UNIVERSIDADE DA CORUÑA

Information Retrieval Models for Recommender Systems

Doctoral Thesis

Daniel Valcarce Silva

2019

Daniel Valcarce: *Information Retrieval Models for Recommender Systems*, Doctoral Thesis, Universidade da Coruña, 2019.

Information Retrieval Models for Recommender Systems

Daniel Valcarce

Doctoral Thesis / 2019

Advisors:

Álvaro Barreiro

Javier Parapar

PhD in Computer Science

UNIVERSIDADE DA CORUÑA

Álvaro Barreiro García, Professor at the Department of Computer Science of Universidade da Coruña,

and

Javier Parapar López, Assistant Professor at the Department of Computer Science of Universidade da Coruña,

HEREBY CERTIFY

that the present Doctoral Thesis entitled ***Information Retrieval Models for Recommender Systems***, submitted to the Universidade da Coruña by Daniel Valcarce Silva, has been carried out under our supervision and fulfills all the requirements for the award of the degree of *PhD in Computer Science with International Mention*.

Álvaro Barreiro García
Advisor

Javier Parapar López
Advisor

To my mom,
infinite source of love.

To Julián,
who continuously inspires me.

We are reaching the stage where the problems we must solve are going to become insoluble without computers. I do not fear computers. I fear the lack of them.

— Isaac Asimov

Computers are good at following instructions, but not at reading your mind.

— Donald E. Knuth,
The TeXbook

ACKNOWLEDGMENTS

A PhD is a long journey, sometimes fraught with uncertainty and frustrations and sometimes extraordinarily rewarding. Without the brilliant guidance and mentorship of my advisors, I would not have been able to get through it. I wish to thank Álvaro Barreiro for his extremely valuable supervision during the course of this thesis. His wise advice from experience helped me on countless occasions. I also wish to thank Javier Parapar for being the most devoted advisor anyone can ask for. He always managed to motivate me to give my best. Thank you wholeheartedly.

I will never forget my dear colleagues (now friends) of the Information Retrieval Lab who were the most incredible people I could surround myself with. Edu, Martín, Alfonso, Gilberto, Feli, Conchi, Ana, Pedro and many more, I owe you a great deal of gratitude. Likewise, I also want to thank David and Josito for their priceless encouragement and support.

During this trip, I was lucky enough to visit other teams that helped me become a better researcher. I keep fond memories of all the members of the HPC Lab of ISTI-CNR in Pisa. Thank you, Raffaele, Franco, Vinicius, Matteo, Cristina, Salvo, Chiara, Nicola, Rossano and the rest of the group for your warm reception. I am also really grateful to all the members of the IRG of the UAM in Madrid and especially to Pablo, Alex, Javi and Rocío. The time I spent with you was enjoyable and helped me grow professionally. Finally, I wish to thank the whole MRI team at Google Münich for the wonderful opportunity they offered me. A very special thank you to Ronald, for hosting me, and to Daniele, Sebastian and Anton for their great guidance throughout my internship.

On a personal note, I would like to thank my friends who have been by my side all along this journey. I am grateful for their understanding and the good times we have shared. And, above all, I am deeply indebted to my beloved ones. To my parents, without their encouragement and care, I would not be here writing these words. To the rest of my family, for their constant support and unconditional love. And to Julián, who has taught me the true meaning of the word *love*.

Last, I would like to acknowledge the efforts of the PhD defense committee and the external reviewers as well as the anonymous reviewers of all my papers. I also acknowledge the financial support of the Government of Spain through grant FPU014/01724.

ABSTRACT

Information retrieval addresses the information needs of users by delivering relevant pieces of information but requires users to convey their information needs explicitly. In contrast, recommender systems offer personalized suggestions of items automatically. Ultimately, both fields help users cope with information overload by providing them with relevant items of information.

This thesis aims to explore the connections between information retrieval and recommender systems. Our objective is to devise recommendation models inspired in information retrieval techniques. We begin by borrowing ideas from the information retrieval evaluation literature to analyze evaluation metrics in recommender systems. Second, we study the applicability of pseudo-relevance feedback models to different recommendation tasks. We investigate the conventional top-N recommendation task, but we also explore the recently formulated user-item group formation problem and propose a novel task based on the liquidation of long tail items. Third, we exploit ad hoc retrieval models to compute neighborhoods in a collaborative filtering scenario. Fourth, we explore the opposite direction by adapting an effective recommendation framework to pseudo-relevance feedback. Finally, we discuss the results and present our conclusions.

In summary, this doctoral thesis adapts a series of information retrieval models to recommender systems. Our investigation shows that many retrieval models can be accommodated to deal with different recommendation tasks. Moreover, we find that taking the opposite path is also possible. Exhaustive experimentation confirms that the proposed models are competitive. Finally, we also perform a theoretical analysis of some models to explain their effectiveness.

RESUMEN

La recuperación de información da respuesta a las necesidades de información de los usuarios proporcionando información relevante, pero requiere que los usuarios expresen explícitamente sus necesidades de información. Por el contrario, los sistemas de recomendación ofrecen sugerencias personalizadas de elementos automáticamente. En última instancia, ambos campos ayudan a los usuarios a lidiar con la sobrecarga de información al proporcionarles información relevante.

Esta tesis tiene como propósito explorar las conexiones entre la recuperación de información y los sistemas de recomendación. Nuestro objetivo es diseñar modelos de recomendación inspirados en técnicas de recuperación de información. Comenzamos tomando prestadas ideas de la literatura de evaluación en recuperación de información para analizar las métricas de evaluación en los sistemas de recomendación. En segundo lugar, estudiamos la aplicabilidad de los modelos de retroalimentación de pseudo-relevancia a diferentes tareas de recomendación. Investigamos la tarea de recomendar listas ordenadas de elementos, pero también exploramos el problema recientemente formulado de formación de grupos usuario-elemento y proponemos una tarea novedosa basada en la liquidación de los elementos de la larga cola. Tercero, explotamos modelos de recuperación ad hoc para calcular vecindarios en un escenario de filtrado colaborativo. En cuarto lugar, exploramos la dirección opuesta adaptando un método eficaz de recomendación a la retroalimentación de pseudo-relevancia. Finalmente, discutimos los resultados y presentamos nuestras conclusiones.

En resumen, esta tesis doctoral adapta varios modelos de recuperación de información para su uso como sistemas de recomendación. Nuestra investigación muestra que muchos modelos de recuperación de información se pueden aplicar para tratar diferentes tareas de recomendación. Además, comprobamos que tomar el camino contrario también es posible. Una experimentación exhaustiva confirma que los modelos propuestos son competitivos. Finalmente, también realizamos un análisis teórico de algunos modelos para explicar su efectividad.

RESUMO

A recuperación de información dá resposta ás necesidades de información dos usuarios proporcionando información relevante, pero require que os usuarios expresen explicitamente as súas necesidades de información. Pola contra, os sistemas de recomendación ofrecen suxestións personalizadas de elementos automaticamente. En última instancia, ambos os campos axudan aos usuarios a lidar coa sobrecarga de información ao proporcionarlles información relevante.

Esta tese ten como propósito explorar as conexións entre a recuperación de información e os sistemas de recomendación. O noso obxectivo é deseñar modelos de recomendación inspirados en técnicas de recuperación de información. Comezamos tomando prestadas ideas da literatura de avaliación en recuperación de información para analizar as métricas de avaliación nos sistemas de recomendación. En segundo lugar, estudamos a aplicabilidade dos modelos de retroalimentación de seudo-relevancia a diferentes tarefas de recomendación. Investigamos a tarefa de recomendar listas ordenadas de elementos, pero tamén exploramos o problema recentemente formulado de formación de grupos de usuario-elemento e propoñemos unha tarefa nova baseada na liquidación dos elementos da longa cola. Terceiro, explotamos modelos de recuperación ad hoc para calcular veciñanzas nun escenario de filtrado colaborativo. En cuarto lugar, exploramos a dirección oposta adaptando un método eficaz de recomendación á retroalimentación de seudo-relevancia. Finalmente, discutimos os resultados e presentamos as nosas conclusións.

En resumo, esta tese doutoral adapta varios modelos de recuperación de información para o seu uso como sistemas de recomendación. A nosa investigación mostra que moitos modelos de recuperación de información pódense aplicar para tratar diferentes tarefas de recomendación. Ademais, comprobamos que tomar o camiño contrario tamén é posible. Unha experimentación exhaustiva confirma que os modelos propostos son competitivos. Finalmente, tamén realizamos unha análise teórica dalgúns modelos para explicar a súa efectividade.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ADS	Absolute Discounting Smoothing
AS	Additive Smoothing
AP	Average Precision
BPRMF	Bayesian Probabilistic Ranking Matrix Factorization
CHI2	Chi-Square
CF	Collaborative Filtering
DLiMe	Document Linear Method
DMM	Divergence Minimization Model
DP	Discriminative Power
DPS	Dirichlet Priors Smoothing
HT	Hitting Time
IB	Item-Based
IDF	Inverse Document Frequency
IF	Information Filtering
InfAP	Inferred Average Precision
IR	Information Retrieval
IRM2	Item Relevance Model 2
JMS	Jelinek-Mercer Smoothing
KLD	Kullback-Leibler Distance
*k*NN	*k* Nearest Neighbors
LiMe	Linear Method
LM	Language Model

LDA	Latent Dirichlet Allocation
LSA	Latent Semantic Analysis
LBSN	Location-Based Social Network
MAP	Mean Average Precision
MRR	Mean Reciprocal Rank
MEDMM	Maximum-Entropy Divergence Minimization Model
MF	Matrix Factorization
MLE	Maximum Likelihood Estimate
MSI	Mean Self-Information
NC	Normalized Cut
nDCG	Normalized Discounted Cumulative Gain
NeuMF	Neural Matrix Factorization
NMF	Non-negative Matrix Factorization
NMLE	Normalized Maximum Likelihood Estimate
NNCosNgbr	Non-Normalized Cosine Neighborhood
PPC	Posterior Probability Clustering
PLSA	Probabilistic Latent Semantic Analysis
PRF	Pseudo-Relevance Feedback
PRIN	Probabilistic Recommender with Item priors and Neural models
PRP	Probability Ranking Principle
PureSVD	Pure Singular Value Decomposition
QE	Query Expansion
RFMF	Relevance Feedback Matrix Factorization
RM	Relevance Model
RM1	Relevance Model 1

RM2	Relevance Model 2
RM3	Relevance Model 3
RS	Recommender Systems
RI	Reliability of Improvement
RSV	Robertson Selection Value
RW	Rocchio Weights
SLIM	Sparse Linear Method
SVD	Singular Value Decomposition
TF	Term Frequency
TLiMe	Term Linear Method
TREC	Text Retrieval Conference
UB	User-Based
UIGF	User-Item Group Formation
UIR	User Item Relevance
VSM	Vector Space Model
WRMF	Weighted Regularized Matrix Factorization
WSR	Weighted Sum Recommender

Part I

PRELIMINARIES

This project is experimental and of course comes without any warranty whatsoever. However, it could start a revolution in information access.

— Tim Berners-Lee,
WorldWideWeb wide-area hypertext app available

INTRODUCTION

Human history has been shaped by how we have managed information. As societies emerged, the development of writing, more than five millennia ago, was the first milestone. We started to compile information that had only been spread by word of mouth. Driven by pragmatic needs, we continued to create new forms of storing and processing information. Libraries, the primary places where written information was stored and preserved, flourished. Nevertheless, the amount of data was minimal because of the high cost of handwriting. However, the invention of the printing press multiplied the rate of growth of written information exponentially. More recently, the invention of the computer led to crucial modern developments such as the design of specialized data structures for querying digital libraries. Finally, the advent of the World Wide Web in 1989 triggered an unprecedented explosion in the availability of information. The creation of Tim Berners-Lee has become a universal repository of human knowledge that transformed how we access information.

Information technologies evolve increasingly faster: writing (ca. 3000 BC), the printing press (ca. 1440), the first computers (ca. 1930) and the World Wide Web (1989).

Information retrieval (IR) and information filtering (IF) are two fields of study that revolve around information processing. The development of computing and communication technologies has boosted the importance of these fields. IR systems deal with the representation, storage and access of information. Their goal is to expose users to relevant pieces of information according to their needs (Baeza-Yates and Ribeiro-Neto 2011; Manning et al. 2008). On the other hand, IF systems aim to select items from an information stream that may be of interest to a given user (Hanani et al. 2001). Among the different types of information filters, recommender systems (RS) are arguably the most prominent nowadays. The goal of a recommender system is to generate personalized suggestions for items based on the interests of a user (Ricci et al. 2015).

This thesis leverages the similarities between information retrieval and recommender systems.

Since the final objective of IR and IF systems is to provide users with relevant information items, some authors consider both fields as two sides of the same coin (Belkin and Croft 1992). Nonetheless, despite the similarities between information retrieval and information filtering, there has been little research about applying classic IR techniques to recommender systems until recently (Bellogín et al. 2013a; Kallumadi et al. 2018; Parapar et al. 2013). In this doctoral thesis, we aim to bridge the gap between information retrieval and recommender systems even more by adapting several IR models to different recommendation tasks and by establishing new analogies between both fields.

1.1 MOTIVATION

RS is a younger field than IR, but it is growing at a breakneck pace.

Early work on information retrieval dates back to the 50's (Kent et al. 1955; Mooers 1951). Since then, the field has evolved tremendously. The first ACM Conference on Information Retrieval (SIGIR) was held in 1971 and nowadays comprises hundreds of attendees. Additionally, the Web has brought a new information access paradigm where search has become crucial. In contrast, recommender systems is a much younger field. This area emerged in the mid-1990s with the explosion of the World Wide Web and the apparition of e-commerce sites (Resnick and Varian 1997; Resnick et al. 1994; Shardanand and Maes 1995). Although the first ACM Conference on Recommender Systems (RecSys) took place in 2007, it has grown very quickly and nowadays attracts hundreds of attendees—many of them from industry.

As the Web provides increasing amounts of information, information systems have to face new challenges. The sheer volume of information available to the public is overwhelming. The difficulty to find and select relevant information increases as more and more content is available. Without proper tools to the deal with information overload, users may miss interesting information or consume uninteresting content. For this reason, information sciences such as information retrieval and information filtering are crucial in the current landscape.

Information retrieval systems are typically oriented towards producing rankings of relevant documents. Although recommender systems were initially oriented towards predicting ratings accurately, a paradigm shift has been brought towards producing a good ranking of items (typically known as top-N recommendation). Therefore, IR and modern RS techniques seem to have very similar objectives.

Recommender systems have become a pervasive technology to address the information overload problem. The enormous growth of data has radically changed the way we access information. Additionally, as information systems offer more advanced capabilities, users are becoming more and more demanding. In this challenging context, traditional search and browse features are not enough. Users expect proactive suggestions from the systems rather than specifying queries that convey their information needs.

The main difference between information retrieval and recommender systems lies in the representation of the information need: while an information retrieval system typically uses an explicit query prompted by the user, a recommender system exploits the user's history as an implicit query. Nevertheless, in the end, both fields share the same goal: providing users with access to relevant pieces of information.

1.2 AIM AND SCOPE

We aim to exploit the parallelisms between IR and RS tasks.

We believe that cross-pollination between information retrieval and recommender systems fields can lead to new, useful approaches. In this doctoral thesis, we go back to the roots of the RS field and explore its relationship with IR. Information retrieval has been around longer than recommender systems; therefore, we think that we can leverage existing work and knowledge developed by the IR community to propose new recommendation models or improve current recommendation methods. Nonetheless, we also think that we can draw inspiration from RS techniques to bring fresh air to consolidated IR tasks.

In this thesis, we restrict ourselves to two IR tasks: ad hoc retrieval and pseudo-relevance feedback.

In this doctoral thesis, we focus on the applicability of some information retrieval models to recommender systems. We limit the scope of this work to two main IR tasks: ad hoc retrieval and pseudo-relevance feedback. In particular, we explore the adaptation of ad hoc retrieval models for computing neighborhoods and pseudo-relevance feedback algorithms for different recommendation tasks. On the other hand, to close the circle, at the end of this thesis we also explore how linear methods used in recommendation can build effective pseudo-relevance feedback models.

Evaluation plays a crucial role in experimental sciences such as information retrieval and recommender systems. In this thesis, we assess the effectiveness and efficiency of the proposed model using offline evaluation. This approach usually constitutes the first step in evaluation due to its reduced costs and high reproducibility. In contrast, online evaluations require experiments with real users which are expensive and difficult to perform in the academy. For these reasons, we run our experiments on public datasets.

1.3 STRUCTURE AND CONTRIBUTIONS OF THE THESIS

This doctoral thesis is divided into six parts with thirteen chapters. The present chapter contains the introduction to this work. Chapter 2 introduces information retrieval and recommender systems concepts and relevant related work. Although a specialist on the topics may skip it, an interested reader may find it a useful introduction to both fields. Chapter 3 details the evaluation protocols and common experimental settings used throughout this research. Chapters 4 to 11 present the novel contributions of this thesis. Contribution chapters are meant to be as self-contained as possible. They can be read and understood with only the background information provided in Chapter 2. Chapter 12 presents the comparison and discussion of the findings of this thesis. Finally, Chapter 13 contains the conclusions and future work. Below we present the organization of the parts and chapters in more detail:

PART I The first part includes Chapter 1, which is the introduction to this thesis, and Chapter 2, which discusses the background work. The introduction presents the context and motivation of the thesis, the aim and scope of our work and the structure and contributions of the study. The background chapter, on the other hand, presents an overview of information retrieval and recommender systems and introduces the main concepts of both fields. Regarding information retrieval, we focus on ad hoc retrieval and pseudo-relevance feedback which we adapt in following chapters to recommendation tasks. We also present previous work that studies or exploits the relationship between IR and RS.

PART II This part of the thesis describes the research methods in two chapters. On the one hand, Chapter 3 describes the information retrieval and recommender systems evaluation methods used throughout this work. On the other hand, Chapter 4 contains a novel study of the robustness and discriminative power of rank-oriented metrics for recommendation. The findings of this study justify the evaluation metrics employed in this thesis.

