a comparative assessment of the offline observations along with a deeper understanding of the extent to which popularity may be or not a noisy signal. Such a user study would help us determine the real benefits (if any) of receiving popular recommendations, since, for instance, by definition these suggestions are not novel, and probably neither serendipitous nor diverse.

In Chapter 6 we have proposed several performance predictors for recommendation based on the same principles as those denoted in IR as pre-retrieval predictors, like the clarity score, where the output of the retrieval engine (or the recommender system in our case) is not used by the predictor. Based on our results, the research possibilities to investigate more performance predictors for recommendation are abundant. In this line, several authors have exploited the **combination of predictors to obtain higher correlation values and stronger predictive power**, such as (Hauff et al., 2009) and (Jones and Diaz, 2007), where penalised regression and linear regression followed by neural network learning were used respectively. In those works the combination of predictors from different nature improved the correlation against the target evaluation metric – i.e., average precision. Thus, we envision the combination of predictors as a worthwhile direction also for recommendation, especially since we have defined predictors based on different inputs that are expected to have low redundancy between them and, when possible, the combination of such predictors may produce higher correlations for different types of inputs. Examples of these combinations may be the mixture of social and temporal dimensions, item-based temporal predictors, and other contextual dimensions not addressed in this thesis.

Moreover, a future investigation could **analyse and adapt to recommender systems post-retrieval performance predictors** defined in the IR literature, such as those based on the analysis of the score distribution from the recommended items to each user. This may provide predictors with stronger correlations and, thus, with more predictive power of the recommenders' performance, as it occurrs in IR where post-retrieval predictors usually obtain higher correlation values than pre-retrieval predictors. The main limitation of this type of predictors is that they cannot be used directly to adapt the output of the recommender, since the complete output – i.e., the ranking – is typically required for the computation of the predictor values. This would require thinking of different applications where this type of predictors could be applied to recommendation

A particular direction worth considering, and also related to Chapter 6, would be the **use of alternative evaluation approaches** beyond correlation metrics, such as those based on clustering the true and estimated performance values (see Section 5.4.2). In our work we have focused on the use of correlation metrics, mainly Pearson's correlation. These metrics have well-known limitations, such as their sensitivity to outliers, and the small (not significative) differences in correlation when a small number of points is used. For this reason, other approaches for assessing the predictive power of the predictors have been proposed. We have to note, however, that the use of a particular evaluation technique should be focused on their application to specific contexts (Pérez-Iglesias and Araujo, 2010); specifically, this requires defining new applications for performance predictors that match the evaluation metric, which we also envision as a potential future work.

Besides, in the same chapter we developed an evaluation methodology to assess the true performance values of the items, in order to evaluate the proposed item predictors. This methodology should be further validated in order to **obtain a fair measure of item performance**, which at this moment is still an open problem. In that way we would be able to define additional item predictors for other input spaces apart from ratings, and improve the predictiveness of the current item performance predictors.

In Chapter 7 we presented experiments regarding the dynamic combination of recommenders in an ensemble. Those experiments were limited to only one performance predictor for a pair of recommenders. We plan to extend these experiments with **ensembles where two predictors are considered** in order to investigate which conditions should be fulfilled by each pair of predictors in order to improve the performance of the ensemble. A related research direction worth of consideration would be the analysis of the sensitivity of the correlation values for which good performance results are obtained in the dynamic hybrid methods. More specifically, we may consider whether it is better to have an overall strong correlation value (in average) or a not very strong average correlation but better estimates for some particular users, where these users would play a significant role in the system such as the power users defined in (Lathia et al., 2008). A study like the one presented in (Hauff et al., 2010) could then be conducted where simulations of predictors with different correlations are evaluated and their effect on the final performance of the ensembles is compared against each other.

Furthermore, another limitation of the experiments presented in Chapter 7 was that the size of ensembles was always two. We aim to consider **ensembles of N recommenders** and, eventually as mentioned above, using one performance predictor for each recommender. This is a natural but non-trivial step towards a generalisation of the proposed framework to larger ensemble recommenders. Alternatively, Machine Learning techniques could be used to learn the best weights to use in the en-

semble in a user and item basis. In this case, a compromise between the computational costs of each technique (machine learning against performance predictors), their predictive power and the tendency to overfitting should be investigated.

Finally, in Chapter 8 we investigated the dynamic neighbour weighting problem using neighbour performance predictors oriented to error-based metrics. The future work related to this chapter could focus on the **adaptation of the neighbour performance metrics used in our approach to ranking-based metrics**, such as precision and recall. As we have already discussed, error metrics are not the best way to measure performance, although they can be considered appropriate in this context since we want to measure the improvement in accuracy of our approaches, along with facilitating comparisons with the state of the art in trust-aware recommendation, where these metrics are prevalent. Therefore, the use of ranking metrics would be a valuable contribution to the field by itself. Furthermore, once a neighbour performance metric based on a ranking metric is provided, we would be able to measure the correlation of the neighbour predictors described in that chapter with such metric, and analyse in detail the predictive power of predictors for ranking metrics. Ideally, we would be able to obtain a predictor with enough predictive power using both types of neighbour performance metrics (based on error and ranking), although this is not easy to grant in general, since each metric is defined to optimise different parameters and concepts.

Part VI

Appendices

An algorithm must be seen to be believed.
Donald Knuth

Appendix A

Materials and methods

In this thesis we have reported results obtained in experiments conducted with a variety of input data sources and recommendation algorithms. In this appendix we provide additional details, not reported in previous chapters, about such datasets and recommenders. Hence, in Section A.1 we present statistics about the datasets, and in Section A.2 we describe the specific configuration setting of the recommenders. Next, in Section A.3 we detail the followed evaluation methodologies. Finally, in Sections A.4 and A.5 we present further results regarding prediction correlations and performance of dynamic recommender ensembles by means of metrics such as MAP@10 and nDCG@50. Despite the fact the thesis is mainly focused on ranking metrics, for the sake of brevity and clarity, in Chapters 6 and 7 we only reported experimental results based on the P@10 metric.

A.1 Datasets

In Section 3.4 we presented the three datasets used in this thesis, namely MovieLens, Last.fm, and CAMRa datasets. We now describe the specific partitions in which we splitted those datasets for experimentation. Specifically, we explain how the data splits were generated, and provide some statistics of the splits, such as their number of users and items, and their density, measured as the percentage of cells in the ratings matrix with known values, i.e., #ratings / (#users × #items) × 100.

A.1.1 MovieLens dataset

In addition to the well-known Netflix dataset, the MovieLens 100K and MovieLens 1M datasets – extracted from the MovieLens movie recommendation system by the GroupLens research group at University of Minnesota, USA – have been the most widely used datasets in the Recommender Systems field. The former has 943 users, 1,682 items, and 100,000 ratings, whereas the latter has 6,040 users, 3,900 items, and 1 million ratings. In this thesis, when we do not explicitly indicate which of the two datasets was used, we refer to the MovieLens 1M dataset.

In the experiments we always performed a 5-fold cross validation strategy to generate 5 random 80-20% disjoint splits of rating sets. As rating sets, in the **MovieLens 100K** dataset we used the partition provided in its public distribution, and in the **MovieLens 1M** dataset we used 5 splits with 200,000 ratings, each of them randomly selected.

Property	Overall	Average	Average in training	Average in test
No. users	943	854.60 (± 165.44)	943 (± 0.00)	766.20 (± 205.06)
No. items	1,682	1,531.20 (± 127.18)	1651.60 (± 4.77)	1410.80 (± 11.52)
No. ratings	100,000	50,000 (± 31,622.78)	80,000 (± 0.00)	20,000 (± 0.00)
Density	6.30%	3.56% (± 1.72)	5.14% (± 0.01)	1.99% (± 0.67)
Avg. no. rated items per user	106.04	56.46 (± 30.56)	84.84 (± 0.00)	28.09 (± 9.43)
Max. no. rated items per user	737	450.80 (± 213.68)	650.20 (± 35.37)	251.40 (± 45.59)
Min. no. rated items per user	20	3.70 (± 3.16)	6.40 (± 2.07)	1 (± 0.00)
Max. no. users rating an item	583	293.30 (± 182.82)	466.40 (± 14.06)	120.20 (± 9.71)
Min. no. users rating an item	1	1 (± 0.00)	1 (± 0.00)	1 (± 0.00)

Table A.1. Summary of statistics of the MovieLens 100K dataset.

Property	Overall	Average	Average in training	Average in test
No. users	6040	6038.40 (± 1.84)	6040 (± 0.00)	6036.80 (± 1.10)
No. items	3706	3,576.50 (± 109.48)	3,680.20 (± 6.26)	3,472.80 (± 6.61)
No. ratings	1,000,000	500,104.50 (± 316,293.86)	800,167.20 (± 0.45)	200,041.80 (± 0.45)
Density	4.47%	2.28% (± 1.39)	3.60 (± 0.01)	0.95 (± 0.00)
Avg. no. rated items per user	165.60	82.81 (± 52.36)	132.48 (± 0.00)	33.14 (± 0.01)
Max. no. rated items per user	2,314	1,157 (± 732.10)	1,851.20 (± 24.00)	462.80 (± 24.00)
Min. no. rated items per user	20	5.90 (± 5.22)	10.80 (± 1.10)	1 (± 0.00)
Max. no. users rating an item	3,428	1,714 (± 1,084.24)	2,742.40 (± 22.95)	685.60 (± 22.95)
Min. no. users rating an item	1	1 (± 0.00)	1 (± 0.00)	1 (± 0.00)

Table A.2. Summary of statistics of the MovieLens 1M dataset.

Table A.1 and Table A.2 show some statistics about the MovieLens datasets. It is interesting to note that the overall sparsity in both datasets is similar (between 4% and 6%), but the number of users vs. items is quite different: in MovieLens 100K there are more items than users, whereas this situation in MovieLens 1M is the opposite.

Finally, in those experiments where we evaluated content-based recommenders, we used extended versions of the MovieLens dataset with information extracted from the Internet Movie Database[9], such as the movies' directors, actors, and genres. Details about how this information was gathered and merged with the MovieLens user profiles can be found in (Cantador, 2008).

A.1.2 Last.fm dataset

Among the two Last.fm datasets distributed by Ò. Celma (Celma and Herrera, 2008), and described in Section 3.4.2, in this thesis we used the one denoted as Last.fm 1K, since it contains the music listening history (scrobbles) of each user at the track level, and includes the timestamp at which a user listenend to a track.

For our experiments we built two versions of that dataset. Table A.3 shows some statistics about them. For building the first version (refered as Last.fm dataset from now on), we applied a 5-fold cross-validation on 80-20% training-test data splits. For building the second version, we performed a single temporal splitting of the user scrobbles, maintaining an 80-20% training-test ratio. This is equivalent to divide the data at some timestamp in such a way that 80% of the scrobblings are con-

[9] The Internet Movie Database, IMDb, http://www.imdb.com

		Five-fold			Temporal		
Property	**Overall**	**Average**	**Average in training**	**Average in test**	**Average**	**Training**	**Test**
No. users	992	991 (± 1.70)	992 (± 0.00)	990 (± 2.00)	932 (± 16.97)	920	944
No. items	176,892	108,065.20 (± 48,266.14)	153,854.20 (± 155.01)	62,276.20 (± 199.80)	118,530 (± 43,371.10)	149,198	87,862
No. scrobblings	19,129,595	9,564,797.50 (± 6,049,542.74)	15,303,676 (± 56,393.95)	3,825,919 (± 56,393.95)	9,564,798.50 (± 8,115,999.81)	15,303,677	3,825,920
No. unique scrobblings	904,309	452,154.50 (± 285,967.77)	723,447.20 (± 313.46)	180,861.80 (± 313.46)	539,553 (± 268,287.63)	729,261	349,845
Density	10.90% 0.52%	8.12% (± 2.02) 0.38% (± 0.10)	10.03% (± 0.04) 0.47% (± 0.00)	6.21% (± 0.11) 0.29% (± 0.00)	7.88% (± 4.62) 0.48% (± 0.08)	11.15% 0.53%	4.61% 0.42%
Avg. no. scrobblings per user	19,283.87	9,645.83 (± 6,094.22)	15,427.09 (± 56.85)	3,864.57 (± 57.35)	10,343.66 (± 8,896.50)	16,634.43	4,052.88
Max.no. scrobblings per user	183,094	93,795.20 (± 55,859.24)	146,475.20 (± 7,036.93)	41,115.20 (± 5,753.13)	117,872.50 (± 84,792.71)	177,830	57,915
Min. no. scrobblings per user	2	1.30 (± 0.48)	1.60 (± 0.55)	1 (± 0.00)	1 (± 0.00)	1	1
Avg. no. scrobbled items per user	911.60	455.99 (± 288.08)	729.28 (± 0.32)	182.69 (± 0.12)	581.64 (± 298.45)	792.68	370.60
Max. no. scrobbled items per user	8,452	4,226.00 (± 2,672.76)	6,761.60 (± 0.55)	1,690.40 (± 0.55)	5,707 (± 1,554.22)	6,806	4,608
Min. no. scrobbled items per user	2	1.30 (± 0.48)	1.60 (± 0.55)	1 (± 0.00)	1 (± 0.00)	1	1
Max. no. users scrobbling an item	710	356.10 (± 233.52)	568.00 (± 10.39)	144.20 (± 7.36)	527.50 (± 146.37)	631	424
Min. no. users scrobbling an item	1	1 (± 0.00)	1 (± 0.00)	1 (± 0.00)	1 (± 0.00)	1	1
Time interval (in days)	3,150	1,586.85 (± 0.02)	1,586.85 (± 0.00)	1,586.84 (± 0.02)	1,574.87 (± 331.64)	1,340	1,809

Table A.3. Summary of statistics of the Last.fm datasets.

tained in the training split. In the built dataset (refered as Last.fm temporal dataset from now on), the above timestamp is 16th October, 2008.

In both datasets we aggregated the user listening data by artist (amounting to $|\mathcal{I}| = 176{,}892$ instead of $|\mathcal{I}| \approx 960{,}000$ if tracks were used) in order to overcome the sparsity at track level. Apart from that, it is also interesting to note the difference between considering separate scrobblings (when a user listens to an artist, in our setting) against considering unique scrobblings (or number of user-item pairs), which corresponds to the typical situation in movie recommendation (where, for each movie, there are not several ratings given by a particular user).

Furthermore, in some experiments we transformed log-based information to explicit ratings by using the method described in (Celma, 2010) and (Celma, 2008). This method takes into account the number of times a user listened to an artist, in such a way that the artists located in the 80-100% interquintile range of the user's listening distribution receive a rating of 5 (in a five point scale), those in the next interquintile range are mapped to a rating of 4, and so on. Additionally, the use of the coefficient of variation $CV(u) = \sigma(u)/\mu(u)$ is proposed in (Celma, 2008) to discriminate between skewed and uniform distributions. We have not considered this coefficient since it produced strange behaviours in the recommenders, such as too many ties in the recommended items and errors in the computation of some correlations (since the mean would have the same value for every rating, see Equation (2.5)).

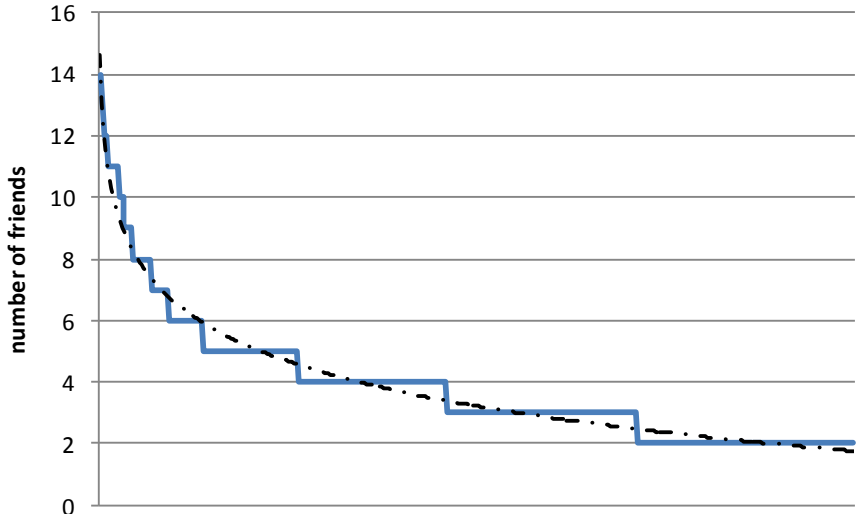


Figure A.1. Friend distribution among the users composing the original test set of the CAMRa '10 social track. Note that a logarithmic regression line fits almost perfectly with the above distribution. The total number of users is 439, and the maximum and minimum numbers of friends are 14 and 2 respectively.

On this dataset, content-based recommenders used the artists' tags. The considered tags for each artist were the most popular tags assigned to that artist according to the Last.fm API[10].

A.1.3 CAMRa dataset

Among the several datasets provided in the different CAMRa challenges (Said et al., 2010) (Said et al., 2011), in this thesis we used the dataset published at the social track of CAMRa '10. This dataset was gathered from the Filmtipset community, and contains social links between users, movie ratings, movie comments, and other attributes of users and movies. The design of this dataset and, more specifically, the selection of test users as provided in the challenge's social track is representative of online applications in which every target user has a non-empty list of contacts (see Figure A.1). This is the case of social-centric systems such as Facebook, Linkedin, and Twitter, but not of many social media applications, such as Delicious and Last.fm, where the coverage of their social network is partial – not all registerd users really use the social part of the system.

In fact, the Filmtipset dataset belongs to the latter case. Considering the set of all users, more than 10,000 out of about 16,500 users do not have any friends in the system. The number of contacts per user follows a power law distribution, where the average number of friends per user is 0.95, and the mode among users (with at least one friend) is 1. If we take this dataset as representative of social media systems, the

[10] Documentation related with the method used, http://www.lastfm.es/api/show/artist.getTopTags

		Social		Collaborative-Social	
Property	**Overall**	**Training**	**Test**	**Training**	**Test**
No. users	16,473	16,473	439	16,473	793
No. items	24,222	24,222	1,915	24,212	2670
No. ratings	3,091,075	3,075,346	15,729	3,069,888	21,187
Density	0.77%	0.77%	1.87%	0.77%	1.00%
Avg. no. rated items per user	187.64	186.69	35.83	186.36	26.72
Max. no. rated items per user	3,435	3,435	314	3,435	314
Min. no. rated items per user	1	1	1	1	1
Max. no. users rating an item	14,339	14,290	111	14,255	134
Min. no. users rating an item	1	1	1	1	1
Avg no. friends per user	0.95	0.95	3.85	0.95	2.13
Max. no. friends per user	24	24	14	24	14
Min. no. friends per user	0	0	2	0	0

Table A.4. Summary of statistics of the CAMRa datasets.

presence of contacts by itself does not guarantee the accuracy of social recommendation. Moreover, intermediate cases, where social data is available but not enough to support optimal recommendation, would rather seem to be the norm. We therefore simulate an alternative scenario by adding an equal amount of users without friends to a new test set by sampling randomly the same number of test users in the original test set (i.e., 439 users), but forcing them to have no friends. We name the original dataset as CAMRa Social, and the modified one as CAMRa Collaborative-Social or simply CAMRa Collaborative, since it is not focused on social information. Table A.4 shows some statistics about these datasets, from which the minimum number of friends per user in the test sets is the main difference between the above datasets (2 in the original, social dataset, and 0 in the collaborative test).

A.2 Configuration of recommendation algorithms

In this section we provide details about the implementation of the recommendation algorithms used in Chapters 4, 6, 7, and 8. Table A.5 shows a summary of such recommenders, along with references to the chapters where the recommenders were evaluated. In the following we describe each of these recommenders, and provide their parameter values, explicit formulations, and references. The recommenders, categorised according to the type of recommendations they provide, are the following:

Recommender	Chapter(s) where the recommender is evaluated
CB	Chapters 6 and 7
IB	Chapters 6 and 7
ItemPop	Chapters 4, 6, and 7
kNN	Chapters 4, 6, and 7
MF	Chapter 4
pLSA	Chapters 4, 6, and 7
Random	Chapters 4 and 6
Resnick	Chapter 8
TFL1	Chapters 6 and 7
TFL2	Chapters 6 and 7

Table A.5. List of the recommenders evaluated in this thesis, and the chapters where their evaluations are reported.

Non-personalised recommenders

- **ItemPop**: a non-personalised recommender based on the popularity of the item being recommended. It basically counts the number of training ratings of an item, and generates a recommendation score for the item based on such number.
- **Random**: a non-personalised recommender that generates a random recommendation score in a required value range.

Content-based recommender

- **CB**: a content-based recommender, which, similarly to the one described in (Martinez et al., 2009), computes the cosine similarity between user and item vectors whose components can represent any possible content-based feature. A different configuration of features is used depending on the dataset. For the MovieLens dataset, we used item attributes such as movie genre, director, and country, as appeared in IMDb. Other features like actors and keywords were also tested, but yielded worse performance. Specifically, we used the most popular 3 countries for each item, 3 directors, and 8 genres per movie, as suggested in (Cantador, 2008); besides, each of these features was weighted as follows: the country feature was assigned a weight of 0.26, director, 0.06, and genre, 0.66. For the Last.fm dataset, we used as features the 50 most popular tags related to each artist.

Rating-based recommenders

- **IB**: an item-based collaborative filtering recommender, in which Equation (2.8) is used along with Pearson's correlation as the similarity metric between items

(analogous to Equation (2.5), but considering items instead of users). Moreover, a constraint was implemented in order to remove some noise from neighbour items: only predictions produced by at least two similar items were considered; that is, if we replaced S_i by $\mathcal{I}_u$, then those users with only one item similar to the target item i in their profiles did not receive any recommendation.

- **kNN**: a user-based (nearest neighbour) collaborative filtering recommender, in which a slight modification of Equation (2.3) is used to adapt the item-based algorithm proposed in (Koren, 2008). We used 100 neighbours and Pearson's correlation as similarity metric, as defined in Equation (2.5). Specifically, we considered the following equation:

$$g(u,i) = b(u,i) + \frac{1}{\sum_{v \in N_k(u,i)} sim_{sh}(u,v)} \sum_{v \in N_k(u,i)} sim_{sh}(u,v)\big(r(v,i) - b(v,i)\big)$$

where $sim_{sh}(u,v)$ is the shrunk similarity $sim(u,v)n(u,v)/(n(u,v)+\lambda_2)$, $n(u,v)$ being $|\mathcal{I}_u \cap \mathcal{I}_v|$, that is, the number of users who rated both items. Besides, $b(u,i) = \mu + b_u + b_i$ is the user-item bias learnt solving the following least squares problem, where μ indicates the overall average rating:

$$\min_{b_u, b_i} \sum_{(u,i)} \left[(r(u,i) - \mu - b_u - b_i)^2 + \lambda_1 \left(\sum_u b_u^2 + \sum_i b_i^2 \right) \right]$$

In our experiments no tuning of the λ_1 regularisation parameter was done (partially because this method indeed optimises RMSE, which is not our main goal), and used a fixed value of 0.02. We used a learning rate of 0.005, and 100 iterations in the optimisation process. Furthermore, we did not shrink the similarity, so an effective $\lambda_2 = 0$ was used. Additionally, at least 5 neighbours had to participate in the prediction process to consider a predicted item as valid.

- **MF**: a matrix factorisation recommender, as implemented in the Mahout library by means of the Expectation Maximisation (EM) algorithm. We used 100 iterations and 50 features, leaving the rest of the parameters as default (i.e., 0.005 as learning rate, 0.02 as regularisation parameter, and 0.005 as random noise). We have to note that the original method proposed in (Koren et al., 2009) used Alternating Least Squares to learn the user and item factorised vectors. However, the version implemented in Mahout provided worse results for this learning method, and thus we used the EM algorithm in our experiments.

- **pLSA**: a probabilistic Latent Semantic Analysis recommender, in which we use the co-occurrence latent semantic model as defined in (Hofmann, 2004). That

is, the prediction is made by computing $p(i|u) = \sum_z p(i|z)p(z|u)$. The models based on ratings produced worse results, and are not applicable with implicit (log-based) user preference data. This is why we preferred to use the co-occurrence model over the one based on ratings. In the experiments we used 50 factors and 50 iterations.

- **Resnick**: a user-based (neearest neighbour) collaborative filtering recommender, implemented as in Equation (2.3), where rating deviations from the user's and neighbour's rating means are considered (Resnick et al., 1994). As discussed in Chapter 8, different neighbourhood sizes were tested. We used Pearson's correlation as the user similarity metric, like in Equation (2.5).
- **TFL1**: an item-based collaborative filtering recommender, which uses the TF method with normalisation s_{00}, as defined in (Bellogín et al., 2011b). This is equivalent to an standard item-based CF algorithm without dividing by the similarities values, that is:

$$\mathrm{g}(u,i) = \sum_{j \in S_i} sim(i,j) r(u,j)$$

  We did not consider the neighbourhood size, replacing S_i by $\mathcal{I}_u$, and using Pearson's correlation as similarity metric $sim(i,j)$.
- **TFL2**: an item-based collaborative filtering recommender in which, similarly to the TFL1 recommender, we used the TF method with normalisation s_{01}, and L_2 norm as defined in (Bellogín et al., 2011b). This is equivalent to an standard item-based CF, but instead of normalising by the sum of similarity values, it divides by the square root of the sum of similarity values squared, that is:

$$\mathrm{g}(u,i) = \frac{1}{\sqrt{\sum_{j \in S_i} sim(i,j)^2}} \sum_{j \in S_i} sim(i,j) r(u,j)$$

  Like before, we did not consider the neighbourhood size, and used Pearson's correlation as the similarity metric.

Social-based recommenders

- **Personal**: a social filtering recommender presented in (Ben-Shimon et al., 2007), which utilises Equation (2.11) for score prediction. We set $K = 2$ and $L = 6$, along with a constraint specifying that at least two friends have to participate on the item prediction in order to be considered valid.
- **PureSocial**: a social filtering recommender, which is an adaptation of the standard user-based collaborative filtering in which friends are used as neighbours,

as proposed in (Liu and Lee, 2010) and (Bellogín et al., 2012). Specifically, we used Equation (2.4) as the user-based method, but where the neighbourhood $N_k(u,i)$ is built by means of the user's social network. No neighbourhood size is used, and, like in the previous recommender, at least two friends have to participate on the item prediction to consider the suggestion as a valid one.

We implemented these recommenders on top of the Mahout library[11], and will make the developed source code publicly available at the following URL: http://ir.ii.uam.es/~alejandro/thesis.

Additionally, we implemented the static and dynamic weighted recommender ensembles evaluated in Chapter 7 using the rank fusion library provided in (Fernández et al., 2006a) and (Fernández et al., 2006b). This library contains a series of techniques for the two basic stages of any rank fusion problem: score normalisation and score combination. In this thesis we conducted the following procedure: a) taking each item ranking (in a user basis) for every recommender in an ensemble, b) normalising each ranking using either rank or score normalisation techniques (Renda and Straccia, 2003), c) combining the normalised rankings using a weighted sum to compute the score of each item (i.e., combSUM method); the weight assigned to each ranking may come from a performance predictor, as explained in Section 7.2.3, and d) ranking the items according to the scores produced in the last combination stage.

A.3 Configuration of evaluation methodologies

In Chapter 4 we presented several evaluation methodologies that have been further elaborated and used in the experiments of Chapters 6 and 7. In this section we provide specific examples regarding how these methodologies build the set of items to recommend. We also indicate the value of the parameters required by each method that have been used in the different chapters of this dissertation.

In Chapter 4 we classified the different target item selection strategies according to three design settings: the base candidate settings (AI or TI), the relevant item selection (AR or 1R), and the non-relevant item selection (AN or NN). Table A.6 shows the specific configurations of these methodologies, according to the notation introduced in Chapter 4. Note that the uniform methodologies (U1R, UAR, and uuUAR) take as additional parameters η and ξ; for the MovieLens 1M dataset, these parameters took the following values: $\eta = 118$, $\nu = 350$, and $\xi = 99$.

[11] Available at http://mahout.apache.org

Methodology	Base candidate	Relevant item selection	Non-relevant item selection	Configuration	Chapters
1R	TI	1R	NN	$N_u = 99$	4, 6, and 7
AR	TI	AR	AN		4, 6, and 7
P1R	TI	1R	NN	$N_u = 99$ $m = 10$	4, 6, and 7
U1R	TI	1R	NN	$N_u = 99$ $r_{test}(i) = \eta, \forall i$	4, 6, and 7
UAR	AI/TI	AR	AN	$r_{test}(i) = \eta, \forall i$	4
uuU1R	TI	1R	NN	$N_u = 99$ $r_{test}(i) = \nu, \forall i$ $r_{test}(u) = \xi, \forall u$	6 and 7

Table A.6. Configuration of the evaluation methodologies used in the thesis.

In the following we present several toy examples to illustrate how the different methodologies behave. For these examples we consider three users as follows:

- A user u_1 with training set $R_{train}(u_1) = \{i_1, i_2\}$, and test set $R_{test}(u_1) = \{i_3, i_4\}$.
- A user u_2 with $R_{train}(u_2) = \{i_1, i_2, i_3\}$, and test set $R_{test}(u_2) = \{i_5\}$.
- A user u_3 with $R_{train}(u_3) = \{i_3, i_4, i_5\}$, and test set $R_{test}(u_3) = \{i_1, i_2\}$.

According to these training and test data, we next compare the target items selected for user u_1 by 1R and AR methodologies. The AR methodology selects all items contained in the test set (TI-AN) as non-relevant, and all items in the active user's test (AR); hence, it produces the following set to be scored and ranked by a given recommender: $\{i_3, i_4, i_5\}$. The 1R methodology, on the other hand, produces a ranking for each relevant item as follows: $\{i_3, i_5\}$ for item i_3, and $\{i_4, i_5\}$ for item i_4, where we use $N_u = 1$ for illustration purposes.

In the example we also have that $r(u_1) = r(u_2) = 4$, and $r(u_3) = 5$, and for the items $r(i_1) = r(i_2) = r(i_3) = 3$ and $r(i_4) = r(i_5) = 2$, where $r(u)$ denotes the number of items rated by a user (and similarly for items) as denoted in Chapter 4. With the uniform methodologies we have to build a different training/test split for each user. Specifically, in the U1R and UAR methodologies we need to ensure that every item has been rated the same number of times in the test, whereas in the uuU1R methodology we also have the constraint that all users have to appear the same number of times in the test set. Therefore, the following split would be valid for the U1R or UAR methodologies:

$$R_{train}(u_1) = \{i_4\};\ R_{test}(u_1) = \{i_1, i_2, i_3\}$$
$$R_{train}(u_2) = \{i_2, i_3, i_5\};\ R_{test}(u_2) = \{i_1\}$$
$$R_{train}(u_3) = \{i_1, i_4, i_5\};\ R_{test}(u_3) = \{i_2, i_3\}$$

In this case $\eta = 2$ for every item in the test set. Then, we can apply the 1R or AR methodology to obtain the corresponding rankings for the U1R or UAR meth-

odologies, respectively. However, since we have a different amount of ratings for each user, this configuration is not valid for the uuU1R methodology. A valid configuration for this methodology would be:

$$R_{train}(u_1) = \{i_3, i_4\};\ R_{test}(u_1) = \{i_1, i_2\}$$
$$R_{train}(u_2) = \{i_2, i_5\};\ R_{test}(u_2) = \{i_1, i_3\}$$
$$R_{train}(u_3) = \{i_1, i_4, i_5\};\ R_{test}(u_3) = \{i_2, i_3\}$$

Here, we have $\nu = 2$ and $\xi = 2$ for all the users and items in the test set. In this context, if we apply the 1R methodology to this split we would obtain results corresponding to the uuU1R methodology.

Alternatively, for uuU1R we can also derive a valid configuration directly from a given uniform split, i.e., the one used for U1R and UAR. To do this, we would exploit the degree of freedom we have to select the set of not-relevant items for each user. Hence, we have to guarantee that all the non-relevant items selected for each user have to appear the same number of times in every target set to be recommended. This constraint would be satisfied depending on the user and item distributions, where a greedy algorithm (by brute force) could be applied if enough ratings are available. For instance, in our previous example we cannot derive a valid configuration from the split presented for U1R and UAR. In such a case, an alternative split should be made as explained above.

For simplicity purposes, in the examples we have assumed that all items contained in the test set are relevant. However, this is not the general case, and a threshold on the minimum rating value should be set. To be consistent across methodologies, only items rated as 5 by a user are considered relevant. Thus, for the AR methodology, although every item in a user's test set is considered, in the evaluation only those items with 5 stars are considered relevant. On the other hand, the 1R methodology builds one ranking for each relevant item. That is, in the example above, if $r(u_1, i_3) = 5$ and $r(u_1, i_4) = 3$, then a unique ranking would be generated and evaluated, the one corresponding to $\{i_3, i_5\}$.

As additional material, we will make publicly available an implementation of the above evaluation methodologies at http://ir.ii.uam.es/~alejandro/thesis.

A.4 Additional results about correlations

Next we report additional experiments to those presented in Chapter 6 regarding the computation of correlation coefficients between the performance predictors and recommenders. Specifically, we provide experimental results obtained with metrics different to P@10, and alternative correlation coefficients such as Spearman's and Kendall's.

A.4.1 Correlations of user predictors using rating data

In this section we provide results obtained by using different correlation coefficients and metrics in the evaluation of user predictors based on ratings. Specifically, we provide results obtained with Spearman's ρ and Kendall's τ correlation coefficients, and metrics such as MAP@10 (because it is considered as more stable than precision (Manning et al., 2008; Baeza-Yates and Ribeiro-Neto, 2011)), recall@10 (as an inversely related metric to precision), and nDCG@50 (because it includes graded relevance and a different cutoff).

Table A.7, Table A.8, and Table A.9 show the results obtained when the above three metrics are used along with the Pearson's correlation coefficient and the AR methodology. Comparing these results with those shown in Table 6.7, we can observe that most of the correlation trends are similar to the ones presented in Chapter 6. Specifically, the results with nDCG@50 are almost equivalent to those obtained with P@10, proving that our predictors are also consistent with other metrics apart from precision. On the other hand, the correlation values with MAP@10 and recall@10 metrics have some differences with respect to the P@10 metric, being lower in general. In fact, the correlation with CB and pLSA recommenders is negative, probably due to the (inverse) relation between precision and recall, which makes very difficult to optimise both metrics at the same time.

Predictor	Random	CB	IB	ItemPop	kNN	pLSA	TFL1	TFL2
Count (training)	0.005	-0.031	0.009	0.047	0.094	-0.049	0.024	0.281
Count (test)	0.004	-0.029	0.009	0.047	0.092	-0.052	0.022	0.276
Mean	0.009	0.010	-0.002	-0.065	-0.036	-0.002	0.007	-0.103
Standard deviation	0.004	0.013	0.011	-0.018	-0.029	-0.023	0.015	-0.069
ItemSimple Clarity	0.007	-0.031	0.010	0.040	0.089	-0.054	0.026	0.274
ItemUser Clarity	0.005	-0.029	0.011	0.039	0.089	-0.054	0.028	0.272
RatUser Clarity	0.006	-0.031	0.009	0.048	0.093	-0.049	0.024	0.273
RatItem Clarity	0.005	-0.029	0.008	0.041	0.087	-0.053	0.022	0.267
IRUser Clarity	0.006	-0.026	0.005	0.051	0.083	-0.046	0.021	0.265
IRItem Clarity	0.003	-0.021	0.009	0.038	0.076	-0.046	0.023	0.234
IRUserItem Clarity	0.005	-0.025	0.007	0.049	0.082	-0.047	0.024	0.261
Entropy	0.000	-0.032	0.007	0.040	0.103	-0.037	0.021	0.296

Table A.7. Pearson's correlation values between rating-based user predictors and MAP@10 for different recommenders, using the AR methodology on the MovieLens 1M dataset.

Predictor	Random	CB	IB	ItemPop	kNN	pLSA	TFL1	TFL2
Count (training)	-0.001	-0.048	0.009	0.012	0.053	-0.120	0.020	0.213
Count (test)	-0.002	-0.047	0.008	0.009	0.051	-0.123	0.018	0.208
Mean	0.009	0.009	-0.002	-0.067	-0.024	-0.017	0.005	-0.104
Standard deviation	0.004	0.014	0.011	-0.038	-0.034	-0.035	0.019	-0.076
ItemSimple Clarity	0.000	-0.049	0.009	0.000	0.047	-0.131	0.023	0.203
ItemUser Clarity	-0.001	-0.046	0.011	0.003	0.049	-0.121	0.025	0.205
RatUser Clarity	0.000	-0.049	0.009	0.011	0.055	-0.116	0.020	0.201
RatItem Clarity	-0.002	-0.046	0.007	0.008	0.050	-0.115	0.019	0.199
IRUser Clarity	0.001	-0.040	0.004	0.018	0.047	-0.108	0.018	0.201
IRItem Clarity	-0.002	-0.036	0.006	0.006	0.039	-0.106	0.020	0.176
IRUserItem Clarity	0.000	-0.040	0.006	0.016	0.046	-0.109	0.021	0.197
Entropy	-0.004	-0.048	0.006	0.010	0.058	-0.124	0.018	0.252

Table A.8. Pearson's correlation values between rating-based user predictors and recall@10 for different recommenders, on the MovieLens 1M dataset and using the AR methodology

Predictor	Random	CB	IB	ItemPop	kNN	pLSA	TFL1	TFL2
Count (training)	0.085	0.012	0.010	0.221	0.228	0.144	0.152	0.528
Count (test)	0.085	0.015	0.010	0.225	0.231	0.148	0.153	0.525
Mean	0.023	0.037	-0.014	-0.035	0.011	0.050	0.040	-0.084
Standard deviation	0.003	0.019	0.010	-0.046	-0.069	-0.048	0.021	-0.096
ItemSimple Clarity	0.094	0.020	0.011	0.226	0.235	0.156	0.173	0.540
ItemUser Clarity	0.084	0.014	0.013	0.205	0.220	0.134	0.167	0.513
RatUser Clarity	0.087	0.005	0.010	0.221	0.240	0.139	0.160	0.517
RatItem Clarity	0.081	0.008	0.008	0.201	0.218	0.123	0.158	0.493
IRUser Clarity	0.079	0.022	-0.001	0.216	0.201	0.136	0.138	0.492
IRItem Clarity	0.074	0.032	0.007	0.184	0.173	0.119	0.145	0.440
IRUserItem Clarity	0.079	0.023	0.003	0.212	0.198	0.133	0.146	0.485
Entropy	0.078	0.025	0.004	0.224	0.235	0.192	0.127	0.561

Table A.9. Pearson's correlation values between rating-based predictors and nDCG@50 for different recommenders, on the MovieLens 1M dataset and using the AR methodology.

As pointed out by other authors, Pearson's correlation values by themselves may not be enough to completely understand the relation between analysed variables (Hauff et al., 2009; Carmel and Yom-Tov, 2010). Because of that, in Table A.10 and Table A.11 we provide results found when Spearman's and Kendall's correlations are used, along with the P@10 metric and the AR methodology. Comparing these results with those shown in Table 6.7, we can observe that strong correlations are also obtained using non parametric coefficients such as Kendall's τ. More specifically, our results show that $r \geq \rho \geq \tau$, with respect to the Pearson's r, Spearman's ρ, and Kendall's τ correlation coefficients.

Predictor	Random	CB	IB	ItemPop	kNN	pLSA	TFL1	TFL2
Count (training)	0.112	0.165	0.020	0.457	0.367	0.465	0.124	0.585
Count (test)	0.114	0.174	0.020	0.473	0.375	0.484	0.124	0.589
Mean	0.015	0.064	0.000	0.010	0.025	0.106	0.030	-0.024
Standard deviation	0.007	0.005	0.009	-0.032	-0.038	-0.037	0.009	-0.064
ItemSimple Clarity	0.117	0.176	0.021	0.474	0.382	0.492	0.130	0.602
ItemUser Clarity	0.112	0.168	0.021	0.442	0.362	0.457	0.131	0.583
RatUser Clarity	0.113	0.157	0.021	0.469	0.388	0.482	0.122	0.598
RatItem Clarity	0.113	0.169	0.019	0.460	0.378	0.477	0.130	0.592
IRUser Clarity	0.110	0.164	0.019	0.449	0.364	0.454	0.123	0.571
IRItem Clarity	0.107	0.174	0.017	0.422	0.318	0.414	0.123	0.532
IRUserItem Clarity	0.110	0.164	0.020	0.447	0.362	0.453	0.125	0.571
Entropy	0.112	0.166	0.020	0.460	0.369	0.468	0.124	0.588

Table A.10. Spearman's correlation values between rating-based user predictors and P@10 for different recommenders, on the MovieLens 1M dataset and using the AR methodology.

Predictor	Random	CB	IB	ItemPop	kNN	pLSA	TFL1	TFL2
Count (training)	0.092	0.134	0.017	0.358	0.296	0.353	0.102	0.471
Count (test)	0.094	0.143	0.017	0.373	0.305	0.371	0.102	0.477
Mean	0.013	0.052	0.000	0.008	0.020	0.079	0.025	-0.018
Standard deviation	0.006	0.004	0.008	-0.024	-0.030	-0.028	0.007	-0.050
ItemSimple Clarity	0.096	0.143	0.017	0.371	0.308	0.374	0.106	0.484
ItemUser Clarity	0.091	0.136	0.018	0.344	0.292	0.346	0.107	0.467
RatUser Clarity	0.093	0.127	0.017	0.367	0.313	0.366	0.100	0.481
RatItem Clarity	0.092	0.137	0.016	0.359	0.304	0.362	0.106	0.476
IRUser Clarity	0.090	0.133	0.015	0.350	0.293	0.344	0.100	0.457
IRItem Clarity	0.087	0.141	0.014	0.328	0.255	0.312	0.101	0.423
IRUserItem Clarity	0.090	0.133	0.017	0.348	0.292	0.343	0.102	0.456
Entropy	0.092	0.134	0.017	0.359	0.297	0.355	0.101	0.472

Table A.11. Kendall's correlation values between rating-based user predictors and P@10 for different recommenders, on the MovieLens 1M dataset and using the AR methodology.

Predictor	Random	CB	IB	kNN	TFL1	TFL2
Count (training)	0.288	0.255	0.170	0.488	0.529	0.545
Count (test)	0.384	0.384	0.242	0.592	0.602	0.623
Mean	-0.063	-0.009	-0.060	-0.018	-0.103	0.012
Standard deviation	-0.011	-0.033	0.045	-0.016	0.031	-0.135
ItemSimple Clarity	0.282	0.257	0.146	0.491	0.521	0.564
ItemUser Clarity	0.289	0.255	0.189	0.479	0.540	0.530
RatUser Clarity	0.282	0.234	0.182	0.469	0.507	0.516
RatItem Clarity	0.239	0.191	0.184	0.395	0.442	0.426
IRUser Clarity	0.149	0.171	-0.092	0.257	0.253	0.399
IRItem Clarity	0.232	0.218	0.152	0.372	0.453	0.416
IRUserItem Clarity	0.279	0.265	0.105	0.444	0.523	0.545
Entropy	0.263	0.256	0.110	0.497	0.499	0.574

Table A.12. Pearson's correlation values between rating-based user predictors and nDCG@50 for different recommenders, on the MovieLens 100K dataset and using the AR methodology.