PART III We present here the adaptation of several pseudo-relevance feedback models to different recommendation tasks. In particular, Chapter 5 improves an existing adaptation of relevance models to top-N recommendation by exploring smoothing techniques and prior probability estimators. Chapter 6 proposes a complementary item-based adaptation of relevance models which we use to solve a novel recommendation problem: how to liquidate long tail items. Chapter 7 employs the item-based adaptation of relevance models with personalized user prior estimators to address the item-driven group formation task. Lastly, we explore the adaptation of pseudo-relevance feedback techniques proposed within Rocchio framework to top-N recommendation in Chapter 8.

PART IV This part contains two chapters focused on neighborhood-based recommendation algorithms. Chapter 9 measures the margin for improvement of different techniques for computing neighborhoods. Based on these findings, we propose principled modifications of cosine similarity—similar to normalization schemes used in information retrieval. Chapter 10, on the other hand, proposes the adaptation of language models from ad hoc retrieval to neighborhood computation.

PART V In this part, we explore the opposite direction by adapting a recommendation technique to perform pseudo-relevance feedback. Particularly, Chapter 11 describes how sparse linear methods, successfully used in recommendation, can also be used to expand queries and improve retrieval effectiveness.

PART VI In the last part, Chapter 12 discusses the results obtained and compares them against the state of the art. Finally, Chapter 13 summarizes the contributions of this thesis, presents the conclusions and suggests future lines of research.

2

BACKGROUND

The first two sections of this chapter introduce the fundamental concepts and tasks of information retrieval (IR) and recommender systems (RS) fields. Nonetheless, we decided to make the contribution chapters as self-contained as possible by providing more advanced and specific background information in each chapter. Therefore, a reader experienced with the foundations of IR and RS may skip these two sections. Afterward, a third section reviews the existing literature that takes inspiration from previous work in information retrieval to develop new recommendation models as well as the other way around.

2.1 INFORMATION RETRIEVAL

Information retrieval is a computer science area that focuses on satisfying the information needs of the users (Baeza-Yates and Ribeiro-Neto 2011; Manning et al. 2008). Search engines are probably the most prominent example of information retrieval systems. More formally, we can use the following definition:

IR is sometimes referred to as the science of search *or the* science of finding.

> Information retrieval deals with the representation, storage, organization of, and access to information items such as documents, Web pages, online catalogs, structured and semi-structured records, multimedia objects. The representation and organization of the information items should be such as to provide the users with easy access to information of their interest. (Baeza-Yates and Ribeiro-Neto 2011)

The importance of information retrieval has exploded after the invention of the World Wide Web. The exponential growth in the volume of information has boosted the development of new IR methods to be able to meet the increasing information needs of the users.

Retrieval models are at the heart of the IR field.

Information retrieval systems such as search engines consist of several components such as crawlers, indexers, retrieval models and user interfaces. Among them, retrieval models are the core of these systems because they are responsible for producing search results. In this thesis, we take inspiration from these models to formulate novel recommendation techniques. In particular, we study models proposed to address two well-known IR tasks: ad hoc retrieval and pseudo-relevance feedback. Each is described below.

2.1.1 *Ad hoc retrieval*

Ad hoc retrieval is considered the core task in IR.

Ad hoc retrieval constitutes the most studied task in information retrieval (Manning et al. 2008). This task is performed on top of a collection of documents. Documents such as web pages, news articles, patents, images, videos or books constitute the retrieval units. This collection is previously indexed to create data structures called inverted indexes that allow efficient retrieval. Ad hoc retrieval consists in finding those documents in the collection that are relevant to the information need of the user. The user has an information need that conveys to the system in the form of a short textual description called *query*. The retrieval engine processes the query against the collection using a retrieval model and produces a ranked list of documents that is the output that the user receives. A document is considered relevant when users find it valuable in relation to their information need.

Ad hoc retrieval models must deal with document filtering and ranking. These models discard those documents in the collection that are not relevant to the user's information need and ranks the remaining ones by decreasing estimated relevance. They compare the document representation against the information need representation to do so. Retrieval models are one of the most fertile areas of research in IR. Therefore, different mathematical models have been proposed to rank documents according to a query. Some of the most influencing retrieval models have been the boolean model (Lancaster and Fayen 1973), the vector space model (Salton et al. 1975), the extended boolean model (Salton et al. 1983), Okapi BM25 (Spärck Jones et al. 2000a,b), probabilistic models such as the binary independence model (Robertson et al. 1976) or the language model (Ponte and Croft 1998; Zhai 2008), and more recent neural models (Mitra and Craswell 2018). Next, we discuss two of the most important ad hoc retrieval frameworks—the vector space model and the language model—that we exploit in later chapters.

2.1.1.1 *Vector space model*

The *vector space model* (VSM) constitutes one of the most standard ad hoc retrieval approaches (Salton et al. 1975). The VSM models queries and documents as sparse high-dimensional vectors of word frequencies. This representation is known as *bag-of-words* (Harris 1954).

We denote a document by a vector $\vec{D}$ with as many dimensions as terms in the collection. Each position represents the weight of a specific term in the document. Different weighting schemes have been proposed, but most of them rely on TF (term frequency) and IDF (inverse document frequency) heuristics. Likewise, queries are also represented by similar vectors. Retrieval is performed by computing the vector similarity between the query and each document in the collection. Cosine similarity (i. e., the cosine of the angle formed by two vectors) is the most common similarity metric in the VSM:

$$\operatorname{cosine}(\vec{Q}, \vec{D}) = \frac{\vec{Q} \cdot \vec{D}}{\|\vec{D}\| \, \|\vec{Q}\|} \tag{2.1}$$

Singhal et al. (1996) introduced a pivoted normalization into this similarity metric to improve its retrieval effectiveness. We follow a similar approach in Chapter 9 to modify cosine similarity to account for the length of the users profiles.

2.1.1.2 *Language models*

The introduction of probabilistic models represented a breakthrough in IR. They have been developed following the probability ranking principle (PRP). This principle states that documents should be ranked in descending order of probability of relevance (Maron and Kuhns 1960; Robertson 1977). The formal statement of the PRP is:

> If a reference retrieval system's response to each request is a ranking of the documents in the collection in order of decreasing probability of usefulness to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data has been made available to the system for this purpose, then the overall effectiveness of the system to its users will be the best that is obtainable on the basis of this data. (Robertson 1977)

Language models (LM) are a formal approach with a sound statistical foundation that follow the PRP (Zhai 2008). Although they do not explicitly include the concept of relevance, they can be derived from a generative relevance model (Lafferty and Zhai 2003). This framework models the occurrences of words in the documents and queries as a random generative process—usually, using a multinomial distribution. In this way, we can infer a language model for each document in the collection. To rank those documents according to a query, we estimate the posterior probability of each document D given the particular query Q:

$$p(D|Q) = \frac{p(Q|D)\,p(D)}{p(Q)} \overset{\text{rank}}{=} p(Q|D)\,p(D) \tag{2.2}$$

The query likelihood model is the first approach for using language models in IR.

where $p(Q|D)$ denotes the query likelihood and $p(D)$ the document prior. We can ignore the query prior $p(Q)$ because, for a given query, it has no effect in the ranking. If we take a uniform document prior, we just need to compute $p(Q|D)$ to rank documents according to a query. This is the *query likelihood model*: we assume that a query is a sample drawn from a language model θ_D. Therefore, to score a document, we only compute the likelihood of the query given the estimated document language model $p(Q|\theta_D)$. The most popular approach to compute the query likelihood is to use a unigram model:

$$\text{score}(Q, D) = p(Q|\theta_D) = \prod_{t \in Q} p(t|\theta_D)^{c(t,Q)} \tag{2.3}$$

where $c(t, Q)$ denotes the count of term t in the query Q. The conditional probability $p(t|\theta_D)$ is computed via the maximum likelihood estimate (MLE) of a multinomial distribution smoothed with a background model (Zhai and Lafferty 2004).

In Chapter 10, we establish a connection between ad hoc retrieval models and neighborhood computation in recommendation which allows us to adapt the query likelihood model to the computation of user and item neighborhoods.

2.1.2 *Pseudo-relevance feedback*

The retrieval model is crucial in the effectiveness of a search engine. Nevertheless, the quality of the search results can be improved without modifying the retrieval model. Users often find difficult to convey their information needs in the form of a query. Therefore, *query expansion* (QE), which consists in expanding the query with new terms, is an effective way to improve retrieval effectiveness (Carpineto and Romano 2012). If done carefully, the expanded query would likely provide better retrieval results than the initial one.

Among QE techniques, *relevance feedback* is one of the most reliable query expansion methods (Rocchio 1971; Ruthven and Lalmas 2003). Relevance feedback requires users to indicate which documents from those presented to them are relevant to their information needs. In this way, the feedback from the users is combined with the original query to generate an expanded query that usually yields a better retrieval ranking. However, obtaining information from the user is expensive and sometimes even infeasible. One alternative is to use the query logs of previous users in the system to infer which documents were relevant by their clicks.

Relevance feedback requires interaction from users.

Another alternative is to use automatic QE techniques that do not require feedback from the users (Carpineto and Romano 2012). Given the utility of these methods, it is not surprising that initial work on automatic query expansion dates from the sixties (Maron and Kuhns 1960). *Pseudo-relevance feedback* (PRF), also known as blind relevance feedback, is an automatic QE method whose foundations date back to the late seventies (Croft and Harper 1979), but it is still a hot research area in IR. In fact, empirical research has shown that PRF is an effective method to improve retrieval (Carpineto et al. 2001; Croft and Harper 1979; Lavrenko and Croft 2001; Lv and Zhai 2009, 2014; Parapar and Barreiro 2011; Zhai and Lafferty 2001).

Automatic QE techniques are appealing because they are transparent to users.

PRF does not require interaction from the users because it assumes that the top documents retrieved with the initial query are relevant. These documents that are assumed to be relevant form the so-called *pseudo-relevant set*. PRF techniques extract terms (with their corresponding weights) from this set to expand the original query. We then use the expanded query for a second retrieval and the results of this second ranking are the ones presented to the user. If the retrieval model provides decent results, this assumption is not too strong and the expanded query may provide better retrieval results than the original one.

Pseudo-relevance feedback assume that the top retrieved documents are relevant.

2.1.2.1 *PRF in the vector space model*

Rocchio (1971) framework was one of the very early successful query expansion methods presented in the context of the vector space model. Rocchio algorithm modifies the query vector in a direction which is closer to the centroid of the relevant documents vectors and further from the centroid of non-relevant documents vectors. We denote the set of relevant document $\mathcal{D}_r$ and the set of non-relevant document by $\mathcal{D}_{nr}$, the original query vector by $\vec{Q}$ and the expanded query vector by $\vec{Q}'$.

Rocchio framework is the most popular approach for query expansion in the VSM.

$$\vec{Q}' = \alpha \vec{Q} + \frac{\beta}{|\mathcal{D}_r|} \sum_{\vec{D} \in \mathcal{D}_r} \vec{D} - \frac{\gamma}{|\mathcal{D}_{nr}|} \sum_{\vec{D} \in \mathcal{D}_{nr}} \vec{D} \tag{2.4}$$

When performing pseudo-relevance feedback, we use the pseudo-relevant set F as an estimate of the set of relevant documents and we ignore the negative feedback. In this way, we obtain the following simplified equation:

$$\vec{Q}' = \vec{Q} + \frac{\alpha}{|F|} \sum_{\vec{D} \in F} \vec{D} \tag{2.5}$$

Carpineto and Romano (2012) and Carpineto et al. (2001) used the Rocchio framework with different term scoring functions to perform PRF. Among them, they used Rocchio weights (Rocchio 1971), the Robertson selection value (Robertson 1990), Chi-square (Carpineto et al. 2001) and Kullback-Leibler divergence methods (Carpineto et al. 2001).

2.1.2.2 *PRF based on language models*

The language modeling framework has been a fertile area of research for PRF techniques.

Among all the PRF techniques in the literature , those developed within the statistical language model framework (Ponte and Croft 1998; Zhai 2008) are arguably the most prominent ones because of their sound theoretical foundation and their empirical effectiveness (Lv and Zhai 2009). Relevance-based language models or, for short, relevance models (RM) are a PRF technique proposed by Lavrenko and Croft (2001) that explicitly introduces the concept of relevance in language models. On the other hand, Zhai and Lafferty (2001) proposed the divergence minimization model (DMM) and later Lv and Zhai (2014) extended it developing the maximum-entropy divergence minimization model (MEDMM). Both techniques build a model which is close to the language model of the documents of the pseudo-relevant set and far away from the background model. Additionally, other PRF approaches based on language models use mixture models such as the simple mixture model (Zhai and Lafferty 2001) and the regularized mixture model (Tao and Zhai 2006).

The KLD model introduces the query language model θ_Q in the retrieval equation.

The difficulty in adding feedback to the query likelihood model has led to the development of *Kullback-Leibler divergence* (KLD) retrieval model (Lafferty and Zhai 2001). This model computes the Kullback-Leibler divergence $D(\cdot\|\cdot)$ between the query and the document language models, θ_Q and θ_D, which is rank equivalent to the negative cross-entropy:

$$\text{score}(Q, D) = -D(\theta_Q \| \theta_D) \overset{\text{rank}}{=} \sum_{t \in \mathcal{V}} p(t|\theta_Q) \log p(t|\theta_D) \tag{2.6}$$

where $\mathcal{V}$ denotes the vocabulary of the collection.

To incorporate the feedback model in the retrieval formulation, instead of using the original query model θ_Q, we use θ'_Q which is the result of the interpolation between θ_Q and the estimated feedback model θ_F (Abdul-Jaleel et al. 2004; Lv and Zhai 2009):

$$p(t|\theta'_Q) = (1 - \alpha)\, p(t|\theta_Q) + \alpha\, p(t|\theta_F) \tag{2.7}$$

where $\alpha \in [0, 1]$ controls the relative importance of the feedback model with respect to the query model. Therefore, the task of a PRF technique under this framework is to provide an estimate of θ_F given the pseudo-relevant set F.

In PART III, we study the applicability of different pseudo-relevance feedback models in diverse recommendation tasks. We analyze the adaptation of relevance models to user-based recommendation proposed by Parapar et al. (2013) in Chapter 5. We also propose an item-based adaptation of relevance models to address the liquidation of long tail items and the user-item group formation problem in Chapters 6 and 7, respectively. Finally, in Chapter 8, we explore the adaptation of term scoring functions used within the Rocchio framework to top-N recommendation.

2.2 RECOMMENDER SYSTEMS

Recommender systems provide personalized items suggestions.

Recommender systems (RS) are tools designed to assist users by providing personalized item suggestions (Resnick and Varian 1997; Ricci et al. 2015). In a world with a growing amount of information, users are demanding more personalized information systems. They want to receive items suggestions instead of browsing and explicitly querying the system. In this landscape, recommender systems have become a pervasive technology to deal with these increasing information demands.

The Netflix Prize marked a turning point in the development of RS.

The Netflix Prize[1] stimulated the research in recommender systems. This open competition held by Netflix from 2006 to 2009 represented a major milestone in the field. The goal was to improve the accuracy of CineMatch, their recommender system at the time, by 10% (Bennett and Lanning 2007). This event led to an explosion of research on different recommendation models.

RS can have a vast impact in a variety of domains.

It is well known how recommender systems impact Web companies. For instance, Sharma et al. (2015) estimated that 30% of page views in Amazon come from recommendations. Zhou et al. (2010a) also estimated that YouTube related video recommendation accounts for about 30% of overall YouTube views. Similarly, Netflix claimed that more than 80% of movies plays originated from recommendations and placed the value of their recommender system at more than $1 billion per year (Gomez-Uribe

1 The official website of the competition is https://www.netflixprize.com.

and Hunt 2015). In view of these figures, it is not surprising that recommender systems have become indispensable tools for many technological companies.

In a recommendation scenario, we can distinguish two main elements: users and items. Users are those subjects who interact with the system and receive recommendations. The set $\mathcal{U}$ denotes the users in the system. Items, on the other hand, are those elements that users interact with and the system recommends. They form the set $\mathcal{I}$. Items can be very diverse in nature because recommender systems are used in many domains to suggest videos, news, e-commerce products, advertisements, songs or trips, to name a few.

Users, items and interactions are the primary elements of recommender systems.

Users and items are connected through interactions. Recommender systems build models from these interactions also known as *feedback*. We can distinguish between explicit and implicit feedback. Explicit feedback are those interactions that deliberately express the users' preferences such as ratings, likes or reviews. In contrast, user actions such as clicks, reproductions or purchases constitute implicit feedback. We can infer preferences from these actions, but the users are not directly communicating their preferences.

We denote the interaction of user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$ by $r(u, i)$. If the user u interacted with item i, $r(u, i)$ is equal to the value of the interaction such as the rating score or the number of purchases. Otherwise, $r(u, i) = 0$. Additionally, $\mathcal{I}_u$ denotes the set of items that user u interacted with. Likewise, $\mathcal{U}_i$ refers to the set of users that interacted with item i. The ranked list of n recommendations for user u is denoted by L_u^n and we refer to the k-th position of this list by $L_u^n[k]$.

User-item interactions are usually represented in the form of a matrix where rows play the role of users and columns represent the items. Additional information such as context or time can be included in additional dimensions, but context-aware recommender systems (Adomavicius and Tuzhilin 2015) and time-aware recommender systems (Campos et al. 2014) are out of the scope of this thesis.

2.2.1 *Recommendation tasks*

The classic recommendation problem consists in generating tailored item recommendations to the users. The typical output of a recommender system is a score $\hat{r}(u, i)$ for each user-item pair (u, i). Therefore, for a given user, we can compute the score for each item. However, recommender systems have evolved in recent years to address different personalized tasks.

In fact, in Chapters 6 and 7, we study less conventional recommendation tasks.

Rating predictors aim to forecast the rating that a user would give to an item.