We also compare the results shown in Table A.9 with those obtained on different datasets. Table A.12 shows the correlation values for the MovieLens 100K dataset, as published in (Bellogín et al., 2011b). We observe that the correlation does not

Predictor	Random	CB	IB	ItemPop	kNN	pLSA	TFL1	TFL2
Count (training)	0.389	0.464	0.100	0.718	0.553	0.697	0.490	0.793
Count (test)	0.392	0.475	0.100	0.728	0.562	0.711	0.492	0.797
Mean	-0.093	-0.151	-0.022	-0.117	-0.041	-0.152	-0.123	-0.104
Standard deviation	0.017	0.036	0.009	-0.029	-0.045	-0.025	0.018	-0.069
ItemSimple Clarity	0.383	0.453	0.100	0.721	0.563	0.700	0.478	0.802
ItemUser Clarity	0.391	0.465	0.112	0.696	0.544	0.678	0.514	0.783
RatUser Clarity	0.387	0.442	0.116	0.702	0.557	0.665	0.498	0.788
RatItem Clarity	0.366	0.426	0.096	0.663	0.524	0.633	0.500	0.765
IRUser Clarity	0.376	0.469	0.068	0.671	0.498	0.664	0.499	0.742
IRItem Clarity	0.358	0.443	0.082	0.622	0.469	0.609	0.482	0.684
IRUserItem Clarity	0.378	0.470	0.083	0.664	0.494	0.658	0.513	0.736
Entropy	0.313	0.442	0.056	0.687	0.508	0.747	0.341	0.710

Table A.13. Pearson's correlation values between rating-based user predictors and P@10 for different recommenders, on the MovieLens 1M dataset and using the AR methodology, but considering all the items in test set as relevant.

change significantly; only the correlation values for the CB, IB and TFL1 recommenders increase in the MovieLens 100K dataset.

Additionally, as shown in Chapter 6, when different experimental designs are tested, the selection of relevant and not relevant items is very important. In the AR methodology, as described in Section A.3, we consider as relevant those items whose ratings are 5 (to some extent in order to be consistent with the rest of the methodologies). Table A.13, on the other hand, shows that the correlation changes when all the items in the test set are considered relevant. The first thing to note when we compare these results with those shown in Table 6.7 is that the absolute correlation values are now much higher. However, if we take the correlations with the Random recommender as a reference, these relative correlations are very similar in both tables. Moreover, the trend in predictive power of the predictors (that is, which predictors have more or less predictive power) is consistent across this dimension, which evidences the stability of the evaluation methodology used to measure the predictive power of the recommendation performance predictors.

A.4.2 Correlations of user predictors using log data

In this section we focus on complementing our results with correlations between the predictor values and the MAP@10 metric, since we have observed in the previous sections that other correlation coefficients are consistent with Pearson's.

Hence, in Table A.14 we report the results with the temporal split, whereas in Table A.15 we present the results for the random split on the Last.fm dataset. Respectively, we have to compare these tables against Table 6.14 and Table 6.15. This comparison shows that the results obtained using the precision and MAP metrics are equivalent, in contrast to what happened in the previous sections. This is because in this experiment we use the 1R methodology where there is only one relevant item, in

Predictor	Random	CB	ItemPop	kNN	pLSA
Average Count	0.001	0.173	0.027	-0.054	0.163
Count	0.028	0.188	-0.044	0.140	0.115
Mean	-0.059	-0.407	0.037	-0.089	-0.220
Standard deviation	-0.034	-0.191	-0.035	-0.119	-0.094
Autocorrelation	0.049	0.156	-0.079	-0.105	0.001
TimeSimple Clarity	-0.044	-0.435	0.053	-0.257	-0.212
ItemTime Clarity	0.024	0.142	0.085	0.316	0.084
ItemPriorTime Clarity	0.069	0.219	0.240	0.345	0.204
Frequency Clarity	-0.039	-0.421	-0.248	-0.307	-0.365
ItemSimple Clarity	0.008	0.118	-0.052	0.275	0.014

Table A.14. Pearson's correlation values between log-based user predictors and MAP@10 for different recommenders, on the Last.fm temporal dataset and using the 1R methodology.

Predictor	Random	CB	ItemPop	kNN	pLSA
Average Count	-0.043	-0.065	-0.139	0.020	-0.136
Count	-0.031	-0.212	-0.194	-0.040	-0.260
Mean	0.004	0.173	0.111	0.008	0.117
Standard deviation	-0.010	0.116	0.116	0.021	0.099
Autocorrelation	0.010	-0.032	-0.078	-0.007	-0.063
TimeSimple Clarity	0.041	0.304	0.277	0.082	0.344
ItemTime Clarity	0.037	-0.147	0.020	0.052	-0.084
ItemPriorTime Clarity	0.036	-0.028	0.210	0.149	0.072
Frequency Clarity	0.001	-0.038	-0.286	-0.158	-0.211
ItemSimple Clarity	0.028	-0.240	-0.131	-0.021	-0.213

Table A.15. Pearson's correlation values between log-based user predictors and MAP@10 for different recommenders, on the Last.fm five-fold dataset and using the 1R methodology.

contrast to the AR methodology used in that experiment, and thus, these two metrics are equivalent. Similar results are obtained for recall and nDCG metrics for the same reasons.

A.4.3 Correlations of user predictors using social data

In this section we present additional results regarding the correlations between social-based performance predictors and evaluation metrics. Like in the previous section, here we only show correlations with respect to the MAP@10 metric, which, in this case, do not provide results equal to those of the precision metric since we use the AR methodology instead of the 1R methodology.

Table A.16 shows Pearson's correlation values on the social version of the CAMRa dataset. We observe that the correlations are much lower than those presented in Table 6.16; in some situations even the sign of the correlation changes, like for most of the values of kNN. A more interesting situation is observed on the CAMRa Collaborative dataset (Table A.17), where strong but negative correlations arise, in particular for the ItemPop and pLSA recommenders. This result may have a direct impact on the performance of the dynamic ensembles, since the correlation of

Predictor	Random	ItemPop	kNN	pLSA	Personal	PureSocial
Count (training)	0.016	0.104	0.022	0.024	0.013	0.066
Count (test)	0.086	-0.032	0.021	-0.167	-0.215	-0.294
Mean	-0.047	0.074	0.075	0.009	0.003	0.025
Standard deviation	-0.030	-0.065	-0.144	-0.061	-0.063	-0.004
Avg neighbour degree	-0.025	-0.057	-0.031	-0.124	-0.120	-0.240
Betweenness centrality	-0.015	0.061	-0.010	-0.015	0.011	-0.076
Clustering coefficient	0.028	-0.024	0.035	-0.106	0.001	-0.133
Degree	-0.026	-0.070	-0.065	-0.130	-0.184	-0.185
Ego components size	-0.044	0.019	-0.051	0.029	-0.036	0.021
HITS	-0.010	-0.018	0.056	-0.002	0.082	0.040
PageRank	-0.020	-0.022	-0.047	0.041	-0.059	0.041
Two-hop neighbourhood	-0.039	-0.040	-0.062	-0.118	-0.148	-0.227
ItemSimple Clarity	0.012	0.132	0.031	0.037	0.023	0.080

Table A.16. Pearson's correlation values between social-based user predictors and MAP@10 for different recommenders, on the CAMRa social dataset and using the AR methodology.

Predictor	Random	ItemPop	kNN	pLSA	Personal	PureSocial
Count (training)	0.004	-0.072	0.054	-0.096	0.012	0.067
Count (test)	0.031	-0.119	0.056	-0.163	-0.215	-0.294
Mean	-0.045	0.119	0.042	0.057	0.005	0.026
Standard deviation	0.041	-0.043	-0.139	-0.041	-0.063	-0.004
Avg neighbour degree	0.086	-0.168	0.044	-0.148	-0.120	-0.240
Betweenness centrality	-0.005	0.010	-0.008	-0.039	0.012	-0.075
Clustering coefficient	-0.002	-0.133	0.099	-0.149	-0.001	-0.135
Degree	0.045	-0.174	0.039	-0.198	-0.183	-0.185
Ego components size	0.030	-0.126	0.016	-0.143	-0.035	0.022
HITS	-0.009	-0.023	0.027	0.085	0.082	0.040
PageRank	0.000	-0.136	0.024	-0.134	-0.058	0.041
Two-hop neighbourhood	0.080	-0.148	0.016	-0.151	-0.146	-0.227
ItemSimple Clarity	0.002	-0.068	0.057	-0.092	0.022	0.081

Table A.17. Pearson's correlation values between social-based user predictors and MAP@10 for different recommenders, on the CAMRa collaborative dataset and using the AR methodology.

the predictors is now very different. We refer the reader to Section A.5.3 where we show how the dynamic ensembles perform when this metric is used.

A.5 Additional results about dynamic ensembles

Next we present additional results obtained in the experiments aimed to compare static and dynamic hybrid recommendations. We report values of metrics different to P@10, which has been extensively used in the thesis. In particular, we focus on MAP@10 in order to provide a full overview of the predictors' behaviour, since correlations with respect to this metric have been presented in the previous sections.

A.5.1 Performance results from dynamic ensembles on rating data

In this section we report experiments where dynamic hybrid recommenders are built by means of rating-based performance predictors. Table A.18 and Table A.19 show performance values (in terms of the MAP metric) of the hybrid recommenders by using the AR and 1R methodologies respectively. Additionally, Table A.20 shows performance values using item predictors along with the uuU1R methodology.

We can observe that the results from Table A.18 are quite similar to those presented in Table 7.2: in most cases the dynamic hybrid recommenders outperform the baseline static recommender, and the best result for each ensemble is obtained either by using the perfect correlation predictor or one of the clarity-based performance predictors. The only difference of these results with those of the P@10 metric (Table 7.2) is that when we evaluate with MAP@10 the baseline outperforms the dynamic ensembles for the combination HRU3. We have to note that the best static recommender is very different for this combination: whereas for P@10 the best result is obtained when $\lambda = 0.9$, for MAP@10 the best result is obtained for $\lambda = 0.5$, where, as we observed in Chapter 7, the rating-based user predictors seem to perform worse when the best static recommender is close to that value.

	HRU1	HRU2	HRU3	HRU4	HRU5	HRU6
R1 (λ=1.0)	0.0005	0.0298	0.0116	0.0116	0.0116	0.0116
R2 (λ=0.0)	0.0086	0.0086	0.0086	0.0001	0.1047	0.0551
Baseline (λ=0.5)	0.0043	0.0268	0.0271	0.0003	0.0794	0.0499
Best static (best λ)	0.0087 (0.1)	0.0310 (0.9)	0.0271 (0.5)	0.0022 (0.9)	0.1108 (0.1)	0.0584 (0.1)
Perfect correlation	0.0099	0.0373	0.0297	0.0119	0.1166	0.0663
PC-OM	0.0097	0.0399	0.0305	0.0045	0.1125	0.0602
PC-FW	0.0097	0.0296	0.0278	0.0008	0.1136	0.0615
Entropy-OM	**0.0057**▼▲	**0.0333**△▲	0.0260▽▽	**0.0009**▼▲	**0.0863**▼▲	**0.0509**▼△
ItemSimple-OM	**0.0090**△▲	**0.0330**△▲	0.0256▽▽	**0.0009**▼▲	**0.1149**▲▲	**0.0572**▼▲
ItemUser-OM	**0.0091**△▲	**0.0332**△▲	0.0255▽▽	**0.0008**▼▲	**0.1161**▲▲	**0.0576**▼▲
RatUser-OM	**0.0093**△▲	**0.0335**△▲	0.0262▽▽	**0.0009**▼▲	**0.1178**▲▲	**0.0575**▼▲
RatItem-OM	**0.0093**△▲	**0.0329**△▲	0.0259▽▽	**0.0008**▼▲	**0.1185**▲▲	**0.0576**▼▲
IRUser-OM	**0.0087**△▲	**0.0326**△▲	0.0257▽▽	**0.0008**▼▲	**0.1146**▲▲	**0.0580**▼▲
IRItem-OM	**0.0091**△▲	**0.0321**△▲	0.0250▽▼	**0.0008**▼▲	**0.1160**▲▲	**0.0577**▼▲
IRUserItem-OM	**0.0090**△▲	**0.0325**△▲	0.0256▽▽	**0.0008**▼▲	**0.1154**▲▲	**0.0578**▼▲
Entropy-FW	**0.0054**▼▲	**0.0282**▼▲	0.0264▽▼	**0.0006**▼▲	**0.0849**▼▲	**0.0508**▼△
ItemSimple-FW	**0.0085**▼▲	**0.0281**▼▲	0.0263▽▼	**0.0006**▼▲	**0.1111**▲▲	**0.0573**▼▲
ItemUser-FW	**0.0088**▲▲	**0.0282**▼▲	0.0262▽▼	**0.0006**▼▲	**0.1131**△▲	**0.0571**▼▲
RatUser-FW	**0.0090**△▲	**0.0281**▼▲	0.0265▼▽	**0.0006**▼▲	**0.1147**▲▲	**0.0569**▼▲
RatItem-FW	**0.0089**△▲	**0.0278**▼▲	0.0264▽▽	**0.0006**▼▲	**0.1154**▲▲	**0.0573**▼▲
IRUser-FW	**0.0085**▼▲	**0.0280**▼▲	0.0266▽▽	**0.0007**▼▲	**0.1094**▼▲	**0.0568**▼▲
IRItem-FW	**0.0089**▲▲	**0.0278**▼▲	0.0256▼▼	**0.0006**▼▲	**0.1123**△▲	**0.0570**▼▲
IRUserItem-FW	**0.0085**▼▲	**0.0281**▼▲	0.0265▽▽	**0.0007**▼▲	**0.1116**△▲	**0.0573**▼▲

Table A.18. Dynamic ensemble performance values (MAP@10) using the rating-based user predictors, on the MovieLens 1M and using the AR methodology.

Table A.19 shows performance values of the hybrid recommenders using the 1R methodology. The outcome of this experiment is identical to that presented in Table 7.3, except that now the best performing ensemble is a dynamic hybrid recommenders, either the perfect correlation or the PC-OM, instead of the best static ensemble, further validating our framework.

Additionally, in Table A.20 we show the performance of the dynamic hybrid recommenders using item predictors with the MAP metric. We may observe that these results are very similar to those presented for P@10 in Table 7.9, which emphasises the flexibility of our approach, in terms of being able to obtain performance improvements when using different evaluation metrics.

	HRU1	HRU2	HRU3	HRU4	HRU5	HRU6
R1 (λ=1.0)	0.0559	0.3335	0.1815	0.1815	0.1815	0.1815
R2 (λ=0.0)	0.0847	0.0847	0.0847	0.0125	0.5163	0.3430
Baseline (λ=0.5)	0.1147	0.2384	0.1974	0.0989	0.4004	0.3211
Best static (best λ)	0.1186 (0.3)	0.3369 (0.9)	0.2248 (0.8)	0.1789 (0.9)	0.5189 (0.1)	0.3618 (0.1)
Perfect correlation	0.1409	0.3625	0.2584	0.1919	0.5139	0.3822
PC-OM	0.1176	0.3014	0.2380	0.1375	0.5213	0.3785
PC-FW	0.1155	0.2678	0.2141	0.1189	0.5012	0.3715
Entropy-OM	0.1138▼▼	**0.3187**▼▲	**0.2103**▼▲	**0.1383**▼▲	0.3802▼▼	0.3087▼▼
ItemSimple-OM	0.1107▼▼	**0.3210**▼▲	**0.2111**▼▲	**0.1398**▼▲	**0.5022**▼▲	**0.3517**▼▲
ItemUser-OM	0.1084▼▼	**0.3174**▼▲	**0.2097**▼▲	**0.1379**▼▲	**0.5093**▼▲	**0.3534**▼▲
RatUser-OM	0.1051▼▼	**0.3216**▼▲	**0.2130**▼▲	**0.1405**▼▲	**0.5162**▼▲	**0.3556**▼▲
RatItem-OM	0.1046▼▼	**0.3187**▼▲	**0.2120**▼▲	**0.1400**▼▲	**0.5168**▼▲	**0.3560**▼▲
IRUser-OM	0.1109▼▼	**0.3131**▼▲	**0.2092**▼▲	**0.1382**▼▲	**0.4991**▼▲	**0.3495**▼▲
IRItem-OM	0.1073▼▼	**0.3062**▼▲	**0.2025**▼▲	**0.1328**▼▲	**0.5030**▼▲	**0.3484**▼▲
IRUserItem-OM	0.1082▼▼	**0.3127**▼▲	**0.2091**▼▲	**0.1381**▼▲	**0.5035**▼▲	**0.3497**▼▲
Entropy-FW	**0.1160**▼▲	**0.2703**▼▲	**0.2042**▼▲	**0.1184**▼▲	0.3965▼▼	0.3196▼▼
ItemSimple-FW	0.1142▼▼	**0.2712**▼▲	**0.2049**▼▲	**0.1192**▼▲	**0.4864**▼▲	**0.3491**▼▲
ItemUser-FW	0.1120▼▼	**0.2700**▼▲	**0.2037**▼▲	**0.1181**▼▲	**0.4951**▼▲	**0.3509**▼▲
RatUser-FW	0.1097▼▼	**0.2713**▼▲	**0.2058**▼▲	**0.1194**▼▲	**0.5049**▼▲	**0.3535**▼▲
RatItem-FW	0.1095▼▼	**0.2707**▼▲	**0.2046**▼▲	**0.1193**▼▲	**0.5066**▼▲	**0.3542**▼▲
IRUser-FW	0.1146▼▼	**0.2699**▼▲	**0.2044**▼▲	**0.1183**▼▲	**0.4828**▼▲	**0.3474**▼▲
IRItem-FW	0.1109▼▼	**0.2672**▼▲	**0.2010**▼▲	**0.1158**▼▲	**0.4905**▼▲	**0.3461**▼▲
IRUserItem-FW	0.1123▼▼	**0.2697**▼▲	**0.2043**▼▲	**0.1181**▼▲	**0.4894**▼▲	**0.3481**▼▲

Table A.19. Dynamic ensemble performance values (MAP@10) using the rating-based user predictors, on the MovieLens 1M dataset and using the 1R methodology.

	HRI1	HRI2	HRI3	HRI4
R1 (λ=1.0)	0.2498	0.2498	0.0835	0.0835
R2 (λ=0.0)	0.0717	0.0914	0.0717	0.0914
Baseline (λ=0.5)	0.1550	0.1746	0.0878	0.0932
Best static (best λ)	0.2146 (0.9)	0.2233 (0.9)	0.0892 (0.7)	0.0954 (0.2)
Entropy-OM	0.1283▼▼	0.1412▼▼	0.0872▼▼	**0.0979**▲▲
UserSimple-OM	**0.2116**▼▲	**0.2273**▲▲	**0.0879**▼▲	**0.0975**▲▲
UserItem-OM	**0.2129**▼▲	**0.2267**▲▲	0.0866▼▼	**0.0973**▲▲
RatItem-OM	**0.2166**▲▲	**0.2305**▲▲	0.0872▼▼	**0.0975**▲▲
RatUser-OM	**0.2231**▲▲	**0.2385**▲▲	0.0870▼▼	**0.0977**▲▲
URItem-OM	**0.2021**▼▲	**0.2136**▼▲	0.0869▼▼	**0.0973**▲▲
URUser-OM	**0.2176**▲▲	**0.2312**▲▲	0.0856▼▼	**0.0974**▲▲
URItemUser-OM	**0.2071**▼▲	**0.2200**▼▲	0.0870▼▼	**0.0973**▲▲
Entropy-FW	0.1382▼▼	0.1517▼▼	0.0873▼▼	**0.0981**▲▲
UserSimple-FW	**0.1770**▼▲	**0.1950**▼▲	**0.0900**▲▲	**0.0975**▲▲
UserItem-FW	**0.1771**▼▲	**0.1953**▼▲	**0.0902**▲▲	**0.0973**▲▲
RatItem-FW	**0.1776**▼▲	**0.1957**▼▲	**0.0902**▲▲	**0.0975**▲▲
RatUser-FW	**0.1793**▼▲	**0.1976**▼▲	**0.0904**▲▲	**0.0974**▲▲
URItem-FW	**0.1737**▼▲	**0.1907**▼▲	**0.0893**▲▲	**0.0969**▲▲
URUser-FW	**0.1782**▼▲	**0.1964**▼▲	**0.0903**▲▲	**0.0971**▲▲
URItemUser-FW	**0.1752**▼▲	**0.1931**▼▲	**0.0899**▲▲	**0.0974**▲▲

Table A.20. Dynamic ensemble performance values (MAP) using the rating-based item predictors, on the MovieLens 1M dataset and using the uuU1R methodology.

A.5.2 Performance results from dynamic ensembles on log data

In this section we compare the performance values of hybrid recommenders using the 1R methodology with log-based predictors, and P@10 and MAP@10 metrics. From Table A.21 and Table A.22 we can observe that the performance values are very similar to those shown in Table 7.11 and Table 7.12, respectively. This may be due to the fact that correlations for MAP@10 presented in Section A.4.2 were also analogous, since precision and MAP are almost equivalent under the 1R methodology as there is only one relevant item for each evaluated ranking. From Table A.21 we have to note, nonetheless, that the combination HL2 obtains worse performance values for the OM weighting strategy. Another difference is that the best performing ensemble for the combination HL2 now is achieved by the perfect correlation method, not by the best static recommender, as shown in Table 7.11 and Table 7.12. This shows that there would be room for improvement in the HL2 combination if we are able to define predictors with stronger correlation values. A similar situation is found for the combination HL3 in the five-fold split.

	HL1	HL2	HL3
R1 (λ=1.0)	0.4094	0.4094	0.7229
R2 (λ=0.0)	0.8255	0.5601	0.4094
Baseline (λ=0.5)	0.6922	0.6244	0.6429
Best static (best λ)	0.7913 (0.1)	0.6326 (0.3)	0.7256 (0.1)
Perfect correlation	0.7485	0.6470	0.6940
PC-OM	0.7272	0.6239	0.6763
PC-FW	0.7188	0.6240	0.6669
ItemSimple-OM	**0.7742**	0.6235	**0.7165**
Autocorrelation-OM	0.6592	0.5962	0.6163
TimeSimple-OM	**0.7676**	0.5913	**0.6955**
ItemTime-OM	**0.7762**	0.6200	**0.7149**
ItemPriorTime-OM	**0.7354**	0.6223	**0.6777**
ItemSimple-FW	**0.7597**	**0.6329**	**0.7106**
Autocorrelation-FW	0.6827	0.6177	0.6353
TimeSimple-FW	**0.7514**	0.6048	**0.6893**
ItemTime-FW	**0.7608**	**0.6292**	**0.7061**
ItemPriorTime-FW	**0.7276**	**0.6278**	**0.6714**

Table A.21. Dynamic ensemble performance values (MAP@10) using the log-based user predictors, on the Last.fm temporal split and using the 1R methodology.

	HL1	HL2	HL3
R1 (λ=1.0)	0.0453	0.0453	0.5824
R2 (λ=0.0)	0.5901	0.5387	0.0453
Baseline (λ=0.5)	0.4233	0.3961	0.3308
Best static (best λ)	0.5728 (0.1)	0.5317 (0.1)	0.5463 (0.1)
Perfect correlation	0.5820	0.5396	0.5728
PC-OM	0.5805	0.5403	0.5768
PC-FW	0.5795	0.5357	0.5611
ItemSimple-OM	**0.5395**	**0.4894**	**0.4397**
Autocorrelation-OM	0.3972	0.3659	**0.3642**
TimeSimple-OM	**0.5561**	**0.5206**	0.2405
ItemTime-OM	**0.5407**	**0.4881**	**0.4375**
ItemPriorTime-OM	**0.4577**	**0.4108**	**0.4276**
ItemSimple-FW	**0.5240**	**0.4791**	**0.3826**
Autocorrelation-FW	0.4191	0.3896	**0.3523**
TimeSimple-FW	**0.5372**	**0.5033**	0.2813
ItemTime-FW	**0.5243**	**0.4779**	**0.3819**
ItemPriorTime-FW	**0.4582**	**0.4201**	**0.3783**

Table A.22. Dynamic ensemble performance values (MAP@10) using the log-based user predictors, on the Last.fm five-fold split and using the 1R methodology.

A.5.3 Performance results from dynamic ensembles on social data

In this section we extend the results presented in Section 7.3.3, where P@10 and methodology AR were used in the two versions of CAMRa dataset. Here we use the same methodology, but the MAP@10 metric, as in the previous sections. In Table A.23 we can observe that there are some differences in the social version of the dataset when compared against the results shown in Table 7.14. We may attribute such differences to the different optimal static hybrid recommenders obtained since the correlations have not changed significantly. For instance, for HS1, the best lambda for static hybrids was 0.3 with P@10 and now it is 0.7 with MAP@10.

It is worth noting that even when the best static is so different from one metric to the other, the best performance is still obtained with the perfect correlation ensemble, which again reinforces the idea that finding predictors with higher correlations would be able to dynamically select the best weights in a user basis.

Finally, in Table A.24 we present the results regarding the collaborative version of the CAMRa dataset. In this case, we have a strong evidence towards the benefits of using performance predictors. First, we have to recall the absolute values of the pLSA correlations are stronger for the MAP metric than for precision. Then, now we

	HS1	HS2	HS3	HS4
R1 (λ=1.0)	0.2002	0.2002	0.2192	0.2192
R2 (λ=0.0)	0.1203	0.0364	0.1203	0.0364
Baseline (λ=0.5)	0.2175	0.2287	0.2555	0.2398
Best static (best λ)	0.2222 (0.3)	0.2349 (0.9)	0.2630 (0.3)	0.2408 (0.9)
Perfect correlation	0.2632	0.2562	0.2652	0.2511
PC-OM	0.2451	0.2576	0.2668	0.2497
PC-FW	0.2355	0.2378	0.2661	0.2446
AvgNeighDeg-OM	**0.2176**▽△	**0.2403**▽△	**0.2570**▽△	0.2229▽▼
BetCentrality-OM	0.2031▼▽	0.2232▽▽	0.2146▼▼	0.2236▽▼
ClustCoeff-OM	0.2082▼▽	0.2201▼▽	0.2261▼▼	0.2155▽▼
Degree-OM	0.2147▼▽	**0.2303**▼△	**0.2615**▽△	0.2166▽▼
EgoCompSize-OM	0.2078▼▽	0.2276▽▽	**0.2577**▽△	0.2214▽▼
HITS-OM	0.2127▼▽	**0.2428**▽△	0.2187▼▼	0.2231▽▼
PageRank-OM	0.2108▼▽	**0.2289**▽△	**0.2573**▽△	0.2238▽▼
TwoHopNeigh-OM	0.2121▼▽	**0.2377**▽△	0.2545▽▽	0.2202▽▼
AvgNeighDeg-FW	**0.2218**▽▲	**0.2370**▽△	**0.2574**▽△	0.2300▼▼
BetCentrality-FW	0.2171▼▽	**0.2351**▽△	0.2460▼▼	0.2344▼▼
ClustCoeff-FW	0.2129▼▽	**0.2348**▽△	0.2434▼▽	0.2344▼▼
Degree-FW	**0.2176**▽▲	**0.2373**▽▲	**0.2627**▽△	0.2332▽▽
EgoCompSize-FW	0.2124▼▽	**0.2373**▽▲	**0.2616**▽△	0.2352▽▼
HITS-FW	0.2164▼▽	**0.2380**▽▲	0.2405▼▼	0.2331▼▼
PageRank-FW	0.2169▼▽	**0.2363**▽△	**0.2574**▽△	0.2336▼▼
TwoHopNeigh-FW	**0.2177**▽▲	**0.2373**▽▲	**0.2587**▽△	0.2323▼▼

Table A.23. Dynamic ensemble performance values (MAP@10) using the log-based user predictors, on the CAMRa social dataset and using the AR methodology.

	HS1	HS2	HS3	HS4
R1 (λ=1.0)	0.1234	0.1234	0.1334	0.1334
R2 (λ=0.0)	0.1474	0.0195	0.1474	0.0195
Baseline (λ=0.5)	0.2190	0.1441	0.2359	0.1520
Best static (best λ)	0.2228 (0.3)	0.1494 (0.9)	0.2440 (0.2)	0.1520 (0.5)
Perfect correlation	0.2379	0.1597	0.2396	0.1590
PC-OM	0.1494	0.1577	0.1657	0.1535
PC-FW	0.1492	0.1462	0.1660	0.1498
AvgNeighDeg-OM	**0.2226**	**0.1511**	**0.2426**	0.1409
BetCentrality-OM	0.2069	0.1412	0.2119	0.1410
ClustCoeff-OM	0.2071	0.1390	0.2199	0.1386
Degree-OM	0.2175	0.1457	**0.2454**	0.1378
EgoCompSize-OM	0.2142	0.1441	**0.2417**	0.1395
HITS-OM	0.2100	**0.1519**	0.2136	0.1419
PageRank-OM	0.2110	**0.1447**	**0.2416**	0.1405
TwoHopNeigh-OM	0.2156	**0.1514**	**0.2469**	0.1392
AvgNeighDeg-FW	**0.2214**	**0.1501**	**0.2410**	0.1464
BetCentrality-FW	0.2145	**0.1479**	0.2254	0.1475
ClustCoeff-FW	0.2167	**0.1474**	0.2266	0.1474
Degree-FW	**0.2197**	**0.1497**	**0.2437**	0.1464
EgoCompSize-FW	0.2189	**0.1490**	**0.2405**	0.1474
HITS-FW	0.2157	**0.1502**	0.2241	0.1461
PageRank-FW	0.2170	**0.1476**	**0.2440**	0.1465
TwoHopNeigh-FW	**0.2216**	**0.1502**	**0.2423**	0.1469

Table A.24. Dynamic ensemble performance values (MAP@10) using the log-based user predictors, on the CAMRa collaborative dataset and using the AR methodology.

have to note that the performance values are now better fo MAP than for precision, in particular for HS1 and HS3, where dynamic hybrid recommenders outperform the baselines in a larger number of cases.

Appendix B

Introducción

En este capítulo presentamos una visión general de la tesis. En las Secciones B.1 y B.2 mostramos las motivaciones y objetivos de nuestro trabajo. En la Sección B.3 resumimos las principales contribuciones de la tesis, y en la Sección B.4 enumeramos las publicaciones resultantes de nuestra investigación. Finalmente, en la Sección B.5 describimos la estructura de este documento.

B.1 Motivación

Las tecnologías de Recuperación de Información (RI) han ganado una prevalencia excepcional en las dos últimas décadas con la explosión del número de repositorios masivos de información existentes en línea, y más en particular en la *World Wide Web*. En RI se han investigado y diseñado formas que buscan maximizar el grado de satisfacción de ciertas condiciones objetivas, típicamente – aunque no necesariamente de manera única – la satisfacción del usuario. La investigación y el desarrollo en RI han girado en torno a la definición de modelos y algoritmos que mejor alcancen dicho objetivo, de metodologías y métricas que permiten evaluar cuánto de bien se consigue esta meta con diferentes sistemas, así como de teorías consolidadas que proveen una base sólida y una orientación en el desarrollo de algoritmos de RI y su consistente evaluación. Entre las muchas tendencias que han surgido desde el flujo principal de investigación y desarrollo, un nuevo reto de investigación ha empezado a ser considerado desde comienzos de los 2000: ¿es posible predecir cómo de bueno será un resultado devuelto por un sistema de RI antes de presentarlo al usuario, o incluso, antes de efectuar por completo la búsqueda del resultado por el sistema (Cronen-Townsend et al., 2002)? Esta pregunta ha dado pie a una fértil corriente de investigación en lo que se ha llamado en RI como **predicción de eficacia**.

La predicción de eficacia tiene muchos usos potenciales en RI. Desde la perspectiva del usuario proporcionaría información que puede ser usada para dirigir una búsqueda, desde la perspectiva del sistema ayudaría a distinguir consultas poco eficaces, y desde la perspectiva del administrador del sistema permitiría identificar consultas relacionadas sobre un tema específico que resultan difíciles para el motor de búsqueda. Las técnicas de predicción de eficacia se basan en el análisis y caracterización de la evidenca usada por un sistema de RI para evaluar la relevancia (utilidad, valor, etc.) de los ítems a recuperar (documentos, artículos, etc.) en tiempo de ejecución (Cronen-Townsend et al., 2002). El escenario más común en recuperación de información supone una consulta del usuario y una colección de documentos como la entrada básica para formar una lista ordenada de resultados de búsqueda, pero otros elementos adicionales pueden tenerse en cuenta para seleccionar y ordenar resultados (Baeza-Yates and Ribeiro-Neto, 2011). Cualquier información que el sistema de recuperación de información tome como entrada puede ser utilizada también para la predicción de su eficacia, y habitualmente los métodos de predicción usan información adicional. El contexto del usuario (la tarea actual, los registros de búsquedas, las preferencias, etc.), las propiedades globales de los documentos de la colección, comparaciones con respecto a (otros) elementos de referencia como datos históricos, o la salida de distintos sistemas, son algunos ejemplos de las diferentes fuentes de información de las que un predictor puede extraer evidencia.

Predecir la eficacia de un subsistema, módulo, función o entrada contrastando la estimación de eficacia de cada componente para una consulta, permite una serie de estrategias de optimización dinámicas que seleccionen en tiempo de ejecución la opción que se predice funcionará mejor o, cuando se usan sistemas a gran escala o aproximaciones híbridas, que ajusten sobre la marcha el grado de participación de cada módulo. En el campo de RI dominan los casos donde información de relevancia, sistemas de recuperación, modelos y criterios se definen en función de la fusión o combinación de sub-modelos. Sistemas personalizados de recuperación (incluyendo técnicas como búsqueda personalizada, sistemas de recomendación, filtrado colaborativo y búsqueda contextualizada) son claros ejemplos donde se puede aplicar la predicción de eficacia dado que dichos sistemas combinan varias fuentes de evidencia para la estimación de la relevancia, como pueden ser consultas explícitas, historial de búsqueda, puntuaciones de usuarios, información social, información del usuario y modelos de contexto.

La predicción de eficacia encuentra una motivación adicional en la recomendación personalizada, puesto que esas aplicaciones pueden decidir si producir recomendaciones u ocultarlas, entregando sólo las suficientemente fiables. Más aún, los Sistemas de Recomendación (SR) actuales se caracterizan por una creciente diversificación de los tipos y fuentes de datos, contenidos, evidencias y métodos disponibles para tomar decisiones y construir los resultados. En este contexto, predecir la eficacia de un método de recomendación específico o de una componente se convierte en un problema atractivo, ya que permite una combinación adecuada de las alternativas disponibles y sacar el máximo provecho de ellas, adaptando dinámicamente la estrategia de recomendación a la situación actual. La cuestión gana mayor relevancia hoy con la proliferación de técnicas de recomendación híbridas para mejorar la precisión de los métodos – siendo el premio Netflix uno de los ejemplos paradigmáticos del uso de estos métodos, donde los participantes mejor situados combinaron múltiples métodos de recomendación. Esto requiere de una investigación en aproximaciones híbridas con un nivel de mecanismos dinámicos auto-ajustables, de manera que se optimice la efectividad resultante de los sistemas de recomendación, tomando oportuna ventaja de datos de alta calidad cuando estén disponibles, pero evitando aferrarse a estrategias fijas cuando se predice que pueden producir resultados pobres bajo ciertas condiciones.

La predicción de eficacia en RI típicamente se evalúa en términos de correlación entre el predictor y los valores de eficacia para cada consulta. Esto requiere métricas de evaluación de la eficacia fiables, las cuales han sido analizadas cuidadosamente y están actualmente bien establecidas en el área de RI, orientadas en su mayor parte a búsqueda ad-hoc. Por el contrario, la evaluación en el campo de los SR está más abierta, y la variabilidad en las técnicas de evaluación y configuraciones experimentales es significativa. Cómo medir la eficacia de un sistema de recomendación es un

asunto clave en nuestra investigación ya que las medidas de calidad del sistema pueden verse influidas por propiedades estadísticas del método de medición y/o por el diseño experimental. A lo largo de esta tesis nos centraremos en la precisión del sistema, donde hemos de evitar que si una métrica estuviera sesgada hacia algún tipo de ruido medido a la vez que la calidad de la recomendación, pues entonces un predictor que sólo capturara dicho ruido podría actuar como un predictor de eficacia erróneamente útil. Así, los sesgos estadísticos (ruidos) de las metodologías de evaluación deben ser bien entendidos para permitir una valoración significativa de los predictores de eficacia.

Tomando el estado del arte de predicción de eficacia en RI como punto de partida, el trabajo actual replantea este problema en el campo de los Sistemas de Recomendación donde ha sido escasamente considerado hasta la fecha. Investigamos definiciones adecuadas de eficacia en el contexto de los SR y los elementos a los que sensatamente se puede aplicar, analizando los sesgos estadísticos que pueden aparecer cuando se adapta el marco de evaluación de RI a SR. De este modo tomamos como dirección principal la aplicación de los predictores de eficacia para obtener mejoras en dos problemas específicos de combinación en el campo de SR, a saber, la combinación dinámica de métodos de recomendación en sistemas de recomendación híbridos, y la agregación dinámica de las señales de vecinos en filtrado colaborativo basado en usuario.

B.2 Objetivos

El principal objetivo de la investigación presentada aquí es encontrar métodos predictivos para la eficacia de componentes específicos de sistemas de recomendación, y mejorar la eficacia de los métodos de recomendación combinados, basados en el análisis y predicción dinámicos y automáticos de la eficacia esperada de los elementos de un método compuesto, con los cuales la participación relativa de cada elemento se ajusta de acuerdo a su efectividad predicha. Para abordar estos problemas nuestro trabajo tiene los siguientes objetivos de investigación concretos:

O1: Análisis y formalización de cómo se define y evalúa la eficacia en los sistemas de recomendación. Dado que pretendemos predecir su eficacia necesitamos desarrollar un estudio en profundidad sobre cómo se pueden evaluar de manera fiable los sistemas de recomendación en términos de valores númericos de una métrica. Más aún, hemos de investigar si existen sesgos en la manera en que los sistemas se evalúan – debidos tanto a metodologías de evaluación como a métricas, ya que cualquier sesgo en el proceso de evaluación podría conducir a resultados inconcluyentes o engañosos con respecto el poder predictivo de los métodos de predicción de eficacia propuestos. Si dichos sesgos existieran, intentaríamos entenderlos de manera precisa, y desarrollar metodologías que los aislaran. Además, deberíamos com-

probar la efectividad de nuestros predictores frente a la de otros métodos conocidos más básicos y observar si cambia al aislar dichos sesgos.

O2: Adaptación y definición de técnicas de predicción de eficacia para sistemas de recomendación. Queremos estudiar el potencial de la predicción de eficacia en problemas y escenarios específicos en el área de los Sistemas de Recomendación. Investigaremos la definición de un marco formal donde los predictores de eficacia puedan ser integrados. Como punto de partida, pretendemos explorar la adaptación de predictores específicamente efectivos en Recuperación de Información como la claridad de la consulta (Cronen-Townsend et al., 2002) al campo de la recomendación. De manera complementaria a la adaptación de técnicas conocidas, queremos investigar la definición de nuevos predictores basados en modelos de la Teoría de Información y Grafos Sociales, además de otras aproximaciones heurísticas y específicas del dominio. Una vez hayamos definido algunos predictores de eficacia para recomendación, evaluaríamos la efectividad de dichos predictores en términos de su correlación con métricas de eficacia de manera que pudiéramos estimar su poder de predicción.

O3: Aplicación de predictores de eficacia a sistemas de recomendación compuestos e híbridos. Queremos identificar e integrar los predictores propuestos en métodos de recomendación combinados para obtener una mejora real en la eficacia de los métodos combinados. Con este objetivo en mente consideraremos problemas donde la agregación de métodos de recomendación es necesaria, y analizaremos cómo aplicar los predictores de eficacia mencionados antes a tales problemas. Además, deberíamos realizar un estudio metodológico para la aproximación experimental, su configuración y las métricas usadas, de manera que se usen métodos base y diseños experimentales apropiados. Finalmente, evaluaremos las mejoras y beneficios de los métodos combinados cuando se utilizan los predictores de eficacia.

B.3 Contribuciones

Esta tesis se dedica al problema de estimar la eficacia de los sistemas de recomendación para usuarios e ítems particulares. Las contribuciones particulares de esta tesis están relacionados con la evaluación de la eficacia de un sistema de recomendación y la predicción de la misma, donde hemos abordado varios problemas relacionados con ambos temas y hemos propuesto modelos y métodos novedosos, que han sido empleados en dos aplicaciones como mostraremos a continuación.

Como un primer paso, esta tesis analiza el paradigma Cranfield de evaluación de Recuperación de Información, dado que los sistemas de recomendación normalmente se consideran como un problema particular del filtrado de información, y, por tanto, de la recuperación de información en general (Belkin and Croft, 1992). En el Capítulo 4 **argumentamos las diferencias involucradas en las alternativas de**

diseños experimentales a partir de las hipótesis habituales hechas en el paradigma Cranfield, lo cual resulta en **la aparición de sesgos estadísticos considerables en Sistemas de Recomendación**, para los que **proponemos diferentes métodos para neutralizar estos sesgos**. De manera adicional, las siguientes contribuciones relacionadas han sido realizadas:

- Proponemos una caracterización precisa y sistemática de las alternativas para la adaptación del paradigma Cranfield a tareas de recomendación. Identificamos hipótesis y condiciones subyacentes en dicho paradigma Cranfield que no pueden ser asumidas en los experimentos habituales de recomendación.
- Detectamos y caracterizamos los sesgos estadísticos resultantes, a saber, la dispersión de test y la popularidad de los ítems, los cuales no aparecen en colecciones de test habituales en RI, pero que interfieren en experimentos de recomendación.
- Proponemos dos diseños experimentales nuevos para neutralizar estos sesgos. Observamos que una evaluación basada en percentiles reduce considerablemente el margen para el sesgo de popularidad, mientras que una aproximación basada en un test uniforme elimina cualquier ventaja estadística obtenida por tener más puntuaciones positivas de test. Más aún, encontramos que las dos propuestas discriminan bien entre recomendaciones puramente basadas en popularidad y un algoritmo de recomendación personalizado.

Además, en esta tesis **mostramos cómo las técnicas de predicción de eficacia de consultas desarrolladas en Recuperación de Información pueden ser adaptadas a los Sistemas de Recomendación, y resultar en predictores eficaces en este dominio**. Presentamos estos predictores de eficacia en el Capítulo 6, donde proponemos distintas adaptaciones del predictor de claridad de consulta basadas en distintas interpretaciones de los modelos de lenguaje subyacentes, así como con modelos de Teoría de Información y de Grafos Sociales. Más aún, en el mismo capítulo **evaluamos la efectividad de dichos predictores midiendo la correlación con respecto a métricas de eficacia**, donde también probamos los métodos propuestos en el Capítulo 4 para neutralizar los sesgos en la evaluación. A continuación resumimos las contribuciones específicas respecto a la predicción de eficacia para recomendación:

- Definimos y elaboramos varios modelos predictivos en el dominio de los sistemas de recomendación de acuerdo a diferentes formulaciones e hipótesis, y basados en tres tipos de datos de preferencia: puntuaciones, registros y sociales.
- Las formulaciones para preferencias en base a puntuaciones se basan en adaptaciones de la claridad de consulta de RI y conceptos de Teoría de la Información como la entropía. En esta adaptación proponemos distintas estimaciones

de probabilidad, donde desarrollamos derivaciones Bayesianas y estimaciones no parámetricas.