One of the first approaches to model the classic recommendation problem was *rating prediction* problem. Rating predictors intend to forecast the ratings that users would give to each item (Gunawardana and Shani 2015; Herlocker et al. 2004). Therefore, these systems aim to output a value that is close to the real rating value: $\hat{r}(u, i) \approx r(u, i)$.

The rationale of modeling the recommendation problem as a rating prediction task is to recommend those items with the highest predicted rating. In this way, if the rating predictor is accurate, the user will be presented with a set of items with high predicted ratings. Nevertheless, this seemingly reasonable approach to recommendation does not lead to good recommenders in practice. The reason is that rating predictors aim to forecast the rating values of those items that the user has decided to rate (Steck 2013). These ratings are biased because ratings are *missing not at random* (Marlin et al. 2007; Steck 2010). However, rating predictors must be able to forecast the rating of any item if they aim to produce good rankings of items.

The top-N recommendation task consists in finding the N most relevant items for each user.

In a production environment, recommender systems usually present a short list of suggestions where the predicted rating values are not shown. This task is usually referred to as *top-N recommendation* (Cremonesi et al. 2010; Herlocker et al. 2004). Top-N recommenders focus on providing a list with good items, not on accurately predicting ratings. Moreover, rating prediction studies how well a system can predict the existing ratings while a top-N recommender only cares about the top relevant items for each user. All these reasons justify the paradigm shift from rating prediction to top-N recommendation.

The traditional top-N recommendation task can be formulated as finding a scoring function $s : \mathcal{U} \times \mathcal{I} \to \mathbb{R}$ such that, for each user u, we can generate a ranked list of n items $L_u^n \in \mathcal{I}^n$ sorted by decreasing score. When developing a top-N recommender, the score is only used to sort the recommendations and, thus, we do not care about the exact value, but the relative order of the items.

2.2.2 *Types of recommender systems*

We can distinguish multiple approaches to recommendation (Adomavicius and Tuzhilin 2005; Ricci et al. 2015). They are often classified in content-based filtering and collaborative filtering systems. Additionally, we also have a third category of hybrid systems that combine models of the previous two groups of recommender systems.

Content-based filtering leverages information about the items that the target user liked.

Content-based filtering or simply *content-based* approaches generate recommendations based on the user profile and the item descriptions: they exploit the information about the items that the user previously liked to suggest similar items (Gemmis et al. 2015). Content-based approaches require accurate and rich item metadata to be able to compute similarities between items.

Collaborative filtering exploits the wisdom of the crowd *to produce recommendations.*

In contrast, *collaborative filtering* (CF) approaches rely on the feedback of other users to produce recommendations (Koren and Bell 2015; Ning et al. 2015). CF approaches consider items as black boxes: items are modeled by their interactions to other users. Therefore, collaborative filtering methods do not require item descriptions. CF techniques are highly effective in scenarios with rich feedback composed of user-item interactions. Nowadays, collaborative filtering approaches tend to work well in several scenarios as they can leverage the increasing amount of information that is available.

Regarding hybrid systems, one effective approach is to combine multiple recommenders by applying metasearch techniques which we have done in previous work (Valcarce et al. 2017b). However, this doctoral thesis is devoted to collaborative filtering recommenders due to the advantage of CF in scenarios with a large and growing amount of information. For the sake of cohesion, we omit this article and focus on our collaborative filtering works. Next, we delve into the taxonomy of this type of recommender systems.

2.2.2.1 *Collaborative filtering recommenders*

Model-based CF recommenders build a predictive model from the data.

We can further classify collaborative filtering techniques in two groups: model-based and neighborhood-based approaches. On the one hand, *model-based* techniques learn a predictive model from the feedback of the users (Koren and Bell 2015). These methods require a training phase to build the model. Once created, this model is able to score user-item pairs.

Among the rich area of model-based recommenders, *matrix factorization* (MF) is arguably the most prominent collaborative filtering approach as it usually yields high-quality recommendations (Koren and Bell 2015; Koren et al. 2009).

Neighborhood-based approaches directly use the feedback of the users to make recommendations.

On the other hand, *neighborhood-based* (also known as memory-based systems) techniques directly use the user-item interactions to compute recommendations (Ning et al. 2015). These approaches usually employ similarities or distance metrics to build neighborhoods. The neighborhood of the user u is formed by similar users in the collection: $V_u = \{v \in$

$\mathcal{U} \mid v$ is similar to $u\}$. Likewise, the neighborhood of the item i comprises a set of similar items: $J_i = \{j \in \mathcal{I} \mid j$ is similar to $i\}$.

The most common practice to compute neighborhoods is the kNN algorithm (Ning et al. 2015). This technique consists in finding the k most similar users to the target user using a pairwise similarity metric such as Pearson's correlation coefficient or cosine similarity.

Neighborhood-based approaches can be categorized in user-based and item-based techniques. User-based systems recommend items that like-minded people enjoyed while item-based systems recommend items that are similar to those the user liked. Similarities are based on the user-item interactions; no information about the content is used in contrast to content-based approaches.

Although previous works showed that model-based approaches tend to perform better than neighborhood-based techniques (Cremonesi et al. 2010; Koren and Bell 2015; Koren et al. 2009; Rendle et al. 2009), model-based techniques are usually more complex since they involve training a model and tuning several hyperparameters. Additionally, those models are usually difficult to interpret. In fact, there have been some recent efforts to make model-based approaches more explainable (Abdollahi and Nasraoui 2016). In contrast, neighborhood-based models are straightforward and efficient, and their output is more easily explainable than the one from model-based recommenders (Ning et al. 2015).

2.2.3 *Recommender systems challenges*

Recommender systems have to face multiple challenges (Ricci et al. 2015). In this section, we explain some of the challenges addressed in this thesis.

Item popularity follows a heavy-tail distribution.

The *long tail* is a term coined by Anderson (2008) to refer to those less popular items that have a low demand in large catalogs. The popularity of items follows a power law distribution. Items in the long tail are very abundant, but user interactions with them are scarce. For these reasons, recommending long tail items is difficult for collaborative filtering models. Additionally, the lack of interactions with unpopular items may bias the recommendation model towards popular items. The long tail may also bias the evaluation. In fact, in Chapter 4, we study the effect of this bias in evaluation metrics.

New tasks continue to appear as RS become more sophisticated.

Recommender systems can be applied in many domains; however, they may require specific adaptations. Top-N recommendation is the most typical problem addressed by RS, but other less traditional tasks may be useful in certain domains. In Chapters 6 and 7, we present recommendation models that tackle unconventional recommendation tasks.

Scalability is critical when processing vast amounts of information.

Usually, the primary concern of the designer of a recommender system is to deliver useful recommendations, but, as the number of users and items increases, *scalability* becomes a crucial issue. Processing large amounts of data give the opportunity of satisfying the information needs of the users with better recommendations at the cost of a much heavier computation load. Particularly, collaborative filtering techniques are very effective in these situations where a lot of data is available. Scalability can be addressed from different perspectives. One way is by developing models with cheaper computational requirements. In Chapter 8, we propose alternative techniques to those studied in Chapter 5 that offer similar effectiveness but at a much lower computational cost. Another path to tackle scalability is by building distributed platforms and algorithms. During the doctoral program, we built a distributed platform for producing recommendations at a large scale (Valcarce et al. 2014, 2015a) and we also developed a distributed implementation of a recommendation algorithm (Valcarce et al. 2018a). Nevertheless, for the sake of simplicity and cohesion, we have decided to leave these works out of the scope of this document.

2.3 BRIDGING THE GAP BETWEEN IR AND RS

Information retrieval tasks consist in providing the information that users demand (Baeza-Yates and Ribeiro-Neto 2011; Manning et al. 2008) while information filtering focuses on selecting relevant pieces of information from a stream of data (Hanani et al. 2001). We can distinguish between passive and active information filters. A passive IF system aim to remove unwanted pieces of information. For example, anti-spam techniques filter out unwanted messages and keep useful communications. On the other hand, active IF systems push relevant information to the users. Recommender systems are probably the most prominent type of active information filters. These systems deliver suggestions to users based on their past behavior. Therefore, they actively push information to the users instead of just filtering it out.

IR usually respond to explicit demands while RS provide proactive suggestions.

Information retrieval and information filtering are two fields of the information sciences. Since they share the same ultimate goal—deliver relevant information—some authors considered them to be sibling fields. Belkin and Croft (1992) even called them *two sides of the same coin*. We argue that the main difference between an IR system and a recommender system lies mainly in the representation of the information need. A typical IR system begins with an explicit query provided by the user while a recommender system actively exploits the user's profile (Valcarce 2015). Therefore, in recommendation, the query is implicit and the recommender must infer it.

As described in Section 2.2.1, rating prediction does not model the recommendation task effectively. Instead, users expect a short ranking of relevant items. An advantage of IR models is that they are traditionally focused on generating a ranked list of documents. Therefore, they can be naturally adapted to top-N recommendation. In fact, we can find in the recent recommender systems literature several works that propose recommendation models inspired in information retrieval ideas. We next review literature on adapting IR models to recommendation and the other way around.

2.3.1 *Ad hoc retrieval models*

Breese et al. (1998) studied different collaborative filtering techniques. Among them, they used the cosine formula from the IR vector space model to compute user similarities. A more general framework was presented by Bellogín et al. (2013a). This framework adapts any ad hoc retrieval model as a memory-based collaborative filtering algorithm. They showed that a standard search engine could be used to generate recommendations for the top-N recommendation task.

Wang et al. (2006) introduced a user-based and an item-based collaborative filtering algorithms for implicit feedback. Following the generative language modeling approach of IR, Wang (2009) also proposed two item scoring functions and a risk-averse model that penalizes less reliable scores for implicit feedback. Regarding explicit feedback, Wang et al. (2008) presented a generative probabilistic CF framework based on the PRP (Robertson 1977) and derived three models: an item-based, a user-based and a unified model.

2.3.2 *Probabilistic graphical models*

To continue with probabilistic models, it is interesting to mention the work of Barbieri and Manco (2011) and Barbieri et al. (2014) on adapting probabilistic graphical models to top-N recommendation. These models, such as *latent Dirichlet allocation* (LDA) or *probabilistic latent semantic analysis* (PLSA), have extensively been used in information retrieval tasks (Wei and Croft 2006; Zhai and Massung 2016). LDA is a generative statistical model initially devised by Blei et al. (2003) for topic modeling while

PLSA was proposed by Hofmann (1999) to address document indexing. Moreover, Hofmann (2004) also developed specific latent semantic models for collaborative filtering. Nevertheless, the main practical problem of probabilistic graphical models is the scalability of the inference methods with the volume of data and the number of latent factors (Barbieri et al. 2014; Croft et al. 2015).

2.3.3 *Query expansion models*

Regarding query expansion techniques, pseudo-relevance feedback models have been adapted to different recommendation tasks with great success. More specifically, Bellogín et al. (2013b) used relevance models (a state-of-the-art PRF technique) to compute user neighborhoods and applied the negative cross entropy ranking principle to generate recommendations. On the other hand, Parapar et al. (2013) presented an analogy between PRF and CF that allowed them to use relevance models to create a user-based collaborative filtering recommender.

Parapar et al. (2013) modeled the top-N recommendation task as a pseudo-relevance feedback task obtaining high figures of accuracy. In fact, Kallumadi et al. (2018) used this approach to address the ACM RecSys 2018 Challenge[2]. Their solution combine several approaches; one of them is the adaptation of relevance models for collaborative filtering proposed by Parapar et al. (2013) to tackle the automatic playlist continuation task. They achieved competitive performance since they ranked 7 out of 112 and 5 out of 31 in the two different tracks of the challenge.

Given the success of adapting pseudo-relevance feedback models to collaborative filtering, we continue this work in PART III. In particular, Chapter 5 studies the effect of smoothing techniques and prior probability estimates in relevance models for collaborative filtering. In Chapter 6, we propose the item-based counterpart of relevance models for CF showing that it is an effective algorithm for tackling a new task: getting rid of long tail items with recommendations. Finally, Chapter 8 present more cost-effective CF models by adapting cheaper PRF techniques based on the Rocchio framework.

2 The RecSys Challenge is an annual open competition organized within the ACM Conference on Recommender Systems. It usually presents a real-world task with a dataset provided by a company. The challenge presented in 2018 consisted in automatic playlist continuation and was organized by Spotify, University of Massachusetts–Amherst and Johannes Kepler University. More information is available at: `http://www.recsyschallenge.com/2018`.

2.3.4 *Matrix factorization models*

Even though there exist multiple approaches to build recommendation algorithms, matrix factorization techniques are predominant in the field (Koren and Bell 2015). Among different MF techniques, *singular value decomposition* (SVD) models are arguably the most popular in the recommender systems literature (Cremonesi et al. 2010; Hu et al. 2008; Koren 2008; Koren et al. 2009). These algebraic approaches compute low-rank approximations of the user-item matrix. SVD is also used in information retrieval under the name of *latent semantic analysis* (LSA). LSA factorizes the document-term matrix using a low-rank approximation and retrieval is performed by using this latent factor representations of documents and terms (Deerwester et al. 1990).

Non-negative matrix factorization (NMF) is another matrix factorization technique commonly used in recommendation (Liu et al. 2010). NMF decomposes the original matrix in the product of two matrices with non-negative elements (Lee, Daniel; Seung 2001). Both SVD and NMF have been used for document clustering tasks in IR (Aggarwal and Zhai 2012).

The main difference in the use of matrix factorization techniques in IR and RS lies in the use of the decomposition. Information retrieval techniques employ the latent vector representation of documents and terms to tackle retrieval or clustering tasks. In contrast, recommender systems literature reconstructs the original matrix with the latent vector decomposition to obtain an estimate of unknown values of the initial matrix. These reconstructed values constitute the output scores of the recommender.

Matrix factorization has also been applied to query expansion. In particular, Zamani et al. (2016) presented relevance feedback matrix factorization (RFMF), a pseudo-relevance feedback technique based on non-negative matrix factorization (NMF). This technique uses the parallelism between PRF and CF proposed by Parapar et al. (2013) in the opposite direction. In PART V, we propose a new pseudo-relevance feedback framework based on linear methods that takes inspiration from similar models in recommender systems literature.

Part II

RESEARCH METHOD

Reality continues to ruin my life.

— Bill Watterson,
Calvin & Hobbes

3

EVALUATION

This chapter presents the evaluation guidelines followed in this doctoral thesis to evaluate information retrieval and recommender systems models. IR and RS are two fields with a strong empirical focus. Many of these information systems are employed by millions of users daily. For this reason, the assessment of retrieval and recommendation models with rigorous experiments is paramount to meet the information demands of the users. Evaluation methods must enable us to select the best model among several competitors.

We can evaluate different models with online or offline experiments. Online experimentation is expensive because it requires to deploy different models and study real users' feedback or behavior. Additionally, online experiments are difficult, if not impossible, to reproduce in an academic position since it requires access to the production environment. Furthermore, online evaluation depends on several variables such as the domain, the demographics of the users or the user interface. For all these reasons, offline experimentation has its place and usually constitutes the first step before online experimentation. Offline evaluation usually exploits datasets collected in real platforms.

This thesis relies on offline evaluation.

Next, we detail information retrieval and recommender systems evaluation methods and protocols. We also describe the datasets and metrics. We begin with information retrieval models and end with recommender systems.

3.1 INFORMATION RETRIEVAL EVALUATION

Evaluation can be carried out from two main perspectives: efficiency and effectiveness. Efficiency takes into account the time and space requirements of a given model. These requirements can be measured theoretically, by a temporal and spatial complexity analysis (Cormen et al. 2009), or empirically, by monitoring and profiling the system. Effectiveness, on the other hand, measures how well the output of the system meets the information needs of the users. In general, this thesis is more focused on effectiveness than efficiency. Therefore, our primary focus is to compare the effectiveness of the proposed models against state-of-the-art baselines. Nonetheless, there are cases where our main contribution does not consist in outperform the effectiveness figures of the state of the art, but to provide a more cost-effective solution with a slight decrease in efficiency. In those cases, we also present the analysis of the efficiency of the model.

Offline evaluation in the information retrieval field is well-established by the Cranfield paradigm and the TREC initiative (Voorhees 2002). Next, we explain this standard IR evaluation protocol.

3.1.1 *Cranfield paradigm and TREC*

IR has well-established evaluation procedures.

The Cranfield paradigm and the TREC initiative provide a standard way of measuring how a retrieval system meets the information needs of the users. This evaluation protocol requires the use of test collections that contain a set of documents, a set of topics and a set of relevance judgments for those topics (Voorhees 2002). The Cranfield collection was the first rigorous test collection developed for information retrieval evaluation. Later, the annual Text Retrieval Conference (TREC) was established by the US National Institute of Standards. The TREC initiative has released multiple test collections within different tracks to tackle several IR tasks. To build these collections, a group of assessors judges the documents to indicate which ones are relevant to each topic. With these relevance judgments, we can evaluate retrieval systems by running the queries from the topics and computing ranking-oriented metrics on the output.

The evaluation paradigm relies on three fundamental assumptions: i) the information need of the user—specified by a topic— can be approximated by topical similarity, ii) relevance is independent of the users which implies that a set of relevance judgments is valid for any user and iii) relevance judgments are complete, that is, all the relevant documents for a topic are identified are known. Although these assumptions are not generally true, they are reasonable and some deficiencies can be compensated (Voorhees 2002). Therefore, this paradigm has become the standard systematic approach to the evaluation of retrieval systems.

For a topic, we generate a list of documents sorted by decreasing score. Then, ranking-oriented metrics evaluate these rankings using the relevance judgments for that topic. The quality of a retrieval strategy is measured as the average metric score for all topics.

The main problem of this approach is that the volume of information in modern test collections is too large to have complete relevance judgments. For this reason, a process called *pooling* is conducted to select which documents are evaluated by human assessors (Spärck Jones and Van Rijsbergen 1975; Voorhees 2002). Those documents that do not appear in the pool are assumed to be non-relevant. Pooling is based on the idea that we only use test collections to make relative evaluations of systems. We are not interested in the absolute values that a metric gives to the systems. Instead, we want to discriminate if a system is better than another. To ensure this, relevance judgments should be unbiased. Since having complete judgments is not feasible, pooling (if performed correctly) can be a good enough approximation (Voorhees 2002). However, large-scale datasets such as ClueWeb[1] contain hundreds of millions of documents which are shallow pooled resulting in many potentially relevant documents unjudged.