- También explotamos atributos temporales al definir los predictores basados en registros. Más específicamente, usamos una versión sensible al tiempo de la divergencia de Kullback-Leibler, junto con otros conceptos de series temporales como la autocorrelación del usuario.
- Usamos métricas basadas en Teoría de Grafos para definir predictores que aprovechan las estructuras de red social y las correlaciones entre las propiedades topológicas de los usuarios y el éxito de las recomendaciones devueltas.
- Encontramos fuertes correlaciones entre las salidas de los predictores y las métricas de eficacia, mostrando por tanto evidencia empírica del poder predictivo de las técnicas propuestas. Más aún, cuando se utilizan las metodologías no sesgadas los predictores conservan buenos valores de correlación, evidenciando que los predictores propuestos no están sólo capturando los sesgos analizados y beneficiándose de los mismos, especialmente cuando los comparamos frente a otros predictores triviales.

Finalmente, los Capítulos 7 y 8 presentan dos aplicaciones de los predictores de eficacia en Sistemas de Recomendación. En el Capítulo 7 **proponemos varios sistemas híbridos ponderados linealmente donde las ponderaciones se ajustan dinámicamente de acuerdo a la salida de los predictores**. Observamos que las correlaciones obtenidas en el Capítulo 6 ayudan a decidir cuáles son las mejores combinaciones a experimentar. Más importante aún, **la correlación entre el predictor y el algoritmo de recomendación tiende a anticipar bien cuándo un sistema híbrido mejorará con respecto a un método base**. Además, el Capítulo 8 **presenta un marco unificado donde los predictores de eficacia se usan para seleccionar y ponderar los vecinos cercanos en un algoritmo estándar de filtrado colaborativo basado en usuario**. La metodología tradicional de predicción de eficacia es adaptada y traducida a este problema, donde definimos novedosas métricas sobre la eficacia de vecinos y evaluamos el poder predictivo de los predictores.

Las contribuciones relacionadas con la parte de aplicaciones de la tesis son, en resumen, las siguientes:

- Proponemos un marco de hibridación dinámica para decidir automáticamente cuándo y cómo debería realizarse la hibridación, dependiendo de distintas condiciones, a saber: las correlaciones entre los algoritmos de recomendación y los predictores, y la eficacia relativa entre los algoritmos combinados.
- En varios experimentos con los predictores de eficacia mencionados previamente, nuestros resultados indican que una fuerte correlación con la eficacia

tiende a corresponder con mejoras en la recomendación híbrida dinámica cuando los predictores se usan para ajustar los pesos de la combinación.

- Proponemos un marco teórico para la selección y ponderación de vecinos en sistemas de recomendación basados en usuarios. Este marco se fundamenta en la predicción de eficacia estableciendo equivalencias entre la tarea de predicción de puntuaciones basada en vecindarios y una agregación dinámica de componentes, en este caso, las predicciones de cada vecino.
- Comparamos varias métricas de confianza del estado del arte así como otras técnicas para valorar vecinos, interpretadas ambas como predictores de eficacia de vecinos. También proponemos varias métricas de eficacia de vecinos que capturan diferentes nociones de la calidad de los vecinos.

B.4 Publicaciones relacionadas con la tesis

En los siguientes artículos de revistas y conferencias internacionales presentamos descripciones, resultados y conclusiones relacionadas con esta tesis:

Predicción de eficacia y evaluación

1. Bellogín, A., Cantador, I., Díez, F., Castells, P., and Chavarriaga, E. (2012). An empirical comparison of social, collaborative filtering, and hybrid recommenders. *ACM Transactions on Intelligent Systems and Technology*, por aparecer.
2. Bellogín, A., Castells, P., and Cantador, I. (2011). Predicting the Performance of Recommender Systems: An Information Theoretic Approach. In Amati, G. and Crestani, F., editors, *ICTIR*, volume 6931 of *Lecture Notes in Computer Science*, pages 27–39, Berlin, Heidelberg. Springer Berlin / Heidelberg.
3. Bellogín, A., Castells, P., and Cantador, I. (2011). Self-adjusting hybrid recommenders based on social network analysis. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information*, SIGIR '11, pages 1147–1148, New York, NY, USA. ACM.
4. Bellogín, A., Castells, P., and Cantador, I. (2011). Precision-oriented evaluation of recommender systems: an algorithmic comparison. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 333–336, New York, NY, USA. ACM.
5. Bellogín, A. and Castells, P. (2010). A Performance Prediction Approach to Enhance Collaborative Filtering Performance. In Gurrin, C., He, Y., Kazai, G., Kruschwitz, U., Little, S., Roelleke, T., Rüger, S., and Rijsbergen, editors,

Advances in Information Retrieval, volume 5993 of *Lecture Notes in Computer Science*, pages 382–393, Berlin, Heidelberg. Springer Berlin / Heidelberg.

6. Bellogín, A. and Castells, P. (2009). Predicting neighbor goodness in collaborative filtering. In And, T. A., Yager and, R. R., And, H. B., And, H. C., and Larsen, H. L., editors, *FQAS*, volume 5822 of *Lecture Notes in Computer Science*, pages 605–616, Berlin, Heidelberg. Springer Berlin / Heidelberg.

Recomendación basada en contenido

7. Cantador, I., Bellogín, A., and Vallet, D. (2010). Content-based recommendation in social tagging systems. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 237–240, New York, NY, USA. ACM.
8. Cantador, I., Bellogín, A., and Castells, P. (2008). News@hand: A Semantic Web Approach to Recommending News. In Nejdl, W., Kay, J., Pu, P., and Herder, E., editors, *Adaptive Hypermedia and Adaptive Web-Based Systems*, volume 5149 of *Lecture Notes in Computer Science*, chapter 34, pages 279–283. Springer Berlin / Heidelberg, Berlin, Heidelberg.
9. Cantador, I., Bellogín, A., and Castells, P. (2009). Ontology-Based Personalised and Context-Aware Recommendations of News Items. In *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT '08. IEEE/WIC/ACM International Conference on*, volume 1, pages 562–565.
10. Cantador, I., Bellogín, A., Fernández-Tobías, I., and López-Hernández, S. (2011a). Semantic Contextualisation of Social Tag-Based Profiles and Item Recommendations. In Huemer, C., Setzer, T., Aalst, W., Mylopoulos, J., Sadeh, N. M., Shaw, M. J., Szyperski, C., Aalst, W., Mylopoulos, J., Sadeh, N. M., Shaw, M. J., and Szyperski, C., editors, *Electronic Commerce and Web Technologies*, volume 85 of *Lecture Notes in Business Information Processing*, chapter 9, pages 101–113. Springer Berlin Heidelberg, Berlin, Heidelberg.
11. Fernández-Tobías, I., Cantador, I., and Bellogín, A. (2011). cTag: Semantic contextualisation of social tags. In *Proceedings of the Workshop on Semantic Adaptive Social Web (SASWeb 2011). CEUR Workshop Proceedings, vol. 730*, pages 45–54. RWTH, Aachen (2011).

Recomendación basada en filtrado colaborativo

12. Bellogín, A., Wang, J., and Castells, P. Bridging Memory-Based Collaborative Filtering and Text Retrieval. Information Retrieval Journal, por aparecer.
13. Bellogín, A., Cantador, I., and Castells, P. A Comparative Study of Heterogeneous Item Recommendations in Social Systems. *Information Sciences*, por aparecer.

14. Bellogín, A. and Parapar, J. (2012). Using Graph Partitioning Techniques for Neighbour Selection in User-Based Collaborative Filtering. In *Proceedings of the sixth ACM conference on Recommender systems*, RecSys '12, pages 213–216, New York, NY, USA. ACM. (premio al mejor artículo corto)
15. Bellogín, A., Wang, J., and Castells, P. (2011). Structured collaborative filtering. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 2257–2260, New York, NY, USA. ACM.
16. Bellogín, A., Wang, J., and Castells, P. (2011). Text Retrieval Methods for Item Ranking in Collaborative Filtering. In Clough, P., Foley, C., Gurrin, C., Jones, G., Kraaij, W., Lee, H., and Mudoch, V., editors, *Advances in Information Retrieval*, volume 6611 of *Lecture Notes in Computer Science*, chapter 30, pages 301–306. Springer Berlin / Heidelberg, Berlin, Heidelberg.
17. Bellogín, A., Cantador, I., and Castells, P. (2010). A study of heterogeneity in recommendations for a social music service. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '10, pages 1–8, New York, NY, USA. ACM.

Recomendación basada en filtrado social

18. Díez, F., Chavarriaga, J. E., Campos, P. G., and Bellogín, A. (2010). Movie recommendations based in explicit and implicit features extracted from the Filmtipset dataset. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, CAMRa '10, pages 45–52, New York, NY, USA. ACM.

Recomendación sensible al tiempo

19. Campos, P. G., Bellogín, A., Díez, F., and Cantador, I. (2012). Time Feature Selection for Identifying Active Household Members. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, New York, NY, USA. ACM (por aparecer).
20. Campos, P. G., Díez, F., and Bellogín, A. (2011). Temporal rating habits: a valuable tool for rating discrimination. In *Proceedings of the 2nd Challenge on Context-Aware Movie Recommendation*, CAMRa '11, pages 29–35, New York, NY, USA. ACM.
21. Campos, P. G., Bellogín, A., Díez, F., and Chavarriaga, J. E. (2010). Simple time-biased KNN-based recommendations. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, CAMRa '10, pages 20–23, New York, NY, USA. ACM.

Sístemas de recomendación híbridos

22. Cantador, I., Castells, P., and Bellogín, A. (2011). An enhanced semantic layer for hybrid recommender systems. *International Journal on Semantic Web and Information Systems*, 7(1):44–78.
23. Cantador, I., Bellogín, A., and Castells, P. (2008). A multilayer ontology-based hybrid recommendation model. *AI Commun.*, 21(2-3):203–210.
24. Cantador, I., Castells, P., and Bellogín, A. (2007). Modelling Ontology-based Multilayered Communities of Interest for Hybrid Recommendations. In *Workshop on Adaptation and Personalisation in Social Systems: Groups, Teams, Communities, at the 11th International Conference on User Modeling*.

Evaluación de la recomendación

25. Bellogín, A., Cantador, I., Castells, P., and Ortigosa, A. (2011). Discerning Relevant Model Features in a Content-based Collaborative Recommender System. In Fürnkranz, J. and Hüllermeier, E., editors, *Preference Learning*, chapter 20, pages 429–455. Springer Berlin Heidelberg, Berlin, Heidelberg.
26. Bellogín, A., Cantador, I., Castells, P., and Ortigosa, A. (2008). Discovering Relevant Preferences in a Personalised Recommender System using Machine Learning Techniques. In *Preference Learning Workshop (PL 2008), at the 8th European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2008)*, pages 82–96.

Estas publicaciones se relacionan con los contenidos de esta tesis como sigue. En [4] analizamos diferentes metodologías de evaluación existentes en la literatura de recomendación (Capítulos 3 y 4). En [2], [5], [6] definimos las formulaciones para el concepto de claridad de usuario basada en puntuaciones (Capítulo 6), mientras que en [1] y [3] definimos los predictores sociales (también en el Capítulo 6). Además, en [1] y [3] también investigamos el uso de los predictores de eficacia para recomendaciones híbridas dinámicas (Capítulo 7). Más aún, en [5] y [6] abordamos el problema de la ponderación de vecinos basada en los predictores de eficacia de vecinos (Capítulo 8).

Además, durante el transcurso de la tesis, la investigación presentada aquí motivó una serie de publicaciones que abordaban temas más amplios en el área, como las recomendaciones basadas en contenido [7-11], el filtrado colaborativo [12-17], las técnicas de filtrado social [18], la recomendación sensible al tiempo [19-21], los sistemas de recomendación híbrida [22-24], y la evaluación de la recomendación [25, 26]. Estas publicaciones han resultado en el uso y construcción de conjuntos de datos, el desarrollo de algoritmos y la investigación y utilización de algunas metodologías y métricas de evaluación que aparecen en esta tesis.

Publicaciones adicionales

Trabajo preliminar relacionado con las propuestas presentadas aquí fue incluido en primer lugar en la tesis de Máster titulada "Performance prediction in recommender Systems: Application to the dynamic optimisation of aggregative methods" (Bellogín, 2009). Específicamente, en dicho trabajo proponemos el concepto de predicción de eficacia para recomendación. Además, la motivación, el potencial impacto y los principales resultados de nuestra investigación fueron publicados como contribuciones en dos simposios doctorales internacionales:

- Bellogín, A. (2011). Predicting performance in recommender systems. Doctoral Symposium. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 371–374, New York, NY, USA. ACM.
- Bellogín, A. (2011). Performance Prediction in Recommender Systems. Doctoral Symposium. In Konstan, J., Conejo, R., Marzo, J., and Oliver, N., editors, *User Modeling, Adaption and Personalization*, volume 6787 of *Lecture Notes in Computer Science*, pages 401–404, Berlin, Heidelberg. Springer Berlin / Heidelberg.

Además, los siguientes artículos están bajo revisión, algunos de ellos altamente relacionados con los temas de la tesis:

- Bellogín, A., Castells, P., and Cantador, I. Statistical Biases in IR Metrics for Recommender Systems: A Methodological Framework for the Adaptation of the Cranfield Paradigm. En revisión.
- Bellogín, A., Castells, P., and Cantador, I. Neighbour Selection and Weighting in User-Based Recommender Systems: A Performance Prediction Approach. En revisión.
- Parapar, J., Bellogín, A., Castells, P., and Barreiro, Á. Relevance-Based Language Modelling for Recommender Systems. En revisión.

B.5 Estructura de la tesis

La tesis está dividida en seis partes. La primera parte presenta y motiva el problema abordado, así como una revisión del estado del arte en el campo de los Sistemas de Recomendación donde esta tesis se enmarca. La segunda parte describe las diferentes técnicas de evaluación usadas en la literatura de sistemas de recomendación, y provee un análisis de las alternativas de diseño y los sesgos estadísticos que pueden aparecer. La tercera parte hace una revisión de la literatura en predicción de eficacia, propone adaptaciones de este concepto al espacio de los sistemas de recomendación, y evalúa el poder predictivo de estas propuestas. La cuarta parte muestra dos aplicaciones de los predictores de eficacia propuestos. La quinta parte concluye y resume las princi-

pales contribuciones de esta tesis. Información adicional y otros detalles se incluyen en la última parte.

Más detalladamente, los contenidos de esta tesis se distribuyen como sigue:

Parte I. Introducción

- El **Capítulo 1** presenta la motivación, objetivos, contribuciones y publicaciones relacionadas con la tesis.
- El **Capítulo 2** presenta una visión general del estado del arte en sistemas de recomendación, teniendo en cuenta una clasificación de los principales tipos de técnicas. También describe los puntos débiles de las distintas técnicas de recomendación y presenta una amplia gama de métodos híbridos de recomendación que ayudan a superar esas limitaciones.

Parte II. Evaluando la Eficacia en los Sistemas de Recomendación

- El **Capítulo 3** describe las principales métricas y metodologías de evaluación usadas en el campo de los sistemas de recomendación. También describe los conjuntos de datos públicos más habitualmente usados.
- El **Capítulo 4** presenta un análisis y formalización de las diferentes metodologías de evaluación descritas en la literatura. Primero, presenta una caracterización sistemática de las alternativas de diseños experimentales. Después identifica y analiza sesgos estadísticos que aparecen cuando algunas metodologías se aplican a recomendación, y propone dos diseños experimentales alternativos que neutralizan tales sesgos satisfactoriamente.

Parte III. Prediciendo la Eficacia en los Sistemas de Recomendación

- El **Capítulo 5** presenta el problema de predicción de eficacia en Recuperación de Información, incluye un resumen de los principales trabajos en dicha área, tanto en la definición de predictores de eficacia (aplicados a consultas) como en la evaluación de los predictores para estimar su poder predictivo.
- El **Capítulo 6** formula el problema de predicción de eficacia en los Sistemas de Recomendación. Define varios predictores de eficacia basados en tres dimensiones de la recomendación donde analizamos de manera cualitativa el poder predictivo de los predictores.

Parte IV. Aplicaciones

- El **Capítulo 7** propone un marco donde los predictores de eficacia se usan para construir sistemas de recomendación híbridos dinámicos. Evalua estos algoritmos de recomendación en los tres espacios de entrada previamente considerados en la definición de los predictores de eficacia, y usa las distintas alternativas de diseño experimental donde algunos sesgos estadísticos han sido neutralizados.

- El **Capítulo 8** reformula el problema de recomendación basado en usuarios, dando una generalización del mismo como un problema de predicción de eficacia. Investiga cómo adoptar dicha generalización para definir un marco unificado donde podamos realizar un análisis objetivo de la efectividad (poder predictivo) de las funciones de valoración de los vecinos.

Parte V. Conclusiones

- El **Capítulo 9** concluye con un resumen de las principales contribuciones de esta tesis y una discusión sobre las líneas de investigación futuras.

Parte VI. Apéndices

- El **Apéndice A** da detalles sobre los métodos propuestos en esta tesis: configuración de los algoritmos de recomendación y los parámetros de los diseños experimentales usados en la evaluación. También reporta estadísticas detalladas sobre los conjuntos de datos utilizados en los experimentos, complementando otros datos incluidos en capítulos previos.
- El **Apéndice B** contiene la traducción a español del Capítulo 1.
- El **Apéndice C** contiene la traducción a español del Capítulo 9.

Appendix C

Conclusiones y trabajo futuro

En esta tesis hemos investigado cómo medir y predecir la eficacia de sistemas de recomendación. Hemos analizado y propuesto un conjunto de métodos basados en la adaptación de predictores de eficacia desde el área de Recuperación de Información – principalmente el predictor de claridad de consulta, que captura la ambigüedad de una consulta con respecto a una colección de documentos dada. Hemos definido varios modelos de lenguaje utilizando distintos espacios de probabilidad para capturar los aspectos de los usuarios e ítems implicados en las tareas de recomendación. En este contexto, hemos propuesto y evaluado técnicas novedosas para distintos espacios de entrada extraídas de la Teoría de la Información y la Teoría de Grafos Sociales, usando propiedades sobre las preferencias de usuario así como métricas de grafos, como PageRank sobre la red social del usuario.

Más aún, dado que queremos predecir la eficacia de un sistema de recomendación particular, necesitamos una metodología de evaluación clara con la cual las predicciones de eficacia puedan ser contrastadas. Así, en esta tesis investigamos la metodología de evaluación como parte del problema abordado, donde hemos identificado sesgos estadísticos en la evaluación de la recomendación – a saber, sesgos de dispersión en test y popularidad – los cuales pueden distorsionar las medidas de eficacia, y por tanto, confundir el poder aparente de los métodos de predicción de eficacia. Hemos analizado en profundidad el efecto de dichos sesgos, y hemos propuesto dos diseños experimentales capaces de neutralizar el sesgo de popularidad: una técnica basada en percentiles y un test uniforme. El análisis sistemático de las metodologías de evaluación y las nuevas variantes propuestas permiten una valoración más completa y precisa de la eficiencia de nuestros métodos de predicción de eficacia.

Por otro lado, hemos explotado los métodos propuestos de predicción de eficacia en dos aplicaciones donde se usan para ponderar dinámicamente distintos componentes de un sistema de recomendación, a saber, el ajuste dinámico de recomendaciones híbridas ponderadas, y la ponderación dinámica de las preferencias de los vecinos en filtrado colaborativo basado en usuario. A través de una serie de experimentos empíricos con varios conjuntos de datos y diseños experimentales, hemos encontrado una correspondencia entre el poder predictivo de nuestros predictores de eficacia y la mejora en eficacia de las dos aplicaciones evaluadas.

Presentamos aquí las conclusiones principales obtenidas en este trabajo de investigación. La Sección C.1 muestra un resumen y discusión de nuestras contribuciones, mientras que en la Sección C.2 mostramos vías de investigación que puedan ser abordadas como trabajo futuro.

C.1 Resumen y discusión de las contribuciones

En las siguientes subsecciones resumimos y discutimos las principales contribuciones de esta tesis, abordando los objetivos enunciados en el Capítulo 1. Estas contribuciones están organizadas de acuerdo a los tres objetivos principales de la tesis. Primero hemos analizado cómo evaluar adecuadamente los sistemas de recomendación para obtener medidas no sesgadas de su eficacia. Segundo hemos propuesto predictores de eficacia que tratan de estimar la eficacia de un método de recomendación. Y tercero hemos usado los predictores de eficacia propuestos para combinar dinámicamente componentes de un sistema de recomendación.

C.1.1 Análisis de la definición y evaluación de la eficacia en sistemas de recomendación

Hemos analizado distintas alternativas de diseño experimental disponibles en la literatura para sistemas de recomendación, orientados, en particular, a la evaluación basada en rankings, y hemos mostrado que **las hipótesis y condiciones subyacentes al paradigma Cranfield no se pueden asumir en los entornos habituales de recomendación**. Específicamente, hemos detectado sesgos estadísticos que aparecen al aplicar dicho paradigma a la evaluación de sistemas de recomendación. Hemos mostrado que el valor específico de la métrica de evaluación es útil en términos comparativos, pero no tiene un sentido particular en términos absolutos. Hemos mostrado que la precisión decrece linealmente con la dispersión de los ítems relevantes (**sesgo de dispersión**) al usar la metodología de evaluación AR, mientras que no sufre de este sesgo al usar la estrategia 1R.