Relevance judgments can be either binary (a document is relevant or non-relevant for a given topic) or graded (documents can have different levels of relevance for a particular topic). Next, we present the most common metrics used in information retrieval to assess the effectiveness of retrieval models.

A consensus of assessors determines relevance.

3.1.2 *Metrics*

Most IR metrics range from zero to one. A value of zero represents the worst possible outcome while a value of one indicates perfect effectiveness. Since these metrics are computed on a per-query basis, we compute the average over all queries to obtain a single aggregated value. It is common to truncate the ranking of results until certain position n (commonly known as *cut-off*) and represented by $@n$ at the end of the metric name. In our IR experiments, if we do not specify the cut-off, it means $n = 1000$ which is the standard cut-off in many TREC tracks. To compute these metrics, we use `trec_eval`[2], a tool developed to assess the effectiveness in multiple TREC tracks. Next, we present the most common information retrieval metrics.

1 More information about ClueWeb collections is available at https://lemurproject.org/clueweb09.php (ClueWeb09) and https://lemurproject.org/clueweb12.php (ClueWeb12).

2 The source code is available at: https://github.com/usnistgov/trec_eval.

3.1.2.1 *Precision*

Precision and recall constitute the fundamental IR metrics.

This metric measures the proportion of retrieved documents that are relevant:

$$P@n = \frac{\#(\text{relevant documents retrieved until position } n)}{n} \tag{3.1}$$

3.1.2.2 *Recall*

Recall measures the proportion of relevant documents that are retrieved with respect to the total number of relevant documents in the collection:

$$R@n = \frac{\#(\text{relevant documents retrieved until position } n)}{\#(\text{relevant documents})} \tag{3.2}$$

3.1.2.3 *Average Precision*

MAP is arguably the most popular IR metric.

This metric approximates the area under the precision-recall curve. More specifically, it computes the precision at the positions where a relevant document is found. In contrast to precision and recall, this metric use positional information. We denote the indicator function (which returns 1 when the argument is true and 0 otherwise) by $\mathbb{I}$. This metric receives the name of mean average precision (MAP) when it is averaged over the set of topics.

$$AP@n = \frac{\sum_{k=1}^{n} \mathbb{I}(\text{document at position } k \text{ is relevant})\, P@k}{\#(\text{relevant documents})} \tag{3.3}$$

3.1.2.4 *Normalized Discounted Cumulative Gain*

When it comes to graded relevance, nDCG is the most common choice.

This metric considers graded relevance as well as positional information (Järvelin and Kekäläinen 2002). First, we need to define discounted cumulative gain (DCG) which discounts the graded relevance value of the retrieved documents by the position they appear at. We denote the relevance value of the document at position k by $\text{rel}(k)$.

$$DCG@n = \sum_{k=1}^{n} \frac{\text{rel}(k)}{\log_2(k+1)} \tag{3.4}$$

To allow comparing distinct models, we need to normalize the values of DCG. To this end, we calculate the ideal DCG (IDCG) which is the DCG score of the perfect ranking of documents. Then, we can simply compute normalized discounted cumulative gain as follows:

$$nDCG@n = \frac{DCG@n}{IDCG@n} \tag{3.5}$$

COLLECTION	#DOCS	AVG DOC LENGTH	TRAINING TOPICS	TEST TOPICS
AP88-89	165k	284.7	51 – 100	101 – 150
TREC-678	528k	297.1	301 – 350	351 – 400
Robust-04	528k	28.3	301 – 450	601 – 700
WT10G	1,692k	399.3	451 – 500	501 – 550
GOV2	25,205k	647.9	701 – 750	751 – 800

Table 3.1: Statistics of information retrieval test collections.

3.1.2.5 *Reliability of improvement*

RI is a metric proposed by Sakai et al. (2005) to assess query expansion. The reliability of improvement measures if the expanded queries tend to outperform the original ones (usually in terms of MAP). It ranges from −1 to 1 and is computed as follows:

$$RI = \frac{\#(\text{topics improved by QE}) - \#(\text{topics hurt by QE})}{\#(\text{topics})} \tag{3.6}$$

3.1.3 *Datasets*

In this thesis, we use five different TREC test collections: AP88-89, TREC-678, Robust-04, WT10G and GOV2. The first one is a subset of the Associated Press collection from years 1988 and 1989. The second collection is based on TREC disks 4 and 5. The third dataset was used in the TREC Robust Track 2004 and consists of poorly performing topics. The fourth one, the WT10G collection, is a general web crawl used in the TREC Web track 2000-2001. Finally, we also ran our experiments on a large dataset, the GOV2 collection, which is a web crawl of `.gov` websites from 2004 (used in the TREC Terabyte track 2004-2006 and the Million query track 2007-2008). Table 3.1 shows the main statistics of each collection.

We applied training and test evaluation on all collections. We use the training topics to tune the hyperparameters of the retrieval models and the test topics for assessing the effectiveness of the models. We tune our models to maximize mean average precision on the training topics of each collection.

3.2 RECOMMENDER SYSTEMS EVALUATION

Compared to information retrieval, there are more controversial issues and open problems in the evaluation of recommender systems. As explained in Section 2.2.1, recommender systems used to be defined as rating predictors: their aimed to forecast the ratings that users would give to each item. Nonetheless, it has been acknowledged that the assessment of RS models based on rating prediction does not lead to good recommenders. For this reason, recommender systems evaluation now focus on top-N recommendation.

Accuracy is measured in terms of error or ranking metrics depending on the task.

Offline evaluation of recommender systems is typically focused on *accuracy* (Gunawardana and Shani 2015; Herlocker et al. 2004). In the rating prediction task, accuracy is measured with error metrics between the predicted score and the actual score. In contrast, in the top-N recommendation task, accuracy is measured with ranking accuracy metrics that assess how a model places relevant items in the recommendations rankings.

Novelty, diversity or serendipity are also important aspects of RS.

In addition to accuracy , there is an increasing interest in evaluating diverse recommendation properties such as diversity and novelty (Castells et al. 2015) or serendipity (Kotkov et al. 2016). *Diversity* measures whether a recommender system suggests different items or, on the contrary, recommends mostly the same items. On the other hand, *novelty* is usually measured as how unusual the recommended items are. Both properties, novelty and diversity, are closely connected and, to some degree, complementary (Castells et al. 2015). An accurate recommendation model can be useless if the user already knows the suggested items or if all the recommendations look the same. Recommender systems should also try to suggest items that users would not have discovered by themselves. This property, called *serendipity*, is difficult to measure and is usually approximated by novelty and relevance (Castells et al. 2015; Kotkov et al. 2016). Somehow, novelty is a similar concept to serendipity but weaker: novel recommendations provide the users with information about uncommon items, although these items could have been discovered eventually.

Recommender systems that strongly focus on accuracy may give poor results on diversity and novelty metrics and vice versa. Intuitively, we can see that if we recommend to the users the most popular items within their neighborhood, the suggestions will be accurate but diversity and novelty will suffer. On the contrary, recommending unusual items can improve novelty and diversity at the risk of making some mistaken suggestions. This balance between accuracy and diversity/novelty is perhaps the most prominent trade-off in the field of recommender systems (Castells et al. 2015; Kunaver and Požrl 2017; Landin et al. 2018; Zhou et al. 2010b).

We can say that a recommendation is accurate when the user likes the item being suggested. Recommender systems test collections do not rely on pooling. Instead, they exploit a dataset of previously collected user-item interactions (such as ratings or clicks) (Gunawardana and Shani 2015). This dataset is usually divided into two splits: the training split is used as input to the recommendation algorithm and the test split is employed for measuring the performance of the recommender system using different evaluation metrics. An extra validation split for tuning the hyperparameters can also be used.

3.2.1 *Evaluation protocols*

In the rating prediction task, the only possible evaluation protocol consists in predicting the score of the items that the user rated in the test set. However, this protocol only assesses the quality of the model for those items that the user has deliberately chosen to rate (Steck 2013). In contrast, in the top-N recommendation task, several offline evaluation protocols have been proposed (Bellogín et al. 2011; Cremonesi et al. 2010; Steck 2013).

TestItems protocol allows reliable evaluation of RS.

In this doctoral thesis , we decided to follow the *TestItems* approach which has been regarded as a fair evaluation protocol (Bellogín et al. 2011). TestItems protocol consists in ranking, for each user, all the items in the test set that have not been rated by that user in the training set. In this way, an ideal recommender system will be able to achieve a perfect score. Note that this evaluation procedure is highly correlated to other variants (Bellogín et al. 2011).

The relevance threshold specifies when a rating indicates relevance.

When evaluating recommender systems with explicit feedback datasets, we need to select a *relevance threshold*. All the items rated by the target user u in the test set with a value below the certain relevance threshold are considered non-relevant items and form the set $\mathcal{N}_u$. Likewise, $\mathcal{R}_u$ represents the set of relevant items for user u, i.e., those items rated by u in the test set with a score greater than or equal to the relevance threshold. When using datasets with ratings ranging from 1 to 5, it is common to set this threshold to 4. Therefore, in this thesis, we set the relevance threshold to that value.

Those items that the target user did not rate are considered unjudged (their relevance is unknown). Most ranking metrics ignore unjudged elements and treat them as non-relevant, but some metrics explicitly consider them separately. It has been acknowledged that considering non-rated items as non-relevant may underestimate the true metric value (since non-rated items can be of interest to the user); however, it provides a better estimation of the recommender quality (Bellogín et al. 2011; McLaughlin and Herlocker 2004).

3.2.2 *Metrics*

Now, we present the metrics we use in this thesis to measure the effectiveness of different recommender systems. The following metrics range from zero to one where the higher the value, the better. We measure three properties: ranking accuracy, diversity and novelty. Except for the diversity metric, the rest are computed on a per-user basis (denoted with the subscript u). To compute a single aggregated value, we compute the average over all users. If a recommendation model cannot provide recommendations for a particular user, we assign a value of zero to all metrics for that user to penalize low user coverage (i.e., not being able to generate recommendations for some users). We denote the ranking cut-off by $@n$ at the end of the metric name.

The ranking accuracy metrics presented here are equivalent to their counterparts in information retrieval. We use `rec_eval`[3], our own fork of `trec_eval` extended for RS evaluation, to compute the ranking accuracy metrics.

3.2.2.1 *Precision*

Precision measures how well a method puts relevant items in the first n recommendations regardless of the rank. Precision is usually more important than recall in RS because we aim to generate relevant recommendations.

$$P_u@n = \frac{|L_u^n \cap \mathcal{R}_u|}{n} \tag{3.7}$$

3.2.2.2 *Normalized Discounted Cumulative Gain*

This metric exploits the values of the ratings and the position of the items in the ranking (Järvelin and Kekäläinen 2002). As in IR, this metric is computed as DCG divided by ideal DCG:

$$nDCG_u@n = \frac{DCG_u@n}{IDCG_u@n} \tag{3.8}$$

3 The source code is available at: `https://github.com/dvalcarce/rec_eval`.

and the discounted cumulative gain (DCG) is defined as:

$$DCG_u@n = \sum_{k=1}^{n} \frac{r(u, L_u^n[k])}{\log_2(k+1)} \tag{3.9}$$

where the gain value at each position is given by the rating that the user gave to the item in that position in the test set.

3.2.2.3 *Gini index*

The Gini index or Gini coefficient measures the diversity of the recommendations. This coefficient was initially proposed for quantifying wealth distribution inequalities, but it has also been utilized for measuring recommendation diversity (Castells et al. 2015; Fleder and Hosanagar 2009; Gunawardana and Shani 2015). Note that we use the complement of this metric (one minus the value of the index) for convenience. In this way, when the index is zero, it indicates that a single item is recommended for every user which corresponds to the minimum diversity scenario. On the contrary, a value of one means that all the items are equally recommended across the users. We compute this metric as follows:

Gini is a global metric of diversity.

$$Gini@n = 1 - \frac{1}{|\mathcal{I}| - 1} \sum_{j=1}^{|\mathcal{I}|} (2j - |\mathcal{I}| - 1)\, p(i_j|\text{rec}@n) \tag{3.10}$$

where $i_1, \cdots, i_{|\mathcal{I}|}$ is the list of items sorted by increasing $p(i_j|\,\text{rec}\,@n)$. This term refers to the probability that item i_j is being recommended in some recommendation list of length n and is given by:

$$p(i|\,\text{rec}\,@n) = \frac{|\{u \in \mathcal{U} | i \in L_u^n\}|}{\sum_{u \in |\mathcal{U}|} |L_u^n|} \tag{3.11}$$

Note that the Gini index is a global metric that is computed using all the recommendation lists. Therefore, it cannot be calculated on a per-user basis.

3.2.2.4 *Mean Self-Information*

MSI is an information theoretic metric. Zhou et al. (2010b) proposed to use the mean self-information to quantify the ability of a recommender system to generate unexpected recommendations. This metric is also called *surprisal* because it measures the improbability of an outcome. To quantify the probability of an outcome, we use the concept of popularity, that is, the proportion of users that interacted with the item.

MSI measures the surprisal of the recommendations.

$$MSI_u@n = \sum_{i \in L_u^n} \log \frac{|\mathcal{U}|}{|\mathcal{U}_i|} \tag{3.12}$$

Dataset	Users	Items	Ratings	Density
MovieLens 100k	943	1682	100 000	6.305 %
MovieLens 1M	6040	3706	1 000 209	4.468 %
MovieLens 10M	71 567	10 681	10 000 054	1.308 %
R3-Yahoo	15 400	1000	365 703	2.375 %
LibraryThing	7279	37 232	749 401	0.277 %
BeerAdvocate	33 388	66 055	1 571 808	0.071 %

Table 3.2: Statistics of recommendation datasets with explicit feedback.

3.2.3 *Datasets*

Rating-based datasets are commonly used in the assessment of collaborative filtering recommenders.

In this thesis, we evaluate recommender systems on datasets with explicit feedback in the form of ratings. We use the MovieLens datasets[4] which come in different sizes (100k, 1M and 10M ratings) and contain movie ratings. We use the R3-Yahoo! Music[5] dataset which comprises music ratings. We also use the LibraryThing and the BeerAdvocate[6] datasets that contain book and beer ratings, respectively. Table 3.2 shows the number of users, items and ratings, as well as the density (percentage of user-item pairs that have a rating) of the datasets with explicit feedback. These datasets contain ratings supplied by users of a platform during normal interaction. However, the R3-Yahoo test set is composed of randomly selected songs collected during a survey. For the rest of the datasets, we created the training and test splits by taking 80% of the ratings of each user for the training set and the remaining data is used as the test set. The rationale of this decision is to avoid certain biases by having the same proportion of training and test data for each user.

Ta-Feng dataset contains purchase information.

In Chapter 6, we also use the Ta-Feng dataset which contains Chinese grocery store transaction data from November 2000 to February 2001. This collection with 32 266 users and 23 812 has 817 741 transactions (density of 0.106 %).

We use these LBSN datasets to evaluate the group formation task.

Finally, in Chapter 7, we conduct experiments on datasets collected from four location-based social networks (LBSN). We use a dataset from Foursquare[7] containing users check-ins, ratings, venues and the social links connecting users (Levandoski et al. 2012; Sarwat et al. 2014). Stemming from this dataset, which is called from now on *FS*, we built a second

4 Available at: https://grouplens.org/datasets/movielens.
5 Available at: https://webscope.sandbox.yahoo.com/catalog.php?datatype=r.
6 Available at: https://snap.stanford.edu/data/web-BeerAdvocate.html.
7 Available at: https://archive.org/details/201309_foursquare_dataset_umn.

Dataset	Users	Items	Links	Check-ins	Ratings
FS	2 138 367	83 999	27 098 472	1 021 966	2 809 580
FS-NYC	103 663	7813	1 890 844	157 064	330 043
Gowalla	196 591	1 280 969	1 900 654	6 442 892	–
Brightkite	58 228	772 966	428 156	4 747 281	–
Weeplaces	15 799	971 307	114 131	7 369 712	–

Table 3.3: Statistics of LBSN datasets.

dataset by selecting only the check-ins that fall in New York City[8]. We called this second dataset *FS-NYC*. We take this subset to evaluate models in a less sparse scenario where all the information is concentrated in a single location. Additionally, we used two other datasets, collected from Brightkite[9] and Gowalla[10] made available by Cho et al. (2011). These datasets record user check-ins and the social links connecting users, but they lack ratings of the visited venues. Finally, we used the Weeplaces dataset[11] which contains check-ins and friendship relationships of Foursquare users who used the Weeplaces application. Table 3.3 shows the main statistics of these datasets.

8 We used the geographical information from: `https://www.flickr.com/places/info/2459115`.
9 Available at: `https://snap.stanford.edu/data/loc-Brightkite.html`.
10 Available at: `https://snap.stanford.edu/data/loc-Gowalla.html`.
11 Available at: `https://www.yongliu.org/datasets`.

4 STUDY OF RANK ACCURACY METRICS FOR RECOMMENDER SYSTEMS

In the previous chapter, we described the IR and RS evaluation methods used in this thesis. We presented the evaluation protocols, the metrics and the datasets. Compared to information retrieval, recommender systems evaluation methods are not well-established yet.

Selecting the appropriate metric is still an open issue in offline evaluation. Ranking accuracy metrics are commonly used in information retrieval and recommender systems evaluation. These metrics have been thoroughly studied in IR; in particular, their robustness and the discriminative power. In this chapter, we aim to shed light on the advantages of different ranking metrics in RS evaluation and find the most reliable ones. To this end, we conduct a robustness and discriminative power analysis of ranking accuracy metrics in the top-N recommendation task.

We aim to find the most reliable rank accuracy metrics for RS evaluation.

The findings of this chapter have shaped aspects of RS evaluation methods used throughout this thesis. More specifically, the choice of ranking accuracy metrics and their cut-off is motivated by the results presented here. The contributions of this chapter have been published recently (Valcarce et al. 2018f).