También hemos observado que un algoritmo de recomendación no personalizado basado en la popularidad de los ítems obtiene valores de eficacia altos, y hemos mostrado y analizado en detalle cómo y por qué esto es debido a un **sesgo de popularidad** en la metodología experimental. Para abordar estos problemas, en esta tesis hemos propuesto **técnicas experimentales novedosas que neutralizan satisfactoriamente el sesgo de popularidad**.

C.1.2 Definiciones y adaptaciones de predictores de eficacia para sistemas de recomendación

En esta tesis hemos definido y elaborado **predictores de eficacia en el contexto de recomendación**, normalmente tomando al usuario como el objeto de la predicción, pero también considerando los ítems como entradas alternativas para la predicción. Específicamente, hemos adaptado el predictor de eficacia de consulta conocido como *claridad* tomando distintas hipótesis y formulaciones para obtener diferentes variaciones

de los predictores de **claridad del usuario**. También hemos usado conceptos relacionados con la Teoría de la Información como la entropía, métricas de grafos como la centralidad, PageRank y HITS, y otras técnicas heurísticas y específicas del dominio. Hemos definido estos predictores basándonos en tres espacios de entrada para las preferencias de los usuarios: **puntuaciones, registros** y **redes sociales**. Sobre puntuaciones y registros hemos definido varios modelos de lenguaje y espacios de vocabulario de tal manera que nuestras adaptaciones de claridad capturen distintos aspectos del usuario en una formulación unificada para ambos espacios de entrada. Dentro del mismo marco, hemos introducido la dimensión temporal en los datos de preferencia basados en registros, considerando y elaborando predictores de eficacia basados en tiempo propuestos en trabajos sobre búsqueda ad-hoc previos en el área de RI.

Además, hemos definido **predictores basados en ítems** cuando se usan preferencias basadas en puntuaciones, los cuales intentan estimar la eficacia de los objetos en consideración (siendo más precisos, la eficacia de un sistema de recomendación cuando sugiere dichos ítems). Aquí el principal problema consiste en cómo definir una métrica de eficacia real de manera que un predictor intente estimarla, ya que los ítems no son la entrada principal del proceso de recomendación. Por esta razón, hemos desarrollado metodologías novedosas donde la eficacia de un ítem pueda ser medida, también considerando posibles sesgos que pudieran aparecer cuando usuarios con muchas puntuaciones pueden distorsionar los resultados por razones estadísticas.

Hemos evaluado el acierto predictivo de nuestros métodos calculando la correlación entre la eficacia estimada y la real, siguiendo la práctica estándar en la literatura de predicción de eficacia en RI. De esta manera, hemos usado las metodologías no sesgadas analizadas a lo largo de esta tesis para **comparar cómo se comportan los predictores cuando los sesgos de dispersión y popularidad han sido neutralizados**. Hemos encontrado fuertes valores de correlación confirmando que nuestras técnicas muestran un **poder predictivo significativo**.

C.1.3 Ponderación dinámica en sistemas de recomendación

La combinación de algoritmos de recomendación es frecuente en la literatura de los Sistemas de Recomendación, en especial lo que se conoce como conjuntos de algoritmos de recomendación (*ensembles*), que son un tipo particular de métodos de recomendación híbrida donde se combinan varios algoritmos, y que actualmente son muy comunes en el área, tal y como se puede comprobar en competiciones recientes (Bennett and Lanning, 2007; Dror et al., 2012). El filtrado colaborativo, una de las técnicas más usadas dentro de la colección de estrategias de recomendación, también se puede ver como una combinación de varias subfunciones de utilidad, cada una correspondiendo a un vecino (en un filtrado basado en usuario). De la misma manera

que los predictores de eficacia en Recuperación de Información se han usado para optimizar la agregación de rankings, nosotros hemos investigado el uso de predictores de eficacia en recomendación para agregar dinámicamente la salida de los algoritmos de recomendación y los vecinos.

Hemos definido un **marco de hibridación dinámica** donde los conjuntos de algoritmos de recomendación pueden beneficiarse de las ponderaciones dinámicas de acuerdo a los predictores de eficacia con los que muestran correlaciones altas. Nuestros resultados indican que correlaciones altas con la eficacia tienden a corresponder con mejoras en los algoritmos de recomendación híbridos dinámicos. Además, los conjuntos dinámicos de recomendación han mostrado mejor eficacia que los conjuntos estáticos para distintas combinaciones de algoritmos y en los tres tipos de predictores de eficacia investigados.

Por otro lado, también hemos propuesto un marco para la **selección y ponderación de vecinos** en sistemas de filtrado colaborative basados en usuario. Hemos definido predictores y métricas de eficacia de vecinos adaptando e integrando algunos de los métodos de la literatura en recomendación sensibles a la confianza de usuarios. Nuestro marco unifica varias nociones de eficacia de vecinos bajo la misma forma, y presenta un análisis objetivo del poder predictivo de diferentes funciones de valoración de vecinos. Una vez el poder predictivo de estos predictores de vecinos ha sido confirmado, usamos dichos métodos para ponderar la información que proviene de cada vecino de manera dinámica, experimentando con distintas estrategias para la combinación de valores de similitud y pesos de los vecinos. Nuestros experimentos confirman una correspondencia entre los análisis de correlación y los resultados finales de eficacia, en el sentido de que los valores de correlación obtenidos entre los predictores y las métricas de eficacia de vecinos anticipan qué predictores obtendrán mejor eficacia cuando se introduzcan en un algoritmo de filtrado colaborativo basado en usuarios.

C.2 Trabajo futuro

La predicción de eficacia en recomendación es un área interesante de investigación también desde una perspectiva de negocio, ya que podríamos decidir cuándo entregar las recomendaciones al usuario, evitando disminuir la confianza de los usuarios sobre la relevancia de las sugerencias del sistema. En este sentido, las predicciones pueden dar un control al proveedor de servicios, un control que podría usarse potencialmente de varias maneras, incluyendo una combinación de métodos más general que la que se ha abordado en esta tesis. Independientemente de cualquier aplicación plausible para la industria, y más allá de los logros presentados a lo largo de esta tesis, contemplamos las líneas de investigación futuras que describimos a continuación.

La evaluación de los sistemas de recomendación es aún un objeto de investigación activa en el campo, donde varias cuestiones requieren atención, como el vacío entre experimentos en línea (*online*) y de fuera de línea (*offline*). No obstante, en esta tesis hemos enfocado nuestra investigación en aspectos relacionados con la predicción de eficacia, lo cual requiere un conocimiento más profundo de las metodologías de evaluación utilizadas. De esta manera, podríamos **extender nuestro análisis de las metodologías de evaluación a otras métricas de ranking**, como a aquellas basadas en dos listas de recomendaciones (NDPM y las correlaciones de Spearman y Kendall) o a aquellas adaptadas de Aprendizaje Automático (por ejemplo, el área bajo la curva o AUC en inglés). De esta manera, podríamos encontrar que alguna de estas métricas no está influida por ninguno de los sesgos descritos en el Capítulo 4, o que ninguno de los diseños alternativos propuestos es capaz de neutralizar esos efectos. Como un ejemplo del interés de este tema, recientemente en (Pradel et al., 2012) los autores analizaron los efectos de popularidad sobre la métrica AUC, y encontraron que considerar los datos no puntuados como información negativa durante el entrenamiento podría mejorar la eficacia, pero también podría favorecer a los algoritmos de recomendación basados en popularidad con respecto a los personalizados.

Además, sería beneficioso para nuestra investigación ser capaces de validar la utilidad de las medidas no sesgadas de eficacia con evaluaciones en línea. Esto sería valioso para tener una valoración comparativa con las observaciones fuera de línea que hemos obtenido, así como un conocimiento más profundo de la magnitud por la cual la popularidad puede ser o no una señal ruidosa. Tal estudio de usuario nos ayudaría a determinar los beneficios reales (si los hubiera) de recibir recomendaciones populares, ya que, por ejempo, por definición estas sugerencias no serían novedosas ni probablemente causales o diversas.

En el Capítulo 6 hemos propuesto varios predictores de eficacia para recomendación basados en los mismos principios de aquellos denominados como predictores pre-búsqueda en Recuperación de Información, como la claridad, donde la salida del algoritmo de búsqueda (o de recomendación en nuestro caso) no es usada por el predictor. Teniendo en cuenta nuestros resultados, las posibilidades para investigar más predictores de eficacia en recomendación son abundantes. En esta línea, varios autores han explotado la **combinación de predictores para obtener valores de correlación mayores y un poder predictivo mayor**, como (Hauff et al., 2009) y (Jones and Diaz, 2007), donde se han usado regresión penalizada y regresión lineal con aprendizaje mediante redes neuronales, respectivamente. En esos trabajos la combinación de predictores de distinta naturaleza mejoró la correlación con respecto a una métrica de evaluación objetivo – en este caso, la precisión promedio. Por ello, creemos que la combinación de predictores puede ser válida también para recomendación, especialmente sabiendo que hemos definido predictores basados en diferentes tipos de datos de los que se espera baja redundancia entre ellos y, por tanto, que dicha

combinación pueda producir correlaciones mayores. Ejemplos de tales combinaciones podrían ser la mezcla de dimensiones sociales y temporales, los predictores temporales basados en ítems, u otras dimensiones contextuales no abordadas en esta tesis.

Más aún, una futura investigación podría **analizar y adaptar también a los sistemas de recomendación predictores de eficacia post-búsqueda** definidos en la literatura de RI, como por ejemplo aquellos basados en el análisis de la distribución de las puntuaciones de los ítems recomendados a cada usuario. Esto podría conseguir predictores con correlaciones más fuertes y, por tanto, con mayor poder predictivo de la eficacia de los algoritmos de recomendación, como ocurre en RI donde los predictores post-búsqueda normalmente obtienen valores de correlación mayores que los pre-búsqueda. La principal limitación de este tipo de predictores es que no pueden ser usados directamente para adaptar la salida de los algoritmos de recomendación, ya que normalmente se requiere la salida completa – es decir, el ranking – para el cálculo de los valores del predictor. Esto obligaría a pensar en distintas aplicaciones donde este tipo de predictores pudieran ser usados en recomendación.

Una dirección particular digna de ser considerada y también relacionada con el Capítulo 6, sería el **uso de técnicas de evaluación alternativas** más allá de las métricas de correlación, como aquellas basadas en el agrupamiento entre los valores de eficacias reales y estimados (ver Sección 5.4.2). En nuestro trabajo nos hemos centrado en el uso de métricas de correlación, principalmente la correlación de Pearson. Estas métricas tienen limitaciones bien conocidas, como su sensibilidad a los valores extremos y correlaciones no significativas cuando se usan un número pequeño de puntos. Por esta razón, se han propuesto otras técnicas para evaluar el poder predictivo de los predictores. Hemos de notar, sin embargo, que el uso de una técnica particular de evaluación debería enfocarse a su aplicación en contextos específicos (Pérez-Iglesias and Araujo, 2010); en particular esto requiere la definición de nuevas aplicaciones para predictores de eficacia que encajen con la métrica de evaluación, lo cual también contemplamos como un potencial trabajo futuro.

También en el Capítulo 6 hemos desarrollado una metodología de evaluación para estimar el valor real de eficacia de los ítems, con el objetivo de evaluar los predictores de ítems propuestos. Esta metodología debería ser validada para **obtener una medida justa de la eficacia del ítem**, lo cual en este momento es aún un problema abierto. De esta manera, seríamos capaces de definir predictores de ítems adicionales para otros espacios de entrada además de las puntuaciones, y de mejorar la capacidad de predicción de los actuales predictores de eficacia de ítems.

En el Capítulo 7 presentamos experimentos sobre combinación dinámica de algoritmos de recomendación en conjunto. Esos experimentos estaban limitados a un único predictor de eficacia por cada par de algoritmos, así que pretendemos extender dichos experimentos con **conjuntos de algoritmos de recomendación donde se consideren dos predictores** para investigar qué condiciones deberían satisfacerse

entre cada par de predictores de manera que se mejore la eficacia del conjunto. Una vía de investigación relacionada a considerar sería el análisis de valores de correlación tales que se obtengan buenos resultados de eficacia en los métodos híbridos dinámicos. Más específicamente, queremos saber si es mejor tener un fuerte valor de correlación en general (en promedio) o un valor medio no tan fuerte pero mejores estimaciones para usuarios particulares, que tendrían un papel significativo en el sistema similar a los usuarios poderosos (*power users*) definidos en (Lathia et al., 2008). En ese punto, se podría realizar un estudio como el presentado en (Hauff et al., 2010), donde simulaciones de predictores con distintos valores de correlación son evaluados, y cuyos efectos sobre la eficacia final en conjuntos de algoritmos de recomendación son comparados.

Además, otra limitación de los experimentos presentados en el Capítulo 7 es que el tamaño de los conjuntos siempre es dos. Pretendemos considerar **conjuntos de algoritmos de recomendación de tamaño N** y a la larga, como se mencionó antes, usar un predictor de eficacia para cada algoritmo de recomendación. Este es un paso natural, pero no trivial hacia la generalización del marco propuesto de conjuntos completos de algoritmos de recomendación. De manera alternativa, podrían usarse técnicas de Aprendizaje Automático para aprender los mejores pesos a usar por cada usuario e ítem en el conjunto. En este caso, se debería investigar un compromiso entre los costes computacionales de cada técnica (aprendizaje automático frente a predictores de eficacia), su poder predictivo y la tendencia a sobreajustar los datos.

Finalmente, en el Capítulo 8 investigamos el problema de ponderación dinámica de vecinos usando predictores de eficacia de vecinos orientados a métricas de error. El trabajo futuro relacionado con este capítulo podría centrarse en la **adaptación de las métricas de eficacia de vecinos usadas en nuestra propuesta hacia métricas de ranking**, tales como la precisión y el *recall*. Como ya hemos discutido, las métricas de error no son la mejor manera de medir la eficacia, aunque se pueden considerar apropiadas en este contexto ya que queremos medir la mejora en la exactitud de nuestras propuestas, y a la vez facilitar comparaciones con el estado del arte en recomendación sensible a la confianza, donde estas métricas son predominantes. Por lo tanto, el uso de métricas de ranking sería una valiosa contribución al campo por sí misma. Además, una vez tuviéramos definidas métricas de eficacia de vecinos basadas en ranking, seríamos capaces de medir la correlación de los predictores de eficacia de vecinos descritos en este capítulo con tales métricas, y analizar en detalle su poder predictivo con métricas de ranking. Idealmente, podríamos obtener un predictor con suficiente poder predictivo usando los dos tipos de métricas de eficacia de vecinos (las basadas en error y en ranking), aunque esto no es fácil de garantizar en general, ya que cada métrica se define para optimizar distintos parámetros y conceptos.

References

Adams, R. A. (2007). Music Recommendation using Collaborative Filtering with Similarity Fusion. Master's thesis, Department of Computer Science, The University of York.

Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749.

Adomavicius, G., Tuzhilin, A., Berkovsky, S., De Luca, E. W., and Said, A. (2010). Context-awareness in recommender systems: research workshop and movie recommendation challenge. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 385–386, New York, NY, USA. ACM.

Aggarwal, C. C., Wolf, J. L., Wu, K.-L., and Yu, P. S. (1999). Horting hatches an egg: a new graph-theoretic approach to collaborative filtering. In *the fifth ACM SIGKDD international conference*, pages 201–212, New York, New York, USA. ACM Press.

Agrawal, R., Gollapudi, S., Halverson, A., and Ieong, S. (2009). Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 5–14, New York, NY, USA. ACM.

Aizawa, A. (2003). An information-theoretic perspective of tf-idf measures. *Information Processing & Management*, 39(1):45–65.

Albert, R. and Barabási, A. L. (2002). Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97.

Ali, K. and van Stam, W. (2004). TiVo: making show recommendations using a distributed collaborative filtering architecture. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 394–401, New York, NY, USA. ACM.

Allan, J. and Raghavan, H. (2002). Using Part-of-speech Patterns to Reduce Query Ambiguity. In *25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 307–314.

Alvarez, M. M., Yahyaei, S., and Roelleke, T. (2012). Semi-automatic document classification: Exploiting document difficulty. In Baeza Yates, R. A., de Vries, A. P., Zaragoza, H., Cambazoglu, B. B., Murdock, V., Lempel, R., Silvestri, F., Baeza Yates, R. A., de Vries, A. P., Zaragoza, H., Cambazoglu, B. B., Murdock, V., Lempel, R., and Silvestri, F., editors, *ECIR*, volume 7224 of *Lecture Notes in Computer Science*, pages 468–471. Springer.

Amati, G., Carpineto, C., and Romano, G. (2004). Query Difficulty, Robustness, and Selective Application of Query Expansion. *Advances in Information Retrieval*, pages 127–137.

Arazy, O., Kumar, N., and Shapira, B. (2009). Improving social recommender systems. *IT Professional*, 11(4):38–44.

Armstrong, T. G., Moffat, A., Webber, W., and Zobel, J. (2009a). Has adhoc retrieval improved since 1994? In Allan, J., Aslam, J. A., Sanderson, M., Zhai, C., and Zobel, J., editors, *SIGIR*, pages 692–693. ACM.

Armstrong, T. G., Moffat, A., Webber, W., and Zobel, J. (2009b). Improvements that don't add up: ad-hoc retrieval results since 1998. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 601–610, New York, NY, USA. ACM.

Aslam, J. A. and Pavlu, V. (2007). Query Hardness Estimation Using Jensen-Shannon Divergence Among Multiple Scoring Functions. In *ECIR*, pages 198–209.

Aslam, J. A., Pavlu, V., and Yilmaz, E. (2006). A statistical method for system evaluation using incomplete judgments. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 541–548, New York, NY, USA. ACM.

Baeza-Yates, R. and Ribeiro-Neto, B. (2011). *Modern Information Retrieval: The Concepts and Technology behind Search (2nd Edition) (ACM Press Books)*. Addison-Wesley Professional, 2 edition.

Balabanovic, M. and Shoham, Y. (1997). Fab: content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72.

Baltrunas, L. and Amatriain, X. (2009). Towards Time-Dependant Recommendation based on Implicit Feedback. In *Context-aware Recommender Systems Workshop at Recsys09*.

Bao, X., Bergman, L., and Thompson, R. (2009). Stacking recommendation engines with additional meta-features. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 109–116, New York, NY, USA. ACM.

Barbieri, N., Costa, G., Manco, G., and Ortale, R. (2011). Modeling item selection and relevance for accurate recommendations: a bayesian approach. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 21–28, New York, NY, USA. ACM.

Barman, K. and Dabeer, O. (2010). What is Popular Amongst Your Friends?

Basu, C., Hirsh, H., and Cohen, W. W. (1998). Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In *AAAI/IAAI*, pages 714–720.

Belkin, N. J. and Croft, W. B. (1992). Information filtering and information retrieval: two sides of the same coin? *Commun. ACM*, 35(12):29–38.

Bell, R. M. and Koren, Y. (2007). Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights. In *ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, pages 43–52, Washington, DC, USA. IEEE Computer Society.

Bellogín, A. (2009). Performance prediction in recommender systems: application to the dynamic optimisation of aggregative methods. Master's thesis, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Madrid, Spain.

Bellogín, A., Cantador, I., and Castells, P. (2010). A study of heterogeneity in recommendations for a social music service. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '10, pages 1–8, New York, NY, USA. ACM.

Bellogín, A., Cantador, I., Díez, F., Castells, P., and Chavarriaga, E. (2012). An empirical comparison of social, collaborative filtering, and hybrid recommenders. *ACM Transactions on Intelligent Systems and Technology*, to appear.

Bellogín, A. and Castells, P. (2010). A Performance Prediction Approach to Enhance Collaborative Filtering Performance. In Gurrin, C., He, Y., Kazai, G., Kruschwitz, U., Little, S., Roelleke, T., Rüger, S., and Rijsbergen, editors, *Advances in Information Retrieval*, volume 5993 of *Lecture Notes in Computer Science*, pages 382–393, Berlin, Heidelberg. Springer Berlin / Heidelberg.

Bellogín, A., Castells, P., and Cantador, I. (2011a). Precision-oriented evaluation of recommender systems: an algorithmic comparison. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 333–336, New York, NY, USA. ACM.

Bellogín, A., Wang, J., and Castells, P. (2011b). Text Retrieval Methods for Item Ranking in Collaborative Filtering. In Clough, P., Foley, C., Gurrin, C., Jones, G., Kraaij, W., Lee, H., and Mudoch, V., editors, *Advances in Information Retrieval*, volume 6611 of *Lecture Notes in Computer Science*, chapter 30, pages 301–306. Springer Berlin / Heidelberg, Berlin, Heidelberg.

Ben-Shimon, D., Tsikinovsky, A., Rokach, L., Meisles, A., Shani, G., and Naamani, L. (2007). Recommender System from Personal Social Networks Advances in Intelligent Web Mastering. In Wegrzyn-Wolska, K. and Szczepaniak, P., editors, *Advances in Intelligent Web Mastering*, volume 43 of *Advances in Soft Computing*, chapter 8, pages 47–55. Springer Berlin / Heidelberg, Berlin, Heidelberg.

Bennett, J. and Lanning, S. (2007). The netflix prize. In *Proceedings of the KDD Cup Workshop 2007*, pages 3–6, New York. ACM.

Berberich, K., Bedathur, S., Alonso, O., and Weikum, G. (2010). A language modeling approach for temporal information needs. In Gurrin, C., He, Y., Kazai, G., Kruschwitz, U., Little, S., Roelleke, T., Rüger, S., and Rijsbergen, K., editors, *Advances in Information Retrieval*, volume 5993 of *Lecture Notes in Computer Science*, pages 13–25, Berlin, Heidelberg. Springer Berlin / Heidelberg.

Bernhardsson, E. (2009). Implementing a scalable music recommender system. Master's thesis, School of Engineering Physics, Royal Institute of Technology, Stockholm, Sweden.