4.1 AN IR PERSPECTIVE FOR EVALUATING RS METRICS

Offline evaluation is standardized in IR by the Cranfield paradigm and the TREC initiative (Voorhees 2002). The Cranfield paradigm (described in Section 3.1.1) measures how a retrieval system meets the information needs of the users using ranking-oriented metrics. Many of these metrics have also been used for assessing the effectiveness of recommender systems in the top-N recommendation task. Since RS lack proper relevance judgments, researchers use a hold-out data from the collection to assess the quality of the recommendations. These judgments are incomplete and obtained in a very different way compared to the IR relevance judgments.

Since the assumptions of the Cranfield paradigm are substantially different from those of the recommender systems evaluation, we may ask: *should IR metrics be applied to RS?* Although most of these metrics are already being used in RS evaluation, they have not been thoroughly studied in this field. Previous work has analyzed the robustness to incomplete relevance judgments and the discriminative power of ranking accuracy metrics in the context of the Cranfield paradigm. We say that a metric is *robust* when it shows the same behavior when fewer relevance judgments are available. Likewise, a metric is *discriminative* when changes in its values indicate statistically significant differences. To answer the former question, we study the robustness and discriminative power of several ranking accuracy metrics for top-N recommendation.

4.2 RELATED WORK

The limitations and biases of the Cranfield paradigm have been extensively studied. There have been efforts to overcome the bias produced by pooling (Buckley et al. 2007; Büttcher et al. 2007). Also, Buckley and Voorhees (2004) studied in IR how the number of relevance judgments affects different precision-oriented metrics. They defined the robustness of a metric with respect to incomplete judgments as how well the metric correlates with itself when the relevance judgments are incomplete. They also designed a new metric for using with incomplete judgments, bpref, that correlates with itself with all judgments and with AP better than other standard IR metrics. They also found that bpref preserves the absolute scores and the relative ranking of systems better than MAP or precision. Yilmaz and Aslam (2008) later proposed three estimates of AP for the incomplete judgments scenario. Their proposals showed a better correlation between themselves and AP than bpref. These correlations between system rankings were measured in terms of Kendall's tau correlation (Kendall 1938). Among the three proposals, inferred average precision was the metric that provided the best results (Yilmaz and Aslam 2008). To measure the robustness to incomplete judgments in these experiments, the metrics were calculated using random subsets of relevance judgments. Buckley and Voorhees (2004) used stratified random sampling while Yilmaz and Aslam (2008) employed random sampling. However, both samplings are identical in expectation (Yilmaz and Aslam 2008). Additionally, Lu et al. (2016) thoroughly studied the effect of the pooling depth in several IR metrics providing a list of advice for IR evaluation.

Besides robustness to incompleteness, discriminative power is another attractive property of evaluation metrics that has been thoroughly studied in IR (Buckley and Voorhees 2000; Lu et al. 2016; Sakai 2006; Sakai and Kando 2008). This property indicates the capability of a metric to discriminate among systems. We should note that the discriminative power not only depends on the metric but also on the test collection and the set of systems being compared. Buckley and Voorhees (2000) proposed a first attempt to study the discriminative power of a metric using a fuzziness value. Later, Sakai (2006) introduced a more formal method based on the bootstrap test. Given a significance level (e.g., $p = 0.05$), he computed the ratio of system pairs for which a statistical test finds a significant difference. More specifically, Sakai (2006) employed the bootstrap test with Student's t statistic for this purpose. To avoid fixing a particular significance level, Lu et al. (2016) proposed to report the median system-pair p-value as a measure of discriminative power where lower values are better. Sakai and Kando (2008) also studied how incomplete judgments also affect the discriminative power in IR.

4.3 IR METRICS FOR RECOMMENDATION

IR and RS evaluation assumptions are quite different. Therefore, the best metrics for each scenario may differ.

We can establish a parallelism between the Cranfield paradigm and recommender systems evaluation. If users play the role of queries (since both are associated with an information need), then we only need to evaluate item rankings instead of document rankings. Cranfield evaluation makes use of relevance judgments to decide whether a document is relevant for a given query. In RS, we lack those judgments, but we can approximate them with hold-out data from the user. The problem with this parallelism is that Cranfield assumptions do not generally hold in recommender systems evaluation. In Table 4.1, we present a comparison of IR and RS evaluation assumptions. In particular, the main difference is that relevance in RS is highly dependent on the users: the same item may not be relevant to two different people. Moreover, relevance judgments are far from complete. Since relevance is personal, we cannot build a set of relevance judgments using a group of experts. Furthermore, since we build the test dataset with hold-out data, the incompleteness of the relevance judgments is intrinsic to the recommendation task.

Another difficulty with recommender systems evaluation is that the community lacks consensus about which metric is the most reliable to

Table 4.1: Comparison between information retrieval and recommender systems evaluation assumptions.

Information retrieval	Recommender Systems
Topical similarity can approximate the user's information need.	User's information need may be estimated in several different ways.
Relevance is independent of users.	Relevance is dependent of users.
Relevance judgments are almost complete (subject to pooling).	Relevance judgments are far from complete.

measure the ranking quality of recommendations. In contrast, AP is traditionally considered the reference metric in IR although recent criticism (Fuhr 2018) advocates for the use of nDCG. In addition, when approximating relevance judgments with a hold-out test set, how much data is used for the training and test splits should be balanced. A larger training subset (at the expense of a reduced test subset) will allow better modeling, but it would provide worse evaluation reliability and vice versa. Finally, the long tail distribution of items in RS impacts the recommendation process. In contrast, IR evaluation does not have to deal with such a great imbalance in the popularity of documents.

In light of these differences between IR and RS evaluation, we propose to study the suitability of the following ranking accuracy metrics that have been widely used in IR and are now being used in RS. In addition to the ranking accuracy metrics presented in Section 3.2.2 (precision and nDCG), we study the following ones:

RECALL measures the proportion of relevant items that are included in the recommendation list with respect to the total number of relevant items for a given user:

$$Recall_u@n = \frac{|L_u^n \cap \mathcal{R}_u|}{|\mathcal{R}_u|} \tag{4.1}$$

AVERAGE PRECISION computes precision at the positions where a relevant item is found. Recall that $\mathbb{I}$ is the indicator function.

$$AP_u@n = \frac{1}{|\mathcal{R}_u|} \sum_{k=1}^{n} \mathbb{I}\left(L_u^n[k] \in \mathcal{R}_u\right) P_u@k \tag{4.2}$$

RECIPROCAL RANK is computed as the inverse of the position of the first relevant element in the ranking. As AP, when averaged over a set of topics, this metric is called mean reciprocal rank (MRR).

$$RR_u = \frac{1}{\min_k \{L_u^n[k] \in \mathcal{R}_u\}} \tag{4.3}$$

BPREF was designed to be highly correlated with AP but more robust to incomplete relevance judgments (Buckley and Voorhees 2004). Bpref is inversely related to the number of judged non-relevant items that are located above each relevant item in the ranking list:

$$bpref_u@n = \frac{1}{|\mathcal{R}_u|} \sum_{k=1}^{n} rel\,(L_u^n[k]) \left(1 - \frac{\min(|L_u^k \cap \mathcal{N}_u|, |\mathcal{R}_u|)}{\min(|\mathcal{N}_u|, |\mathcal{R}_u|)}\right) \tag{4.4}$$

INFERRED AVERAGE PRECISION yields the same score MAP provides when the relevance judgments are complete; however, it is also a statistical estimate of AP when using incomplete judgments (Yilmaz and Aslam 2008). InfAP has shown a better correlation with AP than bpref under this scenario. This metric is given by:

$$infAP_u@n = \frac{1}{|\mathcal{R}_u|} \sum_{k=1}^{n} \mathbb{I}\,(L_u^n[k] \in \mathcal{R}_u)\, E[P@k] \tag{4.5}$$

where the expected precision at position k is defined as:

$$E[P@k] = \frac{1}{k} + \frac{k-1}{k} \frac{|L_u^{k-1} \cap \mathcal{R}_u| + \varepsilon}{|L_u^{k-1} \cap \mathcal{R}_u| + |L_u^{k-1} \cap \mathcal{N}_u| + 2\varepsilon} \tag{4.6}$$

and ε is a small constant (we set ε to 0.00001 in our experiments).

4.4 METHODOLOGIES TO STUDY RS METRICS

In this section, we propose methodologies to analyze the robustness to incompleteness and the discriminative power of the aforementioned metrics in the evaluation of top-N recommenders. We take inspiration from previous work of these properties in IR. We start with the analysis on robustness to incompleteness because relevance judgments are very scarce in the recommendation scenarios which hinders a reliable assessment of recommender systems. On the other hand, when preferring one recommendation model over another, we need to have statistically sound guarantees—the discriminative power of a metric measures this desirable property.

4.4.1 *Robustness to incompleteness*

Incompleteness is intrinsic to the recommendation scenario.

When evaluating recommender systems, incompleteness is pervasive. The ratings in the test set form the relevance judgments which are incomplete. In fact, this is an intrinsic property of the recommendation task: if the information is complete, we would know everything about the scenario and no recommendations could be made. Therefore, a desirable metric for recommendation should be robust to incompleteness in the test set. Incompleteness in an IR scenario has been simulated using unbiased random sampling techniques (Buckley and Voorhees 2004; Yilmaz and Aslam 2008). We propose a similar approach to induce incompleteness when evaluating recommender systems. However, RS evaluation has two types of sources of incompleteness (Bellogín et al. 2017). When we use ranking accuracy metrics to assess recommender systems, two well-differentiated biases arise: the *sparsity bias* and the *popularity bias*. For this reason, next, we analyze the robustness to the sparsity bias and the robustness to the popularity bias independently.

4.4.1.1 *Sparsity bias*

Recommenders have to deal with sparse data.

The sparsity bias arises in RS evaluation when we lack known relevance for all the user-items pairs (Bellogín et al. 2017). In recommendation, users' profiles are incomplete by definition: we build the test set as a hold-out subset of the users profile. Moreover, in a scenario without incompleteness, we would be unable to recommend anything because nothing unknown would be available to suggest. Note that the sparsity bias causes the absolute values of the metrics to lose meaning, but the relative values can still be valid for comparative purposes (Bellogín et al. 2017).

We propose to measure the robustness of different metrics to the sparsity bias by evaluating those metrics using random samples of the test set. We create test sets by removing relevance judgments randomly. For each test set size, we take different random samples. Given a set of recommender systems, we evaluate those systems according to a particular metric and compute the ranking of systems. Then, we measure the correlation of this ranking with respect to the ranking obtained by evaluating those systems using the original test set. We also use Kendall's tau coefficient as rank correlation measure (Kendall 1938). Finally, by averaging the rank correlation for each sample of the same size, we obtain a final estimate of the robustness of a metric for each test size. A highly robust metric will yield higher average correlation values.

4.4.1.2 *Popularity bias*

The long tail distribution of data introduces another bias into the RS.

In contrast to IR , missing relevance judgments are not uniformly distributed: this has been referred to as *missing not at random* (Marlin et al. 2007; Steck 2010, 2013). The distribution of ratings in a recommendation scenario follows a heavy skewed long-tail distribution. Bellogín et al. (2017) studied this popularity bias and found that it strongly affects the reliability of several ranking accuracy metrics .

Since previous works in RS remove popular items to deal with the popularity bias (Bellogín et al. 2017; Cremonesi et al. 2010), we propose to build progressively smaller test sets removing ratings from the most popular items to measure the robustness of a metric to this bias. Then, we can study the change in the correlation between systems rankings of different subsets of the test set and the original test set. The higher the value of the rank correlation, the higher the robustness of such metric to the popularity bias. We use again Kendall's tau coefficient to measure the correlation between ranking of systems (Kendall 1938).

4.4.2 *Discriminative power*

A metric should allow us to distinguish which model is better than the rest.

In addition to robustness to an incomplete test set, discriminative power is another desirable property of an evaluation metric. When we compare two recommendation techniques, we expect the variation in the values of a metric to indicate a statistically significant difference. Otherwise, if the difference is not significant, we would not be able to conclude anything with that metric.

We propose to measure the discriminative power of several information retrieval metrics on different datasets. We follow a procedure similar to the method presented by Sakai (2006), although we propose to change the statistical test. Instead of using Bootstrap with the Student's t statistic, we can employ the permutation test (also known as Fisher's randomization test) with the difference in means as test statistic (Efron and Tibshirani 1993). The reason is that the permutation test provides a better estimation of the p-value with a high number of permutations. Since computing the exact p-value requires the computation of 2^n permutations (where n is the number of test users), we can approximate the result of this test using Monte Carlo sampling. With 100,000 samples, we can compute a two-sided p-value of 0.05 with an estimated error of ±0.0007 and a p-value of 0.01 with an error of ±0.000315 according to the coefficient of variation of the estimated p-value (Efron and Tibshirani 1993).

For each metric, we plot the p-values of the statistical test between all possible system pairs sorted by decreasing value (Sakai 2006). We call each of those curves the p-value curve of a metric. Since a highly discriminative metric must yield low p-values, we prefer metrics with p-value curves close to the origin. Furthermore, we also propose to compute a single value that summarizes the discriminative power of a metric. For this purpose, we use the sum of the p-values between all system pairs as an approximation of the area under the p-value curve. We call this value DP (discriminative power). The lower the value of DP, the higher the discriminative power of the metric. Note that DP is only intended to compare metrics when using the same set of systems on the same datasets: the absolute values are not comparable across different experimental settings.

4.5 EXPERIMENTAL SETTINGS

We use three datasets from different domains: MovieLens 1M, LibraryThing and BeerAdvocate. On the one hand, to assess the robustness to incompleteness, we follow the approach proposed in Section 4.4.1.1 sweeping from samples with 100% of the ratings of the original test set to samples with 5% of the ratings in steps of 5% to simulate the sparsity bias. We compute the average of 50 samples of each test set size which provides a good estimate in our experiments. On the other hand, when using the methodology for analyzing the popularity bias, proposed in Section 4.4.1.2, we start from using the ratings of 100% of items to using only the ratings of the 80% least popular items in steps of 1%.

When examining ranking accuracy metrics, we need recommender systems to compare. Previous works in IR studying different metrics employed the runs submitted to TREC (Buckley and Voorhees 2004; Lu et al. 2016; Yilmaz and Aslam 2008). Since we do not have an equivalent in RS, we implemented 21 recommendation techniques and used their outputs to study the properties of several IR metrics[1]. Note that we have chosen multiple types of algorithms to have a representative set of recommendation techniques.

- **Random**: produces random suggestions for each user.
- **Popularity**: non-personalized algorithm. Its output is the list of items ranked decreasingly by the number of ratings.

1 We provide the source code of the experiments and the complete output of the recommender systems at `https://www.dc.fi.udc.es/~dvalcarce/metrics.html`.

- **RW**, **RSV**, **CHI2** and **KLD**: neighborhood-based techniques that stem from Rocchio's feedback model (Valcarce et al. 2016c). We present them in Chapter 8.
- **RM1** and **RM2**: neighborhood-based techniques based on relevance models (Parapar et al. 2013; Valcarce et al. 2016a). We extensively study them in Chapter 5.
- **LM-WSR-UB** and **LM-WSR-IB**: user-based and item-based recommendation approaches that use language models to compute neighborhoods (Valcarce et al. 2016b,e). We present them in Chapter 10.
- **NNCosNgbr-UB** and **NNCosNgbr-IB**: user-based and item-based versions of a neighborhood-based recommendation model (Cremonesi et al. 2010).
- **SLIM**: neighborhood-learning recommendation technique based on sparse linear methods (Ning and Karypis 2011).
- **HT**: model-based technique designed to deal with long tail items modeling the recommendation task as a random walk in a graph (Yin et al. 2012).
- **BPRMF**: matrix factorization technique based on Bayesian probabilistic ranking (Rendle et al. 2009).
- **SVD**: matrix factorization approach based on using singular value decomposition (Takács et al. 2009).
- **PureSVD**: matrix factorization technique that computes the SVD over all the entries of the user-item rating matrix (Cremonesi et al. 2010).
- **WRMF**: matrix factorization technique that weights the missing information in the rating matrix (Hu et al. 2008).
- **LDA**: model-based recommender that uses latent Dirichlet allocation (Blei et al. 2003).
- **PLSA**: model-based recommender that uses probabilistic latent semantic analysis (Hofmann 2004).
- **UIR-IB**: probabilistic technique that builds a probabilistic user-item relevance model (Wang et al. 2006).

4.6 CHOOSING AMONG CUT-OFFS

We study the same metric using different cut-offs.

When applying a ranking metric, we have to select the cut-off. Recommenders usually show only a few suggested items because users seldom consider more than the top ones. For this reason, recommender systems literature usually employ shallow cut-offs such as 5 or 10 (Gunawardana and Shani 2015). However, the selection of the exact value of the cut-off in some research papers is somewhat arbitrary. Although RS typically present few recommendations to their users, deeper cut-offs may provide a more reliable assessment of the recommenders offline evaluation. Therefore, next, we analyze which cut-offs are preferable regarding robustness and discriminative power.

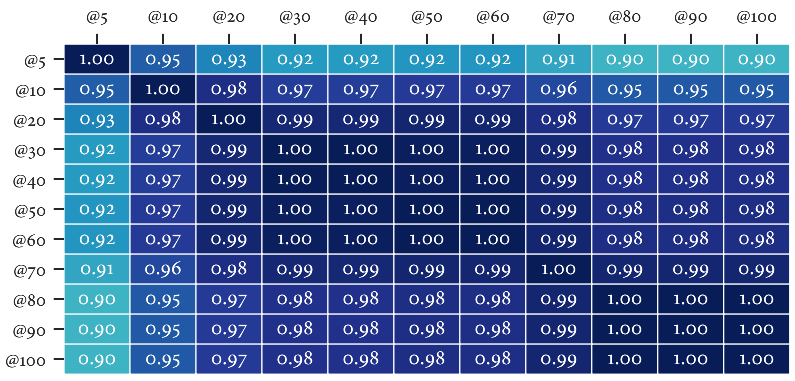
4.6.1 *Correlation among cut-offs*

We study Kendall's correlation between systems when using the same metric with different cut-offs. We find high correlations between rankings when studying cut-offs from 5 to 100. Overall, the correlation between cut-offs above 20 is very high. Those correlations are almost always higher than 0.9 on the LibraryThing and BeerAdvocate datasets. Note that previous work has considered that two rankings with a correlation above 0.9 are almost equivalent (Voorhees 2001). On the MovieLens dataset, most of the correlations are above 0.85 and the lowest found correlation was between P@5 and P@100 and Recall@5 and Recall@100 with a value of 0.76. For the sake of space, we choose a representative example: Figure 4.1 shows the correlation between different cut-offs of nDCG on MovieLens 1M.