Best, D. J. and Gipps, P. G. (1974). Algorithm AS 71: The upper tail probabilities of kendall's tau. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 23(1):98–100.

Billsus, D. and Pazzani, M. J. (1998). Learning collaborative information filters. In Shavlik, J. W. and Shavlik, J. W., editors, *ICML*, pages 46–54. Morgan Kaufmann.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Bollen, D., Knijnenburg, B. P., Willemsen, M. C., and Graus, M. (2010). Understanding choice overload in recommender systems. In *RecSys '10: Proceedings of the fourth ACM conference on Recommender systems*, pages 63–70, New York, NY, USA. ACM.

Bourke, S., McCarthy, K., and Smyth, B. (2011). Power to the people: exploring neighbourhood formations in social recommender system. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 337–340, New York, NY, USA. ACM.

Bradley, K. and Smyth, B. (2001). Improving Recommendation Diversity. In *Proceedings of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science*.

Brants, T., Chen, F., and Tsochantaridis, I. (2002). Topic-based document segmentation with probabilistic latent semantic analysis. In *CIKM*, CIKM '02, pages 211–218, New York, NY, USA. ACM.

Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 43–52.

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117.

Broder, A. (2002). A taxonomy of web search. *SIGIR Forum*, 36(2):3–10.

Buckley, C. (2004). Topic prediction based on comparative retrieval rankings. In Sanderson, M., Järvelin, K., Allan, J., Bruza, P., Sanderson, M., Järvelin, K., Allan, J., and Bruza, P., editors, *SIGIR*, pages 506–507, New York, NY, USA. ACM.

Buckley, C., Dimmick, D., Soboroff, I., and Voorhees, E. (2006). Bias and the limits of pooling. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 619–620, New York, NY, USA. ACM.

Buckley, C., Dimmick, D., Soboroff, I., and Voorhees, E. (2007). Bias and the limits of pooling for large collections. *Information Retrieval*, 10(6):491–508.

Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370.

Burke, R. (2004). Hybrid Recommender Systems with Case-Based Components. pages 91–105.

Burke, R. (2010). Evaluating the dynamic properties of recommendation algorithms. In *RecSys '10: Proceedings of the fourth ACM conference on Recommender systems*, pages 225–228, New York, NY, USA. ACM.

Cantador, I. (2008). *Exploiting the conceptual space in hybrid recommender systems: a semantic-based approach*. PhD thesis, Universidad Autonoma de Madrid.

Cantador, I., Bellogín, A., and Vallet, D. (2010). Content-based recommendation in social tagging systems. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 237–240, New York, NY, USA. ACM.

Cantador, I., Brusilovsky, P., and Kuflik, T. (2011). Second workshop on information heterogeneity and fusion in recommender systems (HetRec2011). In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 387–388, New York, NY, USA. ACM.

Cantador, I. and Castells, P. (2006). Multilayered Semantic Social Network Modeling by Ontology-Based User Profiles Clustering: Application to Collaborative Filtering. In *Managing Knowledge in a World of Networks*, pages 334–349.

Carmel, D. and Yom-Tov, E. (2010). *Estimating the Query Difficulty for Information Retrieval*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers.

Carmel, D., Yom-Tov, E., Darlow, A., and Pelleg, D. (2006). What makes a query difficult? In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 390–397, New York, NY, USA. ACM.

Celma, O. (2008). *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain.

Celma, O. (2010). *Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space*. Springer, 1st edition. edition.

Celma, O. and Cano, P. (2008). From hits to niches?: or how popular artists can bias music recommendation and discovery. In *NETFLIX '08: Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, pages 1–8, New York, NY, USA. ACM.

Celma, O. and Herrera, P. (2008). A new approach to evaluating novel recommendations. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 179–186, New York, NY, USA. ACM.

Chandar, P. and Carterette, B. (2010). Diversification of search results using webgraphs. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 869–870, New York, NY, USA. ACM.

Clarke, C. L. A., Kolla, M., Cormack, G. V., Vechtomova, O., Ashkan, A., Büttcher, S., and MacKinnon, I. (2008). Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 659–666, New York, NY, USA. ACM.

Clements, M., de Vries, A., and Reinders, M. J. T. (2009). Exploiting Positive and Negative Graded Relevance Assessments for Content Recommendation. In *Proceedings of the 6th International Workshop on Algorithms and Models for the Web-Graph*, WAW '09, pages 155–166, Berlin, Heidelberg. Springer-Verlag.

Clements, M., De Vries, A. P., and Reinders, M. J. T. (2010). The task-dependent effect of tags and ratings on social media access. *ACM Trans. Inf. Syst.*, 28.

Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory*. Wiley-Interscience, 99th edition.

Cremonesi, P., Garzotto, F., Negro, S., Papadopoulos, A., and Turrin, R. (2011). Comparative evaluation of recommender system quality. In *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems*, CHI EA '11, pages 1927–1932, New York, NY, USA. ACM.

Cremonesi, P., Koren, Y., and Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 39–46, New York, NY, USA. ACM.

Croft, B., Metzler, D., and Strohman, T. (2009). *Search Engines: Information Retrieval in Practice*. Addison Wesley, 1 edition.

Cronen-Townsend, S., Zhou, Y., and Croft, W. B. (2002). Predicting query performance. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 299–306, New York, NY, USA. ACM.

Cronen-Townsend, S., Zhou, Y., and Croft, W. B. (2006). Precision prediction based on ranked list coherence. *Information Retrieval*, 9(6):723–755.

Cummins, R. (2012). Investigating performance predictors using monte carlo simulation and score distribution models. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '12, pages 1097–1098, New York, NY, USA. ACM.

Cummins, R., Jose, J., and O'Riordan, C. (2011). Improved query performance prediction using standard deviation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information*, SIGIR '11, pages 1089–1090, New York, NY, USA. ACM.

Dang, V., Bendersky, M., and Croft, W. B. (2010). Learning to rank query reformulations. In Crestani, F., Maillet, S. M., Chen, H. H., Efthimiadis, E. N., Savoy, J., Crestani, F., Maillet, S. M., Chen, H. H., Efthimiadis, E. N., and Savoy, J., editors, *SIGIR*, SIGIR '10, pages 807–808, New York, NY, USA. ACM.

Das, A. S., Datar, M., Garg, A., and Rajaram, S. (2007). Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 271–280, New York, NY, USA. ACM.

De Choudhury, M., Mason, W. A., Hofman, J. M., and Watts, D. J. (2010). Inferring relevant social networks from interpersonal communication. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 301–310, New York, NY, USA. ACM.

de Gemmis, M., Lops, P., Semeraro, G., and Basile, P. (2008). Integrating tags in a semantic content-based recommender. In *Proceedings of the 2008 ACM conference on Recommender systems*, RecSys '08, pages 163–170, New York, NY, USA. ACM.

Demidova, E., Fankhauser, P., Zhou, X., and Nejdl, W. (2010). DivQ: diversification for keyword search over structured databases. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 331–338, New York, NY, USA. ACM.

Deshpande, M. and Karypis, G. (2004). Item-based top-N recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177.

Desrosiers, C. and Karypis, G. (2011). A comprehensive survey of neighborhood-based recommendation methods. In Ricci, F., Rokach, L., Shapira, B., Kantor, P. B., Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, chapter 4, pages 107–144. Springer, Boston, MA.

Diaz, F. (2007). Performance prediction using spatial autocorrelation. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 583–590, New York, NY, USA. ACM.

Diaz, F. and Jones, R. (2004). Using temporal profiles of queries for precision prediction. In *SIGIR '04: Proceedings of the 27th annual international conference on Research and development in information retrieval*, pages 18–24. ACM Press.

Diederich, J. and Iofciu, T. (2006). Finding communities of practice from user profiles based on folksonomies. In Tomadaki, E., Scott, P. J., Tomadaki, E., and Scott, P. J., editors, *EC-TEL Workshops*, volume 213 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

Dror, G., Koenigstein, N., Koren, Y., and Weimer, M. (2012). The yahoo! music dataset and KDD-cup'11. *JMLR Workshop and Conference Proceedings*, 18:3–18.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2 edition.

Eirinaki, M., Vazirgiannis, M., and Varlamis, I. (2003). SEWeP: using site semantics and a taxonomy to enhance the web personalization process. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108, New York, NY, USA. ACM Press.

Elahi, M. (2011). Adaptive Active Learning in Recommender Systems. In Konstan, J., Conejo, R., Marzo, J., and Oliver, N., editors, *User Modeling, Adaption and Personalization*, volume 6787 of *Lecture Notes in Computer Science*, chapter 40, pages 414–417. Springer Berlin / Heidelberg, Berlin, Heidelberg.

Endres, D. M. and Schindelin, J. E. (2003). A new metric for probability distributions. *Information Theory, IEEE Transactions on*, 49(7):1858–1860.

Fernández, M., Vallet, D., and Castells, P. (2006a). Probabilistic Score Normalization for Rank Aggregation. In *28th European Conference on Information Retrieval (ECIR 2006)*, pages 553–556. Springer Verlag Lecture Notes in Computer Science, Vol. 3936.

Fernández, M., Vallet, D., and Castells, P. (2006b). Using historical data to enhance rank aggregation. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 643–644, New York, NY, USA. ACM.

Filippone, M. and Sanguinetti, G. (2010). Information Theoretic Novelty Detection. *Pattern Recognition*, 43(3):805–814.

Fleder, D. and Hosanagar, K. (2009). Blockbuster Culture's Next Rise or Fall: The Impact of Recommender Systems on Sales Diversity. *Manage. Sci.*, 55:697–712.

Foltz, P. W. and Dumais, S. T. (1992). Personalized information delivery: An analysis of information filtering methods. *Commun. ACM*, 35(12):51–60.

Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41.

Gantner, Z., Rendle, S., and Lars, S. T. (2010). Factorization models for context-/time-aware movie recommendations. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, CAMRa '10, pages 14–19, New York, NY, USA. ACM.

Ge, M., Battenfeld, C. D., and Jannach, D. (2010). Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 257–260, New York, NY, USA. ACM.

Golbeck, J. (2006). Trust on the world wide web: A survey. *Foundations and Trends in Web Science*, 1(2):131–197.

Golbeck, J. (2009). Trust and nuanced profile similarity in online social networks. *ACM Trans. Web*, 3(4):1–33.

Golbeck, J. and Hendler, J. (2006). FilmTrust: movie recommendations using trust in web-based social networks. In *Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE*, pages 282–286.

Goldberg, D., Nichols, D. A., Oki, B. M., and Terry, D. B. (1992). Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70.

Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. (2001). Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Inf. Retr.*, 4(2):133–151.

Grivolla, Jourlin, and Mori, D. (2005). Automatic classification of queries by expected retrieval performance. In *Predicting Query Difficulty - Methods and Applications, SIGIR 2005*.

Gunawardana, A. and Meek, C. (2009). A unified approach to building hybrid recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 117–124, New York, NY, USA. ACM.

Gunawardana, A. and Shani, G. (2009). A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. *J. Mach. Learn. Res.*, 10:2935–2962.

Guo, G., Zhang, J., and Thalmann, D. (2012). A simple but effective method to incorporate trusted neighbors in recommender systems. In Masthoff, J., Mobasher, B., Desmarais, M. C., and Nkambou, R., editors, *User Modeling, Adaptation, and Personalization*, volume 7379 of *Lecture Notes in Computer Science*, pages 114–125, Berlin, Heidelberg. Springer Berlin / Heidelberg.

Hauff, C. (2010). *Predicting the Effectiveness of Queries and Retrieval Systems*. PhD thesis, Univ. of Twente, Enschede.

Hauff, C., Azzopardi, L., and Hiemstra, D. (2009). The Combination and Evaluation of Query Performance Prediction Methods. In Boughanem, M., Berrut, C., Mothe, J., Dupuy, C. S., Boughanem, M., Berrut, C., Mothe, J., and Dupuy, C. S., editors, *ECIR*, volume 5478 of *Lecture Notes in Computer Science*, pages 301–312. Springer.

Hauff, C., Azzopardi, L., Hiemstra, D., and Jong, F. (2010). Query performance prediction: Evaluation contrasted with effectiveness. In Gurrin, C., He, Y., Kazai, G., Kruschwitz, U., Little, S., Roelleke, T., Rüger, S., and Rijsbergen, K., editors, *Advances in Information Retrieval*, volume 5993 of *Lecture Notes in Computer Science*, pages 204–216, Berlin, Heidelberg. Springer Berlin / Heidelberg.

Hauff, C., Hiemstra, D., and de Jong, F. (2008a). A survey of pre-retrieval query performance predictors. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 1419–1420, New York, NY, USA. ACM.

Hauff, C., Murdock, V., and Yates, R. B. (2008b). Improved query difficulty prediction for the web. In *Proceeding of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 439–448, New York, NY, USA. ACM.

He, B. and Ounis, I. (2004). Inferring Query Performance Using Pre-retrieval Predictors. In *String Processing and Information Retrieval, SPIRE 2004*, pages 43–54.

He, J., Larson, M., and de Rijke, M. (2008). Using Coherence-Based Measures to Predict Query Difficulty. In Macdonald, C., Ounis, I., Plachouras, V., Ruthven, I., and White, R. W., editors, *Advances in Information Retrieval*, volume 4956 of *Lecture Notes in Computer Science*, chapter 80, pages 689–694. Springer Berlin Heidelberg, Berlin, Heidelberg.

Herlocker, J., Konstan, J. A., and Riedl, J. (2002). An Empirical Analysis of Design Choices in Neighborhood-Based Collaborative Filtering Algorithms. *Information Retrieval*, 5(4):287–310.

Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 230–237, New York, NY, USA. ACM.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53.

Hiemstra, D. (1998). A Linguistically Motivated Probabilistic Model of Information Retrieval. In *ECDL '98: Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries*, pages 569–584, London, UK. Springer-Verlag.

Hofmann, T. (2003). Collaborative filtering via gaussian probabilistic latent semantic analysis. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 259–266, New York, NY, USA. ACM.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115.

Hotho, A., Jäschke, R., Schmitz, C., and Stumme, G. (2006). Information Retrieval in Folksonomies: Search and Ranking. In Sure, Y. and Domingue, J., editors, *The Semantic Web: Research and Applications*, volume 4011 of *Lecture Notes in Computer Science*, chapter 31, pages 411–426. Springer Berlin / Heidelberg, Berlin, Heidelberg.

Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative Filtering for Implicit Feedback Datasets. In *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, volume 0, pages 263–272, Washington, DC, USA. IEEE.

Huang, Z., Zeng, D. D., and Chen, H. (2006). A Unified Recommendation Framework Based on Probabilistic Relational Models. *Social Science Research Network Working Paper Series*.

Hummel, S., Shtok, A., Raiber, F., Kurland, O., and Carmel, D. (2012). Clarity re-visited. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '12, pages 1039–1040, New York, NY, USA. ACM.

Hwang, C.-S. and Chen, Y.-P. (2007). Using Trust in Collaborative Filtering Recommendation. In Okuno, H. and Ali, M., editors, *New Trends in Applied Artificial Intelligence*, volume 4570 of *Lecture Notes in Computer Science*, chapter 105, pages 1052–1060. Springer Berlin / Heidelberg, Berlin, Heidelberg.

Jahrer, M., Töscher, A., and Legenstein, R. (2010). Combining predictions for accurate recommender systems. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 693–702, New York, NY, USA. ACM.

Jamali, M. and Ester, M. (2009). Using a trust network to improve top-N recommendation. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 181–188, New York, NY, USA. ACM.

Jambor, T. and Wang, J. (2010a). Goal-Driven Collaborative Filtering â€“ A Directional Error Based Approach. In Gurrin, C., He, Y., Kazai, G., Kruschwitz, U., Little, S., Roelleke, T., Rüger, S., and Rijsbergen, K., editors, *Advances in Information Retrieval*, volume 5993, chapter 36, pages 407–419. Springer Berlin Heidelberg, Berlin, Heidelberg.

Jambor, T. and Wang, J. (2010b). Optimizing multiple objectives in collaborative filtering. In *RecSys '10: Proceedings of the fourth ACM conference on Recommender systems*, pages 55–62, New York, NY, USA. ACM.

Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

Jawaheer, G., Szomszor, M., and Kostkova, P. (2010). Comparison of implicit and explicit feedback from an online music recommendation service. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '10, pages 47–51, New York, NY, USA. ACM.

Jensen, E. C., Beitzel, S. M., Grossman, D., Frieder, O., and Chowdhury, A. (2005). Predicting query difficulty on the web by learning visual clues. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 615–616, New York, NY, USA. ACM.

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–20.

Jones, R. and Diaz, F. (2007). Temporal profiles of queries. *ACM Trans. Inf. Syst.*, 25(3).

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632.

Kohavi, R., Longbotham, R., Sommerfield, D., and Henne, R. M. (2009). Controlled experiments on the web: survey and practical guide. *Data Min. Knowl. Discov.*, 18(1):140–181.

Kohrs, A. and Merialdo, B. (1999). Clustering for Collaborative Filtering Applications. In *Computational Intelligence for Modelling, Control & Automation (CIMCA '99)*.

Kompaoré, D., Mothe, J., Baccini, A., and Déjean, S. (2007). Prédiction du sri à utiliser en fonction des critères linguistiques de la requête. In *CORIA*, pages 239–254. Université de Saint-Étienne.

Konstas, I., Stathopoulos, V., and Jose, J. M. (2009). On social networks and collaborative recommendation. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 195–202, New York, NY, USA. ACM.

Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 426–434, New York, NY, USA. ACM.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37.

Koren, Y. and Bell, R. M. (2011). Advances in collaborative filtering. In Ricci, F., Rokach, L., Shapira, B., Kantor, P. B., Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, chapter 5, pages 145–186. Springer, Boston, MA.

Kossinets, G. and Watts, D. J. (2006). Empirical analysis of an evolving social network. *Science*, 311(5757):88–90.

Kulkarni, A., Teevan, J., Svore, K. M., and Dumais, S. T. (2011). Understanding temporal query dynamics. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 167–176, New York, NY, USA. ACM.

Kuncheva, L. I. (2004). *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience.

Kuncheva, L. I. and Whitaker, C. J. (2003). Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy. *Machine Learning*, 51(2):181–207.

Kwok, K. L., Grunfeld, L., Sun, H. L., and Deng, P. (2004). TREC 2004 Robust Track Experiments Using PIRCS. In *Online Proceedings of 2004 Text REtrieval*.

Kwon, K., Cho, J., and Park, Y. (2009). Multidimensional credibility model for neighbor selection in collaborative recommendation. *Expert Systems with Applications*, 36(3):7114–7122.

Kwon, Y. (2008). Improving top-n recommendation techniques using rating variance. In *Proceedings of the 2008 ACM conference on Recommender systems*, RecSys '08, pages 307–310, New York, NY, USA. ACM.

Lathia, N. (2010). *Evaluating Collaborative Filtering Over Time*. PhD thesis, University of London, Department of Computer Science, University College London.

Lathia, N., Hailes, S., and Capra, L. (2008). kNN CF: a temporal social network. In *RecSys '08: Proceedings of the 2008 ACM Conference on Recommender Systems*, pages 227–234, New York, NY, USA. ACM.

Lathia, N., Hailes, S., Capra, L., and Amatriain, X. (2010). Temporal diversity in recommender systems. In *SIGIR '10: Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 210–217, New York, NY, USA. ACM.

Lavrenko, V., Allan, J., Deguzman, E., Laflamme, D., Pollard, V., and Thomas, S. (2002). Relevance models for Topic Detection and Tracking. In *Human Language Technology 2002*, pages 104–110.

Lee, D. H. and Brusilovsky, P. (2009). Reinforcing Recommendation Using Implicit Negative Feedback. In *Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization: formerly UM and AH*, volume 5535 of *UMAP '09*, pages 422–427, Berlin, Heidelberg. Springer-Verlag.

Lee, T., Park, Y., and Park, Y. (2008). A time-based approach to effective recommender systems using implicit feedback. *Expert Systems with Applications*, 34(4):3055–3062.

Lieberman, H. (1995). Letizia: An agent that assists web browsing. In *IJCAI (1)*, pages 924–929. Morgan Kaufmann.

Linden, G., Smith, B., and York, J. (2003). Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80.

Liu, F. and Lee, H. J. (2010). Use of social network information to enhance collaborative filtering performance. *Expert Systems with Applications*, 37(7):4772–4778.

Liu, K., Fang, B., and Zhang, W. (2010). Speak the same language with your friends: augmenting tag recommenders with social relations. In *Proceedings of the 21st ACM conference on Hypertext and hypermedia*, HT '10, pages 45–50, New York, NY, USA. ACM.

Lops, P., de Gemmis, M., and Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In Ricci, F., Rokach, L., Shapira, B., Kantor, P. B., Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, chapter 3, pages 73–105. Springer, Boston, MA.

Ma, H., King, I., and Lyu, M. R. (2007). Effective missing data prediction for collaborative filtering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 39–46, New York, NY, USA. ACM.

Ma, H., King, I., and Lyu, M. R. (2009). Learning to recommend with social trust ensemble. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 203–210, New York, NY, USA. ACM.

Ma, H., Yang, H., Lyu, M. R., and King, I. (2008). SoRec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 931–940, New York, NY, USA. ACM.

Ma, H., Zhou, D., Liu, C., Lyu, M. R., and King, I. (2011). Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 287–296, New York, NY, USA. ACM.

Macdonald, C., He, B., and Ounis, I. (2005). Predicting Query Performance in Intranet Search. In *Predicting Query Difficulty - Methods and Applications, SIGIR 2005*.

Magnini, B. and Strapparava, C. (2001). Improving user modelling with Content-Based techniques. In Bauer, M., Gmytrasiewicz, P. J., Vassileva, J., Bauer, M., Gmy-

trasiewicz, P. J., and Vassileva, J., editors, *User Modeling*, volume 2109 of *Lecture Notes in Computer Science*, pages 74–83. Springer.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press, 1 edition.

Marlin, B. (2003). Modeling user rating profiles for collaborative filtering. In *In NIPS*17*.

Marlin, B. M., Zemel, R. S., Roweis, S., and Slaney, M. (2007). Collaborative filtering and the missing at random assumption. In *Proc. of the 23rd Conference on Uncertainty in Artificial Intelligence*.

Martinez, A., Arias, J., Vilas, A., Garcia, and Lopez (2009). What's on TV tonight? An efficient and effective personalized recommender system of TV programs. *Consumer Electronics, IEEE Transactions on*, 55(1):286–294.

Marx, P., Thurau, T. H., and Marchand, A. (2010). Increasing consumers' understanding of recommender results: a preference-based hybrid algorithm with strong explanatory power. In *RecSys '10: Proceedings of the fourth ACM conference on Recommender systems*, pages 297–300, New York, NY, USA. ACM.

Massa, P. and Avesani, P. (2004). Trust-Aware Collaborative Filtering for Recommender Systems. In Meersman, R. and Tari, Z., editors, *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, volume 3290 of *Lecture Notes in Computer Science*, chapter 31, pages 492–508. Springer Berlin / Heidelberg, Berlin, Heidelberg.

Massa, P. and Avesani, P. (2007a). Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*, RecSys '07, pages 17–24, New York, NY, USA. ACM.

Massa, P. and Avesani, P. (2007b). Trust metrics on controversial users: balancing between tyranny of the majority and echo chambers. *International Journal on Semantic Web and Information Systems*.

Massa, P. and Bhattacharjee, B. (2004). Using trust in recommender systems: An experimental analysis. In Jensen, C. D., Poslad, S., Dimitrakos, T., Jensen, C. D., Poslad, S., and Dimitrakos, T., editors, *iTrust*, volume 2995 of *Lecture Notes in Computer Science*, pages 221–235, Berlin, Heidelberg. Springer.

McLaughlin, M. R. and Herlocker, J. L. (2004). A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 329–336, New York, NY, USA. ACM.

McNee, S. M., Riedl, J., and Konstan, J. A. (2006). Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI '06 extended abstracts on Human factors in computing systems*, CHI EA '06, pages 1097–1101, New York, NY, USA. ACM.

Melucci, M. (2009). Weighted rank correlation in information retrieval evaluation. In Lee, G. G., Song, D., Lin, C. Y., Aizawa, A. N., Kuriyama, K., Yoshioka, M., Sakai, T., Lee, G. G., Song, D., Lin, C. Y., Aizawa, A. N., Kuriyama, K., Yoshioka, M., and Sakai, T., editors, *AIRS*, volume 5839 of *Lecture Notes in Computer Science*, pages 75–86. Springer.

Michlmayr, E. and Cazer, S. (2007). Learning user profiles from tagging data and leveraging them for personal(ized) information access. In *Tagging and Metadata for Social Information Organization Workshop in conjunction with the 16th International World Wide Web Conference*.

Milgram, S. (1967). The small world problem. *Psychology Today*, 1:61–67.

Mooney, R. J. and Roy, L. (2000). Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, DL '00, pages 195–204, New York, NY, USA. ACM.

Mothe, J. and Tanguy, L. (2005). Linguistic features to predict query difficulty. In *Predicting Query Difficulty - Methods and Applications, SIGIR 2005*.

Newman, M. E. J. (2003). Ego-centered networks and the ripple effect. *Social Networks*, 25(1):83–95.

Oard, D. and Kim, J. (1998). Implicit Feedback for Recommender Systems. In *in Proceedings of the AAAI Workshop on Recommender Systems*, pages 81–83.

O'Connor, M. and Herlocker, J. (1999). Clustering Items for Collaborative Filtering. In *ACM SIGIR Workshop on Recommender Systems*.

O'Donovan, J. and Smyth, B. (2005). Trust in recommender systems. In *Proceedings of the 10th international conference on Intelligent user interfaces*, IUI '05, pages 167–174, New York, NY, USA. ACM.

O'Madadhain, J., Fisher, D., White, S., and Boey, Y. B. (2003). The JUNG (java universal Network/Graph) framework. Technical Report UCI-ICS 03-17, University of California.

Onuma, K., Tong, H., and Faloutsos, C. (2009). TANGENT: a novel, 'Surprise me', recommendation algorithm. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 657–666, New York, NY, USA. ACM.

Parra, D. and Amatriain, X. (2011). Walk the Talk. In Konstan, J. A., Conejo, R., Marzo, J. L., and Oliver, N., editors, *User Modeling, Adaption and Personalization*, volume 6787 of *Lecture Notes in Computer Science*, pages 255–268, Berlin, Heidelberg. Springer Berlin / Heidelberg.

Pavlov, D., Manavoglu, E., Pennock, D. M., and Giles, C. L. (2004). Collaborative Filtering with Maximum Entropy. *IEEE Intelligent Systems*, 19(6):40–48.

Pazzani, M. and Billsus, D. (1997). Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Machine Learning*, 27(3):313–331.

Pazzani, M. and Billsus, D. (2007). Content-Based Recommendation Systems The Adaptive Web. In Brusilovsky, P., Kobsa, A., and Nejdl, W., editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, chapter 10, pages 325–341. Springer Berlin / Heidelberg, Berlin, Heidelberg.

Pazzani, M. J. (1999). A Framework for Collaborative, Content-Based and Demographic Filtering. *Artificial Intelligence Review*, 13(5):393–408.

Pazzani, M. J., Muramatsu, J., and Billsus, D. (1996). Syskill & webert: Identifying interesting web sites. In Clancey, W. J., Weld, D. S., Clancey, W. J., and Weld, D. S., editors, *AAAI/IAAI, Vol. 1*, pages 54–61. AAAI Press / The MIT Press.

Pérez Iglesias, J. (2012). *Predicción del rendimiento de consultas basado en rankings de documentos y nuevo marco de evaluación*. PhD thesis, Universidad Nacional de Educación a Distancia.

Pérez-Iglesias, J. and Araujo, L. (2009). Ranking List Dispersion as a Query Performance Predictor. pages 371–374.

Pérez-Iglesias, J. and Araujo, L. (2010). Evaluation of Query Performance Prediction Methods by Range. In Chavez, E. and Lonardi, S., editors, *String Processing and Information Retrieval*, volume 6393 of *Lecture Notes in Computer Science*, chapter 23, pages 225–236–236. Springer Berlin / Heidelberg, Berlin, Heidelberg.

Plachouras, V., He, B., and Ounis, I. (2004). University of Glasgow at TREC 2004: Experiments in Web, Robust, and Terabyte Tracks with Terrier. In Voorhees, E. M., Buckland, L. P., Voorhees, E. M., and Buckland, L. P., editors, *TREC*, volume Special Publication 500-261. National Institute of Standards and Technology (NIST).

Plachouras, V., Ounis, I., van Rijsbergen, C. J., and Cacheda, F. (2003). University of Glasgow at the Web Track: Dynamic Application of Hyperlink Analysis using the Query Scope. In *TREC*, pages 646–652.

Ponte, J. M. and Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '98, pages 275–281, New York, NY, USA. ACM.

Pradel, B., Usunier, N., and Gallinari, P. (2012). Ranking with non-random missing ratings: influence of popularity and positivity on evaluation metrics. In *Proceedings of the sixth ACM conference on Recommender systems*, RecSys '12, pages 147–154, New York, NY, USA. ACM.

Pu, P., Chen, L., and Hu, R. (2012). Evaluating recommender systems from the user's perspective: survey of the state of the art. *User Modeling and User-Adapted Interaction*, 22(4):317–355.

Radlinski, F., Bennett, P. N., Carterette, B., and Joachims, T. (2009). Redundancy, diversity and interdependent document relevance. *SIGIR Forum*, 43(2):46–52.

Radlinski, F., Kleinberg, R., and Joachims, T. (2008). Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 784–791, New York, NY, USA. ACM.

Rafiei, D., Bharat, K., and Shukla, A. (2010). Diversifying web search results. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 781–790, New York, NY, USA. ACM.

Rafter, R., O'Mahony, M., Hurley, N., and Smyth, B. (2009). What Have the Neighbours Ever Done for Us? A Collaborative Filtering Perspective. In Houben, G.-J., McCalla, G., Pianesi, F., and Zancanaro, M., editors, *User Modeling, Adaptation, and Personalization*, volume 5535 of *Lecture Notes in Computer Science*, chapter 36, pages 355–360. Springer Berlin / Heidelberg, Berlin, Heidelberg.

Renda, E. M. and Straccia, U. (2003). Web metasearch: rank vs. score based rank aggregation methods. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 841–846, New York, NY, USA. ACM Press.

Rennie, J. D. M. and Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 713–719, New York, NY, USA. ACM.

Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P., and Riedl, J. (1994). GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina. ACM.

Ricci, F., Rokach, L., and Shapira, B. (2011). Introduction to recommender systems handbook. In Ricci, F., Rokach, L., Shapira, B., Kantor, P. B., Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, chapter 1, pages 1–35. Springer, Boston, MA.

Rich, E. (1979). User modeling via stereotypes. *Cognitive Science*, 3:335–366.

Roelleke, T. and Wang, J. (2008). TF-IDF uncovered: a study of theories and probabilities. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 435–442, New York, NY, USA. ACM.

Rojsattarat, E. and Soonthornphisaj, N. (2003). Hybrid Recommendation: Combining Content-Based Prediction and Collaborative Filtering. In *Intelligent Data Engineering and Automated Learning*, pages 337–344.

Said, A., Berkovsky, S., and De Luca, E. W. (2010). Putting things in context: Challenge on Context-Aware movie recommendation. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, CAMRa '10, pages 2–6, New York, NY, USA. ACM.

Said, A., Berkovsky, S., De Luca, E. W., and Hermanns, J. (2011). Challenge on context-aware movie recommendation: CAMRa2011. In *Proceedings of the fifth ACM*

conference on Recommender systems, RecSys '11, pages 385–386, New York, NY, USA. ACM.

Salakhutdinov, R., Mnih, A., and Hinton, G. E. (2007). Restricted boltzmann machines for collaborative filtering. In Ghahramani, Z. and Ghahramani, Z., editors, *ICML*, volume 227 of *ACM International Conference Proceeding Series*, pages 791–798, New York, NY, USA. ACM.

Salter, J. and Antonopoulos, N. (2006). CinemaScreen Recommender Agent: Combining Collaborative and Content-Based Filtering. *IEEE Intelligent Systems*, 21(1):35–41.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA. ACM.

Schein, A., Popescul, A., Ungar, L., and Pennock, D. (2002). Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002)*, pages 253–260.

Schein, A. I., Popescul, A., Ungar, L. H., and Pennock, D. M. (2001). Generative models for cold-start recommendations. In *Proceedings of the 2001 SIGIR Workshop on Recommender Systems*.

Semeraro, G., Degemmis, M., Lops, P., and Basile, P. (2007). Combining learning and word sense disambiguation for intelligent user profiling. In Veloso, M. M. and Veloso, M. M., editors, *IJCAI*, pages 2856–2861.

Shani, G. and Gunawardana, A. (2011). Evaluating Recommendation Systems. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, chapter 8, pages 257–297. Springer US, Boston, MA.

Shardanand, U. and Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth". In Katz, I. R., Mack, R. L., Marks, L., Rosson, M. B., Nielsen, J., Katz, I. R., Mack, R. L., Marks, L., Rosson, M. B., and Nielsen, J., editors, *CHI*, CHI '95, pages 210–217, New York, NY, USA. ACM/Addison-Wesley.

Shepitsen, A., Gemmell, J., Mobasher, B., and Burke, R. (2008). Personalized Recommendation in Social Tagging Systems using Hierarchical Clustering. In *Proceedings of the 2008 ACM Conference on Recommender Systems (RecSys 2008)*, RecSys '08, pages 259–266, New York, NY, USA. ACM.

Shtok, A., Kurland, O., and Carmel, D. (2009). Predicting query performance by Query-Drift estimation. In *Advances in Information Retrieval Theory*, pages 305–312.

Shtok, A., Kurland, O., and Carmel, D. (2010). Using statistical decision theory and relevance models for query-performance prediction. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 259–266, New York, NY, USA. ACM.

Snedecor, G. W. and Cochran, W. G. (1989). *Statistical Methods*. Iowa State University Press, 8 edition.

Soboroff, I. (2004). On evaluating web search with very few relevant documents. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 530–531, New York, NY, USA. ACM.

Steck, H. (2011). Item popularity and recommendation accuracy. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 125–132, New York, NY, USA. ACM.

Su, L. T. (1992). Evaluation measures for interactive information retrieval. *Information Processing & Management*, 28(4):503–516.

Sun, A. and Bhowmick, S. S. (2009). Image tag clarity: in search of visual-representative tags for social images. In *Proceedings of the first SIGMM workshop on Social media*, WSM '09, pages 19–26, New York, NY, USA. ACM.

Sun, A. and Datta, A. (2009). On stability, clarity, and co-occurrence of Self-Tagging. In Baeza Yates, R. A., Boldi, P., Ribeiro Neto, B. A., Cambazoglu, B. B., Baeza Yates, R. A., Boldi, P., Ribeiro Neto, B. A., and Cambazoglu, B. B., editors, *WSDM (Late Breaking-Results)*. ACM.

Teevan, J., Dumais, S. T., and Liebling, D. J. (2008). To personalize or not to personalize: modeling queries with variation in user intent. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 163–170, New York, NY, USA. ACM.

Tomlinson, S. (2005). European ad hoc retrieval experiments with hummingbird SearchServer TM at. In *CLEF 2005. Working Notes for the CLEF 2005 Workshop*.

Townsend, S. C., Zhou, Y., and Croft, B. W. (2004). A framework for selective query expansion. In Grossman, D., Gravano, L., Zhai, C., Herzog, O., Evans, D. A., Grossman, D., Gravano, L., Zhai, C., Herzog, O., and Evans, D. A., editors, *CIKM*, pages 236–237. ACM.

van Rijsbergen, C. J. (1989). Towards an information logic. *SIGIR Forum*, 23(SI):77–86.

van Setten, M. (2005). *Supporting people in finding information: hybrid recommender systems and goal-based structuring*. PhD thesis, University of Twente, Enschede.

Vargas, S. and Castells, P. (2011). Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 109–116, New York, NY, USA. ACM.

Vinay, V., Cox, I. J., Milic-Frayling, N., and Wood, K. (2006). On ranking the effectiveness of searches. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 398–404, New York, NY, USA. ACM Press.

Voorhees, E. M. (2002a). Overview of trec 2002. In *TREC*.

Voorhees, E. M. (2002b). The philosophy of information retrieval evaluation evaluation of Cross-Language information retrieval systems. In Peters, C., Braschler, M., Gonzalo, J., and Kluck, M., editors, *Evaluation of Cross-Language Information Retrieval Systems*, volume 2406 of *Lecture Notes in Computer Science*, chapter 34, pages 143–170. Springer Berlin / Heidelberg, Berlin, Heidelberg.

Voorhees, E. M. (2005a). Overview of the TREC 2004 robust retrieval track. In *Proceedings of the Thirteenth Text REtrieval Conference, TREC 2004*, pages 70–79.

Voorhees, E. M. (2005b). The TREC robust retrieval track. *SIGIR Forum*, 39(1):11–20.

Voorhees, E. M. (2006). The TREC 2005 robust track. *SIGIR Forum*, 40(1):41–48.

Voorhees, E. M. and Harman, D. K., editors (2005). *TREC: Experiment And Evaluation in Information Retrieval*. MIT Press, Cambridge, MA.

Walter, F. E., Battiston, S., and Schweitzer, F. (2009). Personalised and dynamic trust in social networks. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 197–204, New York, NY, USA. ACM.

Wang, J. (2009). Language Models of Collaborative Filtering. In Lee, G. G., Song, D., Lin, C.-Y., Aizawa, A., Kuriyama, K., Yoshioka, M., and Sakai, T., editors, *Information Retrieval Technology*, volume 5839, chapter 19, pages 218–229. Springer Berlin Heidelberg, Berlin, Heidelberg.

Wang, J., de Vries, A., and Reinders, M. (2006a). A User-Item Relevance Model for Log-Based Collaborative Filtering. In Lalmas, M., MacFarlane, A., Rüger, S., Tombros, A., Tsikrika, T., and Yavlinsky, A., editors, *Advances in Information Retrieval*, volume 3936 of *Lecture Notes in Computer Science*, chapter 5, pages 37–48–48. Springer Berlin / Heidelberg, Berlin, Heidelberg.

Wang, J., de Vries, A. P., and Reinders, M. J. T. (2006b). Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 501–508, New York, NY, USA. ACM.

Wang, J., de Vries, A. P., and Reinders, M. J. T. (2008a). Unified relevance models for rating prediction in collaborative filtering. *ACM Trans. Inf. Syst.*, 26(3):1–42.

Wang, J., Robertson, S., de Vries, A., and Reinders, M. (2008b). Probabilistic relevance ranking for collaborative filtering. *Information Retrieval*, 11(6):477–497.

Wang, X., Fang, H., and Zhai, C. (2008c). A study of methods for negative relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 219–226, New York, NY, USA. ACM.

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442.

Weng, J., Miao, C., and Goh, A. (2006). Improving collaborative filtering with trust-based metrics. In Haddad, H. and Haddad, H., editors, *SAC*, pages 1860–1864, New York, NY, USA. ACM.

Weng, L. T., Xu, Y., Li, Y., and Nayak, R. (2007). Improving Recommendation Novelty Based on Topic Taxonomy. *Web Intelligence and Intelligent Agent Technology, International Conference on*, 0:115–118.

Xiang, L., Yuan, Q., Zhao, S., Chen, L., Zhang, X., Yang, Q., and Sun, J. (2010). Temporal recommendation on graphs via long- and short-term preference fusion. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 723–732, New York, NY, USA. ACM.

Xin, Y. and Steck, H. (2011). Multi-value probabilistic matrix factorization for IP-TV recommendations. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 221–228, New York, NY, USA. ACM.

Xue, G. R., Lin, C., Yang, Q., Xi, W., Zeng, H. J., Yu, Y., and Chen, Z. (2005). Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 114–121, New York, NY, USA. ACM.

Yao, Y. Y. (1995). Measuring retrieval effectiveness based on user preference of documents. *JASIS*, 46(2):133–145.

Yilmaz, E., Aslam, J. A., and Robertson, S. (2008). A new rank correlation coefficient for information retrieval. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 587–594, New York, NY, USA. ACM.

Yom-Tov, E., Fine, S., Carmel, D., and Darlow, A. (2005a). Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 512–519, New York, NY, USA. ACM.

Yom-Tov, E., Fine, S., Carmel, D., and Darlow, A. (2005b). Metasearch and Federation using Query Difficulty Prediction. In *Predicting Query Difficulty - Methods and Applications, SIGIR 2005*.

Yu, K., Schwaighofer, A., Tresp, V., Ma, W. Y., and Zhang, H. (2003). Collaborative Ensemble Learning: Combining Collaborative and Content-Based Information Filtering via Hierarchical Bayes. In Meek, C., Kj, U., Meek, C., and Kj, U., editors, *UAI*, pages 616–623. Morgan Kaufmann.

Zar, J. H. (1972). Significance testing of the spearman rank correlation coefficient. *Journal of the American Statistical Association*, 67(339):578–580.

Zhang, M. and Hurley, N. (2008). Avoiding monotony: improving the diversity of recommendation lists. In *Proceedings of the 2008 ACM conference on Recommender systems*, RecSys '08, pages 123–130, New York, NY, USA. ACM.

Zhang, M. and Hurley, N. (2009). Statistical Modeling of Diversity in Top-N Recommender Systems. *Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*, 1:490–497.

Zhang, Y., Callan, J., and Minka, T. (2002). Novelty and redundancy detection in adaptive filtering. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 81–88, New York, NY, USA. ACM.

Zhao, Y., Scholer, F., and Tsegay, Y. (2008). Effective Pre-retrieval Query Performance Prediction Using Similarity and Variability Evidence. In Macdonald, C., Ounis, I., Plachouras, V., Ruthven, I., and White, R. W., editors, *Advances in Information Retrieval*, volume 4956 of *Lecture Notes in Computer Science*, chapter 8, pages 52–64. Springer Berlin Heidelberg, Berlin, Heidelberg.

Zhou, T., Kuscsik, Z., Liu, J.-G., Medo, M., Wakeling, J. R., and Zhang, Y.-C. (2010). Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, 107(10):4511–4515.

Zhou, Y. (2007). *Retrieval Performance Prediction and Document Quality*. PhD thesis, University of Massachusetts.

Zhou, Y. and Croft, W. B. (2006). Ranking robustness: a novel framework to predict query performance. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, CIKM '06, pages 567–574, New York, NY, USA. ACM.

Zhou, Y. and Croft, W. B. (2007). Query performance prediction in web search environments. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 543–550, New York, NY, USA. ACM.

Ziegler, C. N. and Lausen, G. (2004). Analyzing correlation between trust and user similarity in online communities. In Jensen, C. D., Poslad, S., Dimitrakos, T., Jensen, C. D., Poslad, S., and Dimitrakos, T., editors, *iTrust*, volume 2995 of *Lecture Notes in Computer Science*, pages 251–265. Springer.

Zimdars, A., Chickering, D. M., and Meek, C. (2001). Using Temporal Data for Making Recommendations. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, UAI '01, pages 580–588, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Zitnick, C. L. and Kanade, T. (2004). Maximum entropy for collaborative filtering. In Chickering, D. M., Halpern, J. Y., Chickering, D. M., and Halpern, J. Y., editors, *UAI*. AUAI Press.