The largest discrepancy is between the cut-off at 5 and the rest of cut-offs. However, all the correlations are at least 0.9 which represents a very strong correlation. Therefore, we can conclude from this experiment that the choice of the cut-off does not affect the ranking of the systems severely; however, it may affect the robustness or the discriminative power, which we analyze next.

4.6.2 *Robustness among cut-offs*

We test the robustness to sparsity and popularity of different cut-offs from 5 to 100 of each metric following the procedure explained in Section 4.4.1. The results confirm that larger cut-offs yield better figures of robustness when increasing the sparsity and the popularity bias of the test set. As



	@5	@10	@20	@30	@40	@50	@60	@70	@80	@90	@100
@5	1.00	0.95	0.93	0.92	0.92	0.92	0.92	0.91	0.90	0.90	0.90
@10	0.95	1.00	0.98	0.97	0.97	0.97	0.97	0.96	0.95	0.95	0.95
@20	0.93	0.98	1.00	0.99	0.99	0.99	0.99	0.98	0.97	0.97	0.97
@30	0.92	0.97	0.99	1.00	1.00	1.00	1.00	0.99	0.98	0.98	0.98
@40	0.92	0.97	0.99	1.00	1.00	1.00	1.00	0.99	0.98	0.98	0.98
@50	0.92	0.97	0.99	1.00	1.00	1.00	1.00	0.99	0.98	0.98	0.98
@60	0.92	0.97	0.99	1.00	1.00	1.00	1.00	0.99	0.98	0.98	0.98
@70	0.91	0.96	0.98	0.99	0.99	0.99	0.99	1.00	0.99	0.99	0.99
@80	0.90	0.95	0.97	0.98	0.98	0.98	0.98	0.99	1.00	1.00	1.00
@90	0.90	0.95	0.97	0.98	0.98	0.98	0.98	0.99	1.00	1.00	1.00
@100	0.90	0.95	0.97	0.98	0.98	0.98	0.98	0.99	1.00	1.00	1.00

Figure 4.1: Correlation between different cut-offs of nDCG metric on the MovieLens 1M dataset.

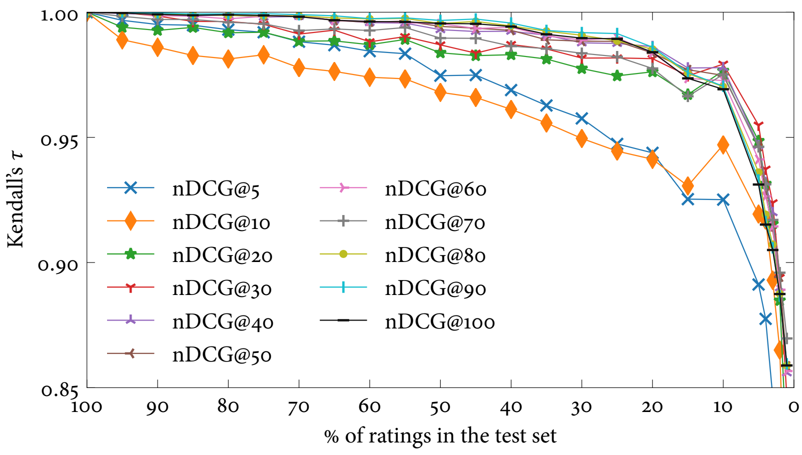


Figure 4.2: Kendall's correlation of different cut-offs of nDCG with respect to themselves using the full test set when increasing the sparsity bias on the MovieLens 1M dataset.

a representative example, Figure 4.2 plots the robustness to the sparsity bias of different cut-offs of nDCG on MovieLens 1M. Likewise, Figure 4.3 plots the robustness to the popularity bias of different cut-offs of nDCG. In both figures, we can see that robustness increases as we use deeper cut-offs. This phenomenon also occurs in the other studied metrics on the three datasets with slight variations. For the sake of brevity, we omit them.

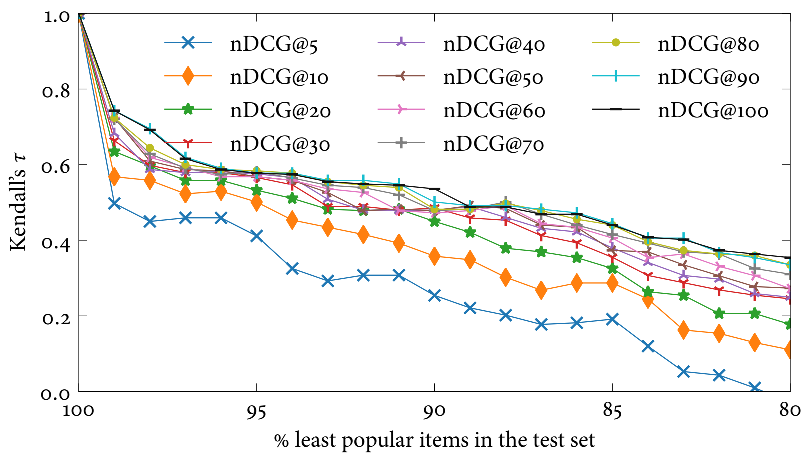


Figure 4.3: Kendall's correlation of different cut-offs of nDCG with respect to themselves using the full test set when increasing the popularity bias on the MovieLens 1M dataset.

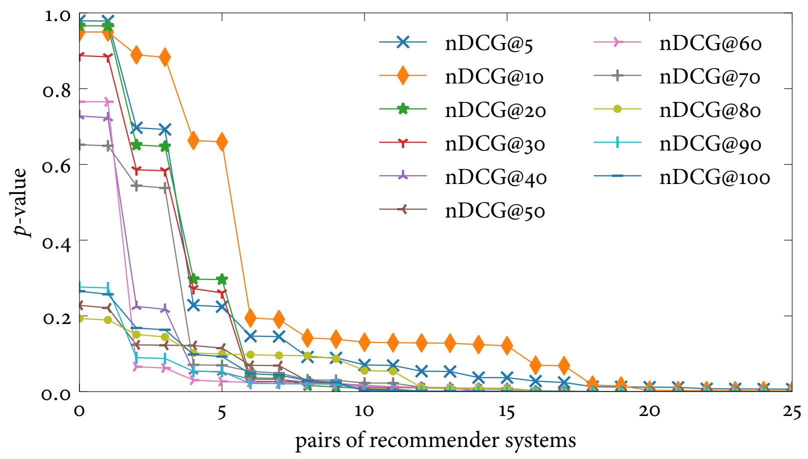


Figure 4.4: Analysis of the discriminative power of nDCG at different cut-offs on the MovieLens 1M dataset.

4.6.3 *Discriminative power among cut-offs*

Finally, we study the discriminative power of each metric using cut-offs from 5 to 100. Using the procedure described in Section 4.4.2, we plot the p-values of the paired statistical tests sorted by decreasing value on the MovieLens 1M (see Figure 4.4). We observe that deeper cut-offs (above 50) consistently provide better figures of discriminative power than shallower cut-offs. Different metrics on the three datasets present similar results.

4.6.4 *Implications*

In light of these results, we can conclude that the studied metrics with deeper cut-offs are more robust to the sparsity and popularity biases and have better discriminative power. Additionally, since the ranking of systems produced by a metric when varying the cut-off from 5 to 100 does not change notably, we should prefer deeper cut-offs. Therefore, if there is no strong reason to choose a shallow cut-off such as 5 or 10, calculating the metric over a larger ranking (let say $n = 100$ recommendations) should be preferred in offline experiments. Note that such deep cut-offs provide better properties even though we may (and generally will) lack n relevance judgments for each user.

4.7 CHOOSING AMONG METRICS

In the previous section, we compared each metric against themselves using different cut-offs and we found that a cut-off of $n = 100$ is a good choice due to its robustness and discriminative properties. Now, we fix the cut-off to 100 and compare the previous metrics among each other to study which have more desirable properties: robustness and discriminative power.

4.7.1 *Correlation among metrics*

Herlocker et al. (2004) have previously studied the correlation among some metrics (some of them barely used anymore) using only variants of one collaborative filtering algorithm on one dataset and recommended further investigation. Therefore, we study the correlation among the system orderings according to different modern ranking metrics on three datasets. Figure 4.5 shows Kendall's correlation among metrics on the MovieLens 1M, LibraryThing and BeerAdvocate datasets. On the LibraryThing collection, all correlations are above 0.9 threshold which indicates that the metrics produce almost identical rankings. On the other two datasets, we observed stronger differences with some correlations below 0.8.

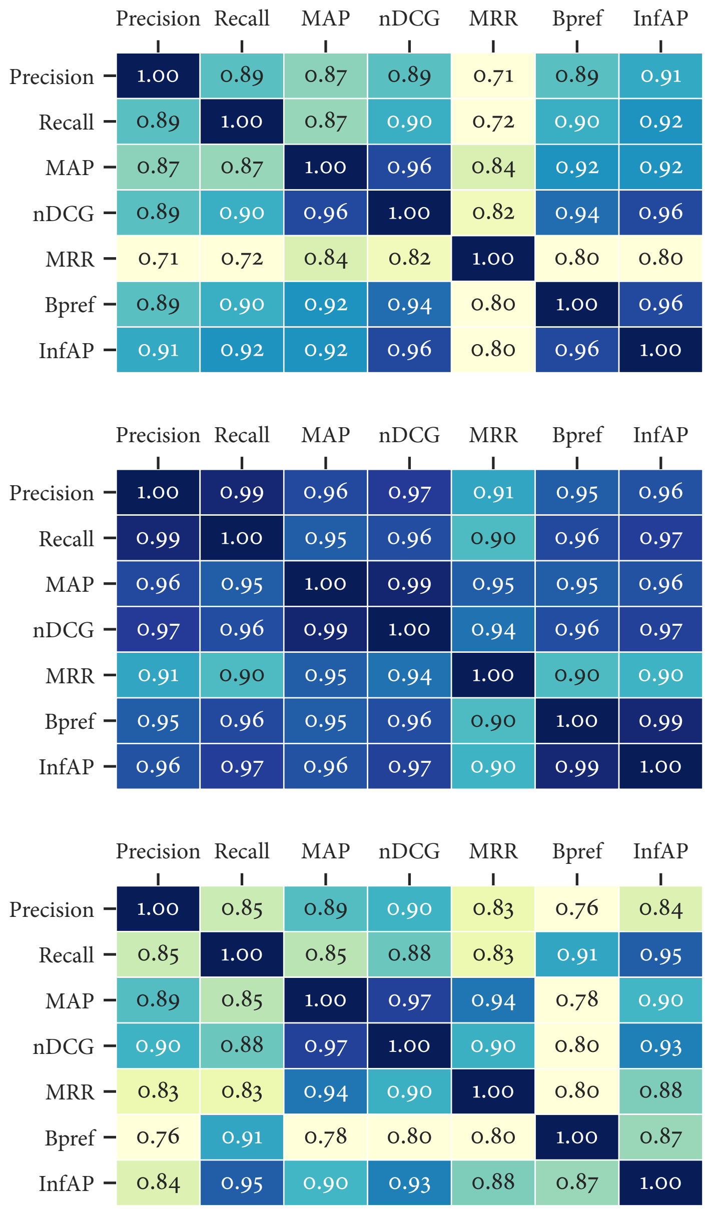
We can see that MRR differ noticeably from the rest, especially on the MovieLens 1M and the LibraryThing datasets. Bpref also shows a low correlation with the other metrics on the BeerAdvocate collection. It is worth remarking that bpref is poorly correlated with MAP on this dataset which is a surprising result since bpref was designed to do so (Buckley and Voorhees 2004). We suspect that this may be produced by the highly skewed long tail of this dataset. Instead, MAP is strongly correlated with nDCG on the three datasets. Nevertheless, the ranking produced by the rest of the metrics showed a fairly strong correlation among them.

4.7.2 *Robustness among metrics*

Figure 4.6 depicts the results of the experiments of robustness to the sparsity bias. We can see that all the metrics are fairly robust to this bias since the correlation is above 0.9 even when removing half of the test set. Precision and nDCG showed very good figures of robustness to sparsity on the three datasets (precision especially on BeerAdvocate). In contrast, bpref, and to a lesser extent InfAP and MRR, show poor robustness to sparsity. This result is interesting because it is different from what happens in IR. On the one hand, bpref and InfAP are techniques proposed for dealing with incomplete judgments in IR (Buckley and Voorhees 2004; Yilmaz and Aslam 2008), but in top-N recommendation they are less robust than other metrics. We should note that bpref and InfAP were designed for approximating average precision in scenarios with incomplete judgments while this metric is not such gold standard in recommendation. Still, it is surprising that MAP showed better robustness figures than bpref and InfAP on LibraryThing and BeerAdvocate. On the other hand, utility-based metrics such as MRR were found to be more resilient to changes in pooling depth which is related to the sparsity bias in recommendation (Lu et al. 2016).

Additionally, we report the results of the experiments of robustness to the popularity bias in Figure 4.7. On the BeerAdvocate dataset, the correlations quickly drop after removing a small percentage of the most popular items even reaching negative correlation values. This phenomenon is likely caused by the highly skewed long tail distribution of this dataset. Therefore, it is difficult to draw conclusions from this collection. Overall, precision is the best metric in terms of robustness to popularity whereas MRR is the worst one. The robustness to the popularity bias of the rest of the metrics depends heavily on the dataset.

We can claim that MRR is the least robust metric. This utility-based metric suffers heavily from sparsity and popularity biases. In contrast, precision is the most robust metric. More sophisticated metrics such as nDCG also present good figures of robustness; however, their additional



	Precision	Recall	MAP	nDCG	MRR	Bpref	InfAP
Precision	1.00	0.89	0.87	0.89	0.71	0.89	0.91
Recall	0.89	1.00	0.87	0.90	0.72	0.90	0.92
MAP	0.87	0.87	1.00	0.96	0.84	0.92	0.92
nDCG	0.89	0.90	0.96	1.00	0.82	0.94	0.96
MRR	0.71	0.72	0.84	0.82	1.00	0.80	0.80
Bpref	0.89	0.90	0.92	0.94	0.80	1.00	0.96
InfAP	0.91	0.92	0.92	0.96	0.80	0.96	1.00

	Precision	Recall	MAP	nDCG	MRR	Bpref	InfAP
Precision	1.00	0.99	0.96	0.97	0.91	0.95	0.96
Recall	0.99	1.00	0.95	0.96	0.90	0.96	0.97
MAP	0.96	0.95	1.00	0.99	0.95	0.95	0.96
nDCG	0.97	0.96	0.99	1.00	0.94	0.96	0.97
MRR	0.91	0.90	0.95	0.94	1.00	0.90	0.90
Bpref	0.95	0.96	0.95	0.96	0.90	1.00	0.99
InfAP	0.96	0.97	0.96	0.97	0.90	0.99	1.00

	Precision	Recall	MAP	nDCG	MRR	Bpref	InfAP
Precision	1.00	0.85	0.89	0.90	0.83	0.76	0.84
Recall	0.85	1.00	0.85	0.88	0.83	0.91	0.95
MAP	0.89	0.85	1.00	0.97	0.94	0.78	0.90
nDCG	0.90	0.88	0.97	1.00	0.90	0.80	0.93
MRR	0.83	0.83	0.94	0.90	1.00	0.80	0.88
Bpref	0.76	0.91	0.78	0.80	0.80	1.00	0.87
InfAP	0.84	0.95	0.90	0.93	0.88	0.87	1.00

Figure 4.5: Correlation of Precision, Recall, MAP, nDCG, MRR, bpref and InfAP (using a cut-off of 100) with each other on the MovieLens 1M (top), LibraryThing (middle) and BeerAdvocate (bottom) datasets.

Table 4.2: Values of DP of precision, recall, MAP, nDCG, MRR, bpref and InfAP (using a cut-off of 100) on the MovieLens 1M, LibraryThing and BeerAdvocate datasets. Lower values are better.

Dataset	P	Recall	MAP	nDCG	MRR	Bpref	InfAP
MovieLens 1M	2.6	7.0	2.8	**1.4**	15.5	9.9	8.4
LibraryThing	1.5	5.9	3.6	**0.2**	2.9	5.4	3.8
BeerAdvocate	**1.9**	8.3	10.7	4.4	5.8	12.7	4.8

complexity may be the reason why they are less robust than simple binary metrics such as precision.

4.7.3 *Discriminative power among metrics*

Figure 4.8 reports our findings in terms of discriminative power of the different studied metrics. We also present the values of DP (an approximation of the area under the p-value curve) in Table 4.2. Although the results vary across datasets, we can find some general trends. We can see that bpref, and to a lesser extent InfAP, presents low discriminative power across all datasets. In contrast, nDCG and precision (in this order) present the highest discriminative power on the test collections with great difference to the rest of the metrics. Finally, MAP, Recall and MRR show an erratic performance in terms of discriminative power depending on the dataset.

4.8 CONCLUSIONS

In this chapter, we studied the robustness and discriminative power of several ranking accuracy metrics, originally used in IR, when applied to the top-N recommendation task. To this end, we adapted and extended previous methodologies developed in IR for studying robustness against incompleteness and discriminative power.

Deeper cut-offs are more reliable in terms of robustness to sparsity and popularity biases.

We found that deeper cut-offs offer better robustness to sparsity and popularity biases than shallower cut-offs which are traditionally used in recommender systems evaluation. Therefore, in this thesis, we use a cut-off of 100 for the ranking accuracy metrics. Although only a few recommendations are usually displayed to the users, our investigation showed that deep cut-offs allow us to perform more robust and discriminative evaluations of recommender systems.

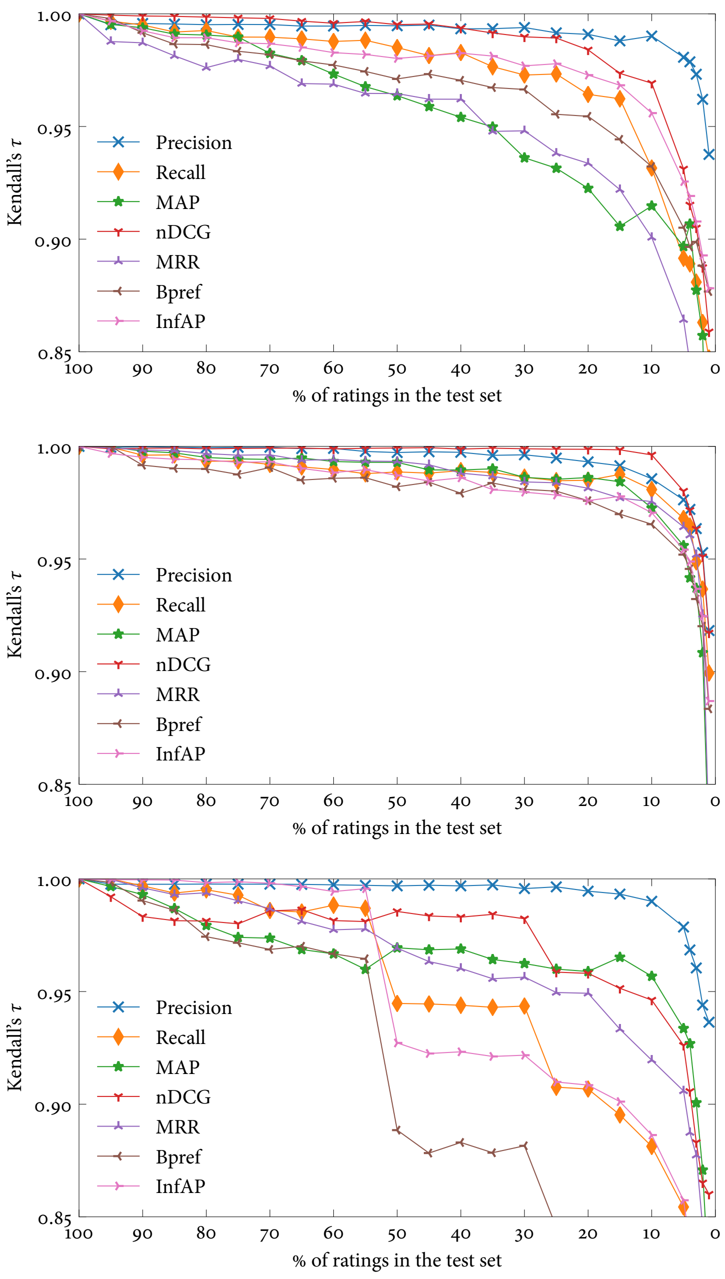


Figure 4.6: Correlation of Precision, Recall, MAP, nDCG, MRR, bpref and InfAP (using a cut-off of 100) with respect to themselves using the test set when increasing the sparsity bias on the MovieLens 1M (top), LibraryThing (middle) and BeerAdvocate (bottom) collections.

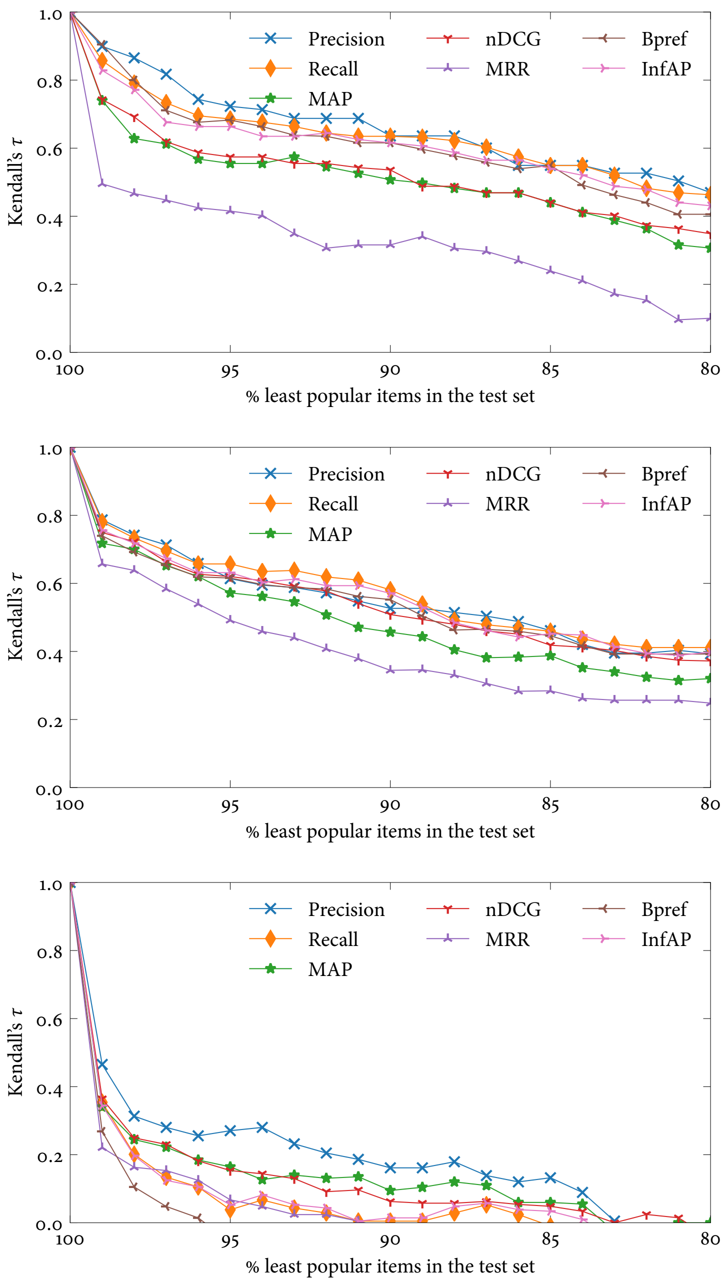


Figure 4.7: Correlation of Precision, Recall, MAP, nDCG, MRR, bpref and InfAP (using a cut-off of 100) with respect to themselves using the test set when increasing the popularity bias on the MovieLens 1M (top), LibraryThing (middle) and BeerAdvocate (bottom) collections.

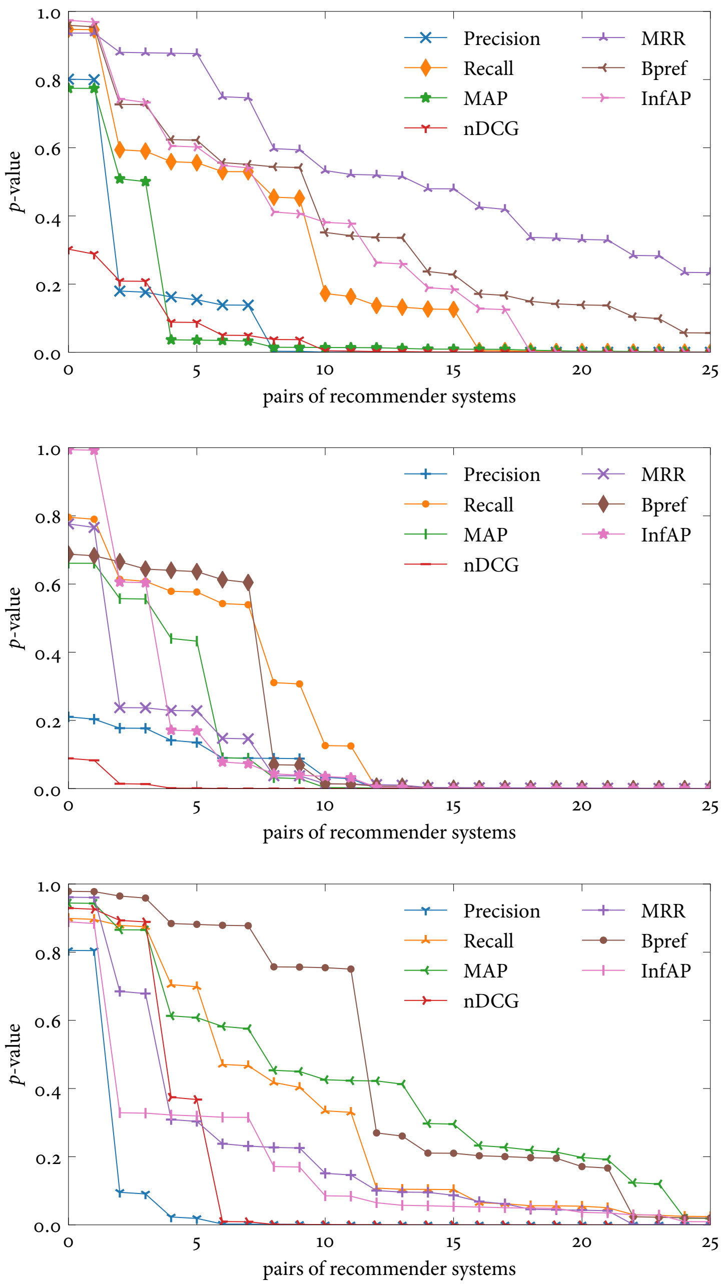


Figure 4.8: Analysis of the discriminative power of Precision, Recall, MAP, nDCG, MRR, bpref and InfAP (using a cut-off of 100) on the MovieLens 1M (top), LibraryThing (middle) and BeerAdvocate (bottom) datasets.

Precision and nDCG offer the best robustness and discriminative power.

Our findings also suggest that precision, a simple binary metric, is very robust to sparsity and popularity biases. Normalized discounted cumulative gain also presents high robustness to the sparsity bias and moderate robustness to the popularity bias. Moreover, in terms of discriminative power, nDCG and to a lesser degree precision showed the best figures of all the tested metrics. For these reasons, we use these metrics for evaluating recommender systems in this thesis.

Surprisingly, we found that bpref and InfAP—which were proposed to address incompleteness in IR— as well as MRR perform poorly in RS evaluation and, therefore, we discourage the use of these metrics.

Part III

PSEUDO-RELEVANCE FEEDBACK MODELS FOR RECOMMENDER SYSTEMS

May, in spite of all distractions generated by technology, all of you succeed in turning information into knowledge, knowledge into understanding, and understanding into wisdom.

— Edsger W. Dijkstra,
Edsger Dijkstra on universities

5

RELEVANCE MODELS FOR USER-BASED RECOMMENDATION

The work of Parapar (2013) and Parapar et al. (2013) constitutes one of the most effective adaptations of information retrieval models to recommender systems. In this work, they established a parallelism between pseudo-relevance feedback and user-based collaborative filtering. In particular, they used relevance models (RM), a state-of-the-art PRF technique based on the language modeling framework, as a neighborhood-based recommender.

In this chapter, we thoroughly study the possibilities of relevance models as user-based CF recommenders. Smoothing is a central issue in the estimation of language models (which RM are based on). Probability estimators require smoothing methods to compensate for data sparsity. On the other hand, the probabilistic modeling of recommender systems using relevance models naturally introduces the concept of user and item prior probabilities into the recommendation task.

We examine the effectiveness of relevance models as user-based collaborative filtering recommenders.

Neither the influence of different smoothing methods nor prior probability estimators have been studied in the context of recommender systems. Therefore, in this chapter, we explore different smoothing approaches from a theoretical and empirical point of view. We also study the effect of different estimators for the item and user prior probabilities under this framework.

The contributions presented in this chapter have been previously published. Valcarce et al. (2015c) presents an empirical study of three smoothing methods for relevance models applied to recommendation. Later, Valcarce et al. (2015b) conducts an empirical study of different prior estimators. Finally, Valcarce et al. (2016a) performs a theoretical study based on axiomatic analysis with the three previous smoothing methods and another one.

Pseudo-relevance feedback	User-based collaborative filtering
Query	Target user: u
Query terms	Items rated by the target user: $\mathcal{I}_u$
Pseudo-relevant set	User neighborhood: $\mathcal{V}_u$
Candidate terms for expansion	Candidate items for recommendation

Table 5.1: Parallelism between pseudo-relevance feedback and user-based collaborative filtering.

5.1 RELEVANCE MODELS AS RECOMMENDERS

Parapar (2013) and Parapar et al. (2013) drew a parallelism between pseudo-relevance feedback and user-based collaborative filtering. The connection between these tasks from different fields allowed them to adapt relevance-based language models or, simply, relevance models (RM) to user-based collaborative filtering recommendation. The authors showed that the task of recommending items to a user could be assimilated to the task of expanding a query with new terms. Instead of a query, we have a user whose profile (i.e., the set of items that the user has rated) has to be expanded with new relevant items. Since PRF models exploit a set of pseudo-relevant documents to extract terms, we can use the neighborhood of the target user as the pseudo-relevant set. In this way, users play a dual role: they act as queries when they are the target user of the recommendation process, but they also act as documents of the pseudo-relevant set when they are neighbors. On the other hand, items only play the role of terms. Note that the last step of pseudo-relevance feedback (performing a second retrieval with the expanded query) is not needed because the expansion items for the users' profiles are our objective. Table 5.1 summarizes the parallelism between PRF and user-based CF.

Lavrenko and Croft (2001) proposed two methods for estimating the relevance models in IR: RM1 and RM2. The first method uses i.i.d. sampling whereas the second is based on conditional sampling. To use RM as a recommendation model, we need to assume that, for all pairs of target user u and neighborhood V_u, an underlying relevance model R_u exists. The recommendations for a user u are generated by computing the relevance model of that user, R_u, and estimating the relevance of each item i under it as follows:

$$\text{RM1:} \quad p(i|R_u) \propto \sum_{v \in V_u} p(v) p(i|v) \prod_{j \in \mathcal{I}_u} p(j|v) \tag{5.1}$$

$$\text{RM2:} \quad p(i|R_u) \propto p(i) \prod_{j \in \mathcal{I}_u} \sum_{v \in V_u} \frac{p(i|v)p(v)}{p(i)} p(j|v) \tag{5.2}$$

Recommendations are presented to the user ordered according to decreasing estimated relevance, that is, decreasing values of $p(i|R_u)$. Parapar et al. (2013) and Valcarce et al. (2015c) showed that RM2 is superior to RM1 in recommendation effectiveness. Conversely, RM1 outperforms RM2 in query expansion (Lavrenko and Croft 2001). Therefore, in this chapter, we focus on RM2 model alone.

RM2 outperforms RM1 in top-N recommendation.

The neighborhood V_u is the set of similar users to u. The most common practice in the literature is to use the kNN algorithm (Ning et al. 2015). This method finds the k most similar users to the target user using a pairwise metric. Parapar et al. (2013) used kNN with Pearson's correlation coefficient as the pairwise metric to compute user neighborhoods for RM. However, Cremonesi et al. (2010) showed that cosine similarity is better suited for top-N recommendation while Pearson's correlation works best in the rating prediction task. In fact, we verified that the kNN algorithm with cosine similarity provides better neighborhoods than with Pearson's correlation coefficient with the RM2 recommendation model (Valcarce et al. 2016a). Therefore, to compute similarities between users u and v, we use the cosine similarity as follows:

Neighborhoods are typically computed using the kNN algorithm.

Cosine similarity yields better results than Pearson's in top-N recommendation.

$$\cos(u, v) = \frac{\sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} r(u,i)\, r(v,i)}{\sqrt{\sum_{i \in \mathcal{I}_u} r(u,i)^2} \sqrt{\sum_{i \in \mathcal{I}_v} r(v,i)^2}} \tag{5.3}$$

The probability of an item i given a user u, $p(i|u)$, can be computed by the maximum likelihood estimate (MLE) of a multinomial distribution of ratings:

$$p_{mle}(i|u) = \frac{r(u,i)}{\sum_{j \in \mathcal{I}_u} r(u,j)} \tag{5.4}$$

The problem of the maximum likelihood estimate stems from its high sparsity: if a user did not rate an item, the estimate yields a value of zero. For this reason, language models are smoothed. In Section 5.2, we study different smoothing methods for the maximum likelihood estimate of relevance models applied to recommendation.

Finally, we need to specify how to calculate item prior probability $p(i)$ and user prior probability $p(v)$ estimates. Parapar et al. (2013) used uniform estimates, but in Section 5.3, we present different alternatives.

5.2 STUDY OF SMOOTHING METHODS

Smoothing improves the effectiveness by avoiding zeros in the MLE.

Language models use smoothed probability estimates to achieve high effectiveness. The selection of the smoothing method is crucial for the performance of language models both in ad hoc retrieval (Zhai and Lafferty 2004) and in top-N recommendation (Valcarce et al. 2015c, 2016a). The main effects of smoothing in information retrieval are: on the one hand, preventing the apparition of zeros due to the data sparsity and, on the other hand, adding the effect of the IDF (inverse document frequency) (Zhai and Lafferty 2004). Additionally, some smoothing techniques also provide the effect of length normalization (Losada and Azzopardi 2008a). Next, we present different smoothing methods to use in recommendation.

5.2.1 *Smoothing methods*

In ad hoc retrieval, the most common practice is to smooth the MLE with the collection model (Zhai and Lafferty 2004). Using a collection-based smoothing method, the probability of an item given a user $p(i|u)$ is calculated by smoothing the maximum likelihood estimate $p_{mle}(i|u)$ with the background model of the collection $p(i|\mathcal{C})$. This collection model is given by:

$$p(i|\mathcal{C}) = \frac{\sum_{v \in \mathcal{U}} r(v, i)}{\sum_{j \in \mathcal{I}} \sum_{v \in \mathcal{U}} r(v, j)} \tag{5.5}$$

Next, we present three collection-based smoothing methods: Jelinek-Mercer (JMS), Dirichlet priors (DPS) and absolute discounting smoothing (ADS) (Zhai and Lafferty 2004). Additionally, we present additive smoothing (AS) which is collection-agnostic. Collection-based smoothing methods have in common that they substitute part of the probability mass provided by the MLE with probability mass obtained from the collection model. This reallocation of probability is performed to avoid zeros in non-rated items (Zhai and Lafferty 2004). Conversely, collection-agnostic smoothing methods reallocate the probability mass in a way that is independent of the collection.

5.2.1.1 *Jelinek-Mercer smoothing (JMS)*

This method performs a linear interpolation between the maximum likelihood estimator and the collection model (Jelinek and Mercer 1980) which is regulated by the parameter $\lambda \in [0, 1]$:

$$p_{\lambda}(i|u) = (1 - \lambda)\, p_{mle}(i|u) + \lambda\, p(i|\mathcal{C}) \tag{5.6}$$

5.2.1.2 *Dirichlet priors smoothing (DPS)*

Derived from a Bayesian analysis using Dirichlet priors (MacKay and Peto 1995), this method has a parameter $\mu > 0$ to control the amount of smoothing applied:

$$p_\mu(i|u) = \frac{r(u,i) + \mu\, p(i|\mathcal{C})}{\mu + \sum_{j \in \mathcal{I}_u} r(u,j)} \tag{5.7}$$

5.2.1.3 *Absolute discounting smoothing (ADS)*

This method subtracts a value of $\delta > 0$ from the count of the rated items (Ney et al. 1994). This discount is compensated with the background collection:

$$p_\delta(i|u) = \frac{\max[r(u,i) - \delta, 0] + \delta\,|\mathcal{I}_u| p(i|\mathcal{C})}{\sum_{j \in \mathcal{I}_u} r(u,j)} \tag{5.8}$$

5.2.1.4 *Additive smoothing (AS)*

Additive smoothing (also known as Laplace smoothing) is a collection-agnostic method. This smoothing method increases all the ratings by a parameter $\gamma > 0$. If the user u has not rated the item i, that item will receive a rating value of γ. The probability estimate with this method is computed as follows:

$$p_\gamma(i|u) = \frac{r(u,i) + \gamma}{\sum_{j \in \mathcal{I}_u} r(u,j) + \gamma\,|\mathcal{I}|} \tag{5.9}$$

5.2.2 *Theoretical analysis of smoothing methods*

An axiomatic analysis is an effective tool for studying language models formally (Fang et al. 2004; Hazimeh and Zhai 2015; Valcarce et al. 2016a). Hazimeh and Zhai (2015) presented an axiomatic analysis of smoothing methods for different pseudo-relevance feedback techniques. They found that applying collection-based smoothing methods to RM1 demotes the IDF effect—a desired property of a retrieval system. Instead, they proposed the use of relevance models with additive smoothing.

We claim that the IDF, which is a fundamental term measure in IR, is related to novelty in RS. Next, we study the connection between these two concepts and its implications in recommendation.

5.2.2.1 *IDF effect and novelty*

IDF is the fundamental term specificity measure.

In information retrieval , the inverse document frequency (IDF) is a measure of term specificity that has become a cornerstone of term weighting (Robertson 2004; Spärck Jones 1972). It is defined as the inverse of the number of documents in the collection that contains the target term. For example, *stopwords* are terms that appear in many document but do not provide much information. In contrast, those terms that only appear in a few documents tend to be highly informative and help in discriminating which documents are relevant. Thus, the IDF effect gives more importance to those query terms that are more specific (i.e., those terms with a higher IDF).

IDF was not born from a formal analysis; however, it was considered a useful and robust heuristic (Spärck Jones 1972). Later, Robertson (2004) provided a theoretical justification for this term weighting function. Mostly all the text retrieval algorithms introduce the IDF effect to weight query terms (Spärck Jones 1972). This property can be included in the retrieval model either explicitly (e. g., the vector space model or BM25) or implicitly (e. g., the probabilistic model or language models).

We claim that when adapting the relevance modeling framework to collaborative filtering, term specificity is related to item novelty. The IDF effect promotes specific terms over popular—and to some extent meaningless—ones. Since items play the role of terms when using relevance models as recommender systems, promoting uncommon terms should be beneficial for improving novelty figures.

Previous work has explored different estimations of RM that promote divergent terms with great success (Carpineto et al. 2001; Parapar and Barreiro 2011). In particular, Hazimeh and Zhai (2015) performed an axiomatic analysis of the IDF effect in several pseudo-relevance feedback methods. They found that collection-based methods penalize the IDF effect on RM1. To overcome this problem, they proposed to use additive smoothing which does not rely on a background collection model. Their analysis showed that this type of smoothing neither promotes nor demotes the IDF effect. However, it is not clear whether this conclusion applies to RM2. For this reason, we perform an axiomatic analysis of the IDF effect on RM2 in the context of recommender systems.

5.2.2.2 *Formalization of the IDF effect in recommendation*

Our goal is to examine different smoothing methods for RM2 in the context of RS. In recommendation, given two items with the same ratings in the neighborhood, the IDF effect promotes the item that is less popular in the collection. The popularity of an item is measured as its probability in the collection as Equation (5.5) defines. This property is desirable in order to enhance the novelty of the recommendations while keeping high accuracy. This effect does not conflict with accuracy because uncommon items are preferred over common items only when they have the same ratings. Formally, we can define the IDF effect in recommender systems as follows:

Definition (IDF effect)**.** Let u be a user from the set of users $\mathcal{U}$ and V_u be her/his neighborhood. Given two items i_1 and i_2 with the same ratings $r(v, i_1) = r(v, i_2) \forall\ v \in V_u$ and different popularity $p(i_1|\mathcal{C}) < p(i_2|\mathcal{C})$, a recommender system that outputs $p(i_1|R_u) > p(i_2|R_u)$ is said to support the IDF effect.

Now we proceed to analyze RM2 axiomatically. If we assume that i_1 and i_2 are two items as in the previous definition, studying the sign of $\Delta = p(i_1|R_u) - p(i_2|R_u)$ allows to check whether RM2 supports the IDF effect. If $\Delta > 0$, the recommender system supports this property. On the contrary, if $\Delta < 0$, the algorithm violates the definition of the IDF effect. Finally, $\Delta = 0$ means that the system neither promotes nor demotes the IDF effect. Given the formula of RM2, Δ is computed as follows:

$$\begin{aligned}
\Delta &= p(i_1|R_u) - p(i_2|R_u) \\
&= p(i_1) \prod_{j \in \mathcal{I}_u} \sum_{v \in V_u} \frac{p(i_1|v)p(v)}{p(i_1)} p(j|v) \\
&\quad - p(i_2) \prod_{j \in \mathcal{I}_u} \sum_{v \in V_u} \frac{p(i_2|v)p(v)}{p(i_2)} p(j|v) \qquad (5.10)
\end{aligned}$$

If we suppose that item priors are uniform, we obtain:

$$\begin{aligned}
\Delta = p(i) \prod_{j \in \mathcal{I}_u} \sum_{v \in V_u} \frac{p(j|v)p(v)}{p(i)} p(i_1|v) \\
- p(i) \prod_{j \in \mathcal{I}_u} \sum_{v \in V_u} \frac{p(j|v)p(v)}{p(i)} p(i_2|v) \qquad (5.11)
\end{aligned}$$

We can observe that the sign of Δ depends on the sign of $p(i_1|v) - p(i_2|v)$ which may vary among smoothing methods. Therefore, we need to analyze each smoothing technique one by one. Next, present an axiomatic analysis for each smoothing method.

5.2.2.3 *Analysis of Jelinek-Mercer smoothing*

We apply Jelinek-Mercer smoothing method from Equation (5.6) to Equation (5.11):

$$\begin{aligned}\Delta = p(i) \prod_{j \in \mathcal{I}_u} \sum_{v \in V_u} \frac{p(j|v)p(v)}{p(i)} \big[(1-\lambda) p_{ml}(i_1|v) + \lambda p(i_1|\mathcal{C})\big] \\ - p(i) \prod_{j \in \mathcal{I}_u} \sum_{v \in V_u} \frac{p(j|v)p(v)}{p(i)} \big[(1-\lambda) p_{ml}(i_2|v) + \lambda p(i_2|\mathcal{C})\big] \\ < 0 \end{aligned} \tag{5.12}$$

and we obtain that the difference is negative because $\lambda \in [0, 1]$, all the probabilities are positive and $p(i_1|C) < p(i_2|C)$ from definition. Note that $p_{ml}(i_1|u) = p_{ml}(i_2|u)$ because both items have the same ratings. Thus, Jelinek-Mercer demotes the IDF effect for RM2 as it does for RM1 in pseudo-relevance feedback (Hazimeh and Zhai 2015).

5.2.2.4 *Analysis of Dirichlet priors smoothing*

We plug the expression of DPS from Equation (5.7) into Equation (5.11) as follows:

$$\begin{aligned}\Delta = p(i) \prod_{j \in \mathcal{I}_u} \sum_{v \in V_u} \frac{p(j|v)p(v)}{p(i)} \frac{r(v, i_1) + \mu p(i_1|\mathcal{C})}{\mu + \sum_{k \in \mathcal{I}_u} r(v, k)} \\ - p(i) \prod_{j \in \mathcal{I}_u} \sum_{v \in V_u} \frac{p(j|v)p(v)}{p(i)} \frac{r(v, i_2) + \mu p(i_2|\mathcal{C})}{\mu + \sum_{k \in \mathcal{I}_u} r(v, k)} \\ < 0 \end{aligned} \tag{5.13}$$

and we obtain that the difference is also negative because $\mu > 0$, all the ratings and probabilities are positive and, by definition, $p(i_1|C) < p(i_2|C)$. We can conclude that Dirichlet priors smoothing violates the IDF effect for RM2. This also happens for RM1 in pseudo-relevance feedback (Hazimeh and Zhai 2015).

5.2.2.5 *Analysis of absolute discounting smoothing*

ADS was not studied in the context of pseudo-relevance feedback; however, since we found that is preferred collection-based smoothing method for RM2 in recommendation (Valcarce et al. 2015c), we analyze if it supports the IDF effect:

$$\begin{aligned}
\Delta &= p(i) \prod_{j \in \mathcal{I}_u} \sum_{v \in V_u} \frac{p(j|v)p(v)}{p(i)} \frac{r_\delta(v, i_1) + \delta|\mathcal{I}_v| p(i_1|\mathcal{C})}{\sum_{k \in \mathcal{I}_v} r(u,k)} \\
&\quad - p(i) \prod_{j \in \mathcal{I}_u} \sum_{v \in V_u} \frac{p(j|v)p(v)}{p(i)} \frac{r_\delta(v, i_2) + \delta|\mathcal{I}_v| p(i_2|\mathcal{C})}{\sum_{k \in \mathcal{I}_v} r(u,k)} \\
&< 0 \qquad (5.14)
\end{aligned}$$

where $r_\delta(v,i) = \max[r_\delta(v,i) - \delta, 0]$. We can observe that the difference Δ is negative taking into account that $\delta > 0$, $|\mathcal{I}_v| > 0$, all the ratings are positive and, by definition, $p(i_1|C) < p(i_2|C)$.

We can observe that the three collection-based smoothing methods demote the IDF effect on RM2 for recommendation. For this reason, next, we also explore additive smoothing as a collection-agnostic smoothing method.

5.2.2.6 *Analysis of additive smoothing*

Since this method is collection-agnostic, it does not rely on the probability of an item in the collection, $p(i|\mathcal{C})$, which is a measure of item popularity and, thus, opposed to novelty. Applying the same axiomatic analysis as before:

$$\begin{aligned}
\Delta &= p(i) \prod_{j \in \mathcal{I}_u} \sum_{v \in V_u} \frac{p(j|v)p(v)}{p(i)} \frac{r(u, i_1) + \gamma}{\sum_{j \in \mathcal{I}_u} r(u,j) + \gamma|\mathcal{I}|} \\
&\quad - p(i) \prod_{j \in \mathcal{I}_u} \sum_{v \in V_u} \frac{p(j|v)p(v)}{p(i)} \frac{r(u, i_2) + \gamma}{\sum_{j \in \mathcal{I}_u} r(u,j) + \gamma|\mathcal{I}|} \\
&= 0 \qquad (5.15)
\end{aligned}$$

we find that this method neither supports nor violates the IDF effect. This result coincides with the analysis of RM1 for pseudo-relevance feedback (Hazimeh and Zhai 2015).

5.2.2.7 *Discussion of the axiomatic analysis*

Our axiomatic analysis proves that the aforementioned collection-based smoothing methods (JMS, DPS and ADS) demote the IDF effect on RM2 in recommendation as it does on RM1 for pseudo-relevance feedback. Moreover, we find that additive smoothing neither promotes nor demotes the IDF effect on RM2. Thus, if the IDF heuristic is valuable for recommendation, additive smoothing should work better than the other methods. Next, we conduct an empirical study of the smoothing methods to verify this hypothesis.

5.2.3 *Experimental evaluation of smoothing methods*

We test experimentally the quality of the recommendations generated by RM2 with different smoothing methods. We used the MovieLens 100k and 1M, R3-Yahoo and LibraryThing datasets. We tuned the smoothing parameters $\gamma \in \{0.001, 0.01, 0.1, 1, 10\}$, $\delta \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$, $\lambda \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ and $\mu \in \{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000\}$ of the AS, ADS, JMS and DPS methods, respectively. We used *k*NN algorithm with cosine similarity for computing neighborhoods. We found that the optimal values with respect to nDCG@100 were $k = 50$ for the MovieLens 100k, $k = 75$ for the MovieLens 1M, $k = 150$ for the R3-Yahoo and $k = 50$ for the LibraryThing dataset. The results in terms of nDCG@100, Gini@100 and MSI@100 for each dataset are presented in Figures 5.1 to 5.4.

As we can observe, smoothing is a crucial aspect of RM2: the choice of the smoothing method, as well as its correct parameter optimization, affects the final quality of the recommendations notably. Although the absolute values of performance vary, the trends in both datasets are very similar. This supports the generalization of these results to other collections.

Overall, additive smoothing provides the best recommendations in terms of precision, diversity and novelty followed by absolute discounting smoothing. In recommendation, there is always a trade-off between accuracy and diversity or novelty (Zhou et al. 2010b). It is straightforward to improve the diversity or novelty of the recommendations at the expense of a reduction of accuracy (e.g., recommending very unpopular items to different users). Therefore, simultaneous improvements in accuracy and in novelty or diversity are highly valuable. Additive smoothing obtains notable improvements on these three aspects. This supports the importance of the IDF effect in recommendation.

An important property of Additive smoothing is the stability to the changes in its parameter. We used a logarithmic scale to visualize very large variations of the parameter γ. This method only showed a small decrease in accuracy, novelty and diversity when we used enormous values of γ. Absolute Discounting Smoothing (ADS) also showed quite stable results, but the method deteriorates with a high amount of smoothing.

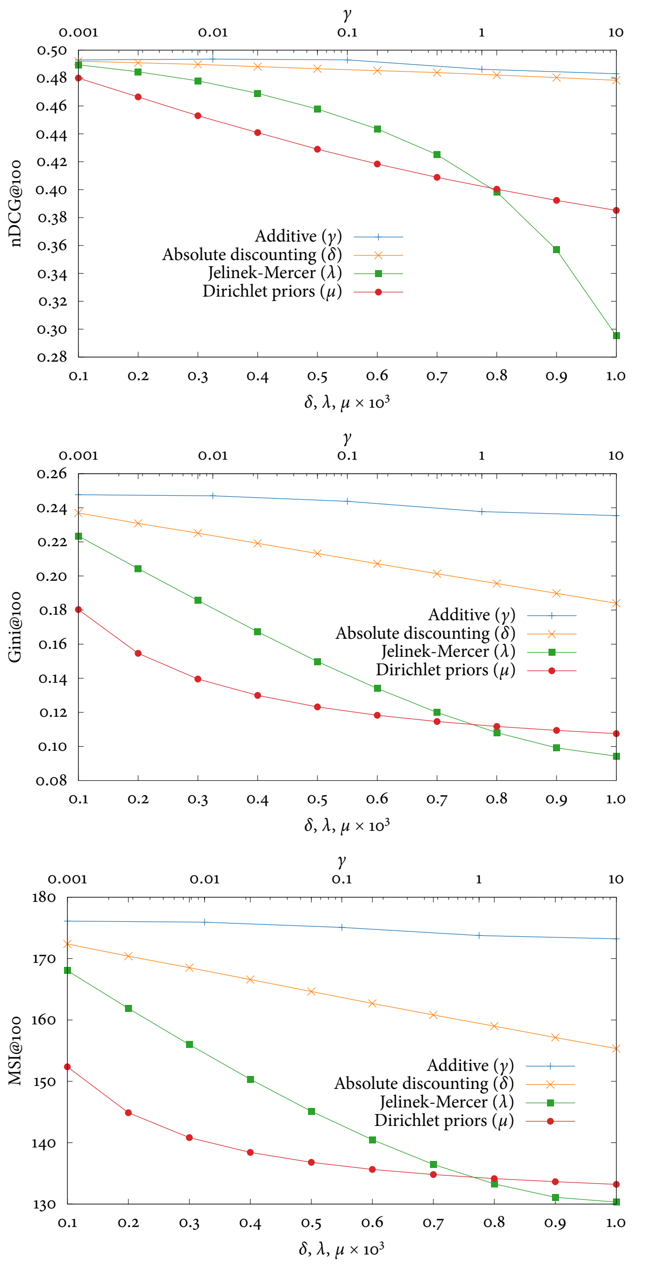


Figure 5.1: Values of nDCG@100 (top), Gini@100 (middle) and MSI@100 (bottom) for RM2 using AS, ADS, JMS and DPS methods on the MovieLens 100k dataset varying the smoothing parameters. Neighborhoods are computed taking the 50 closest users according to cosine similarity.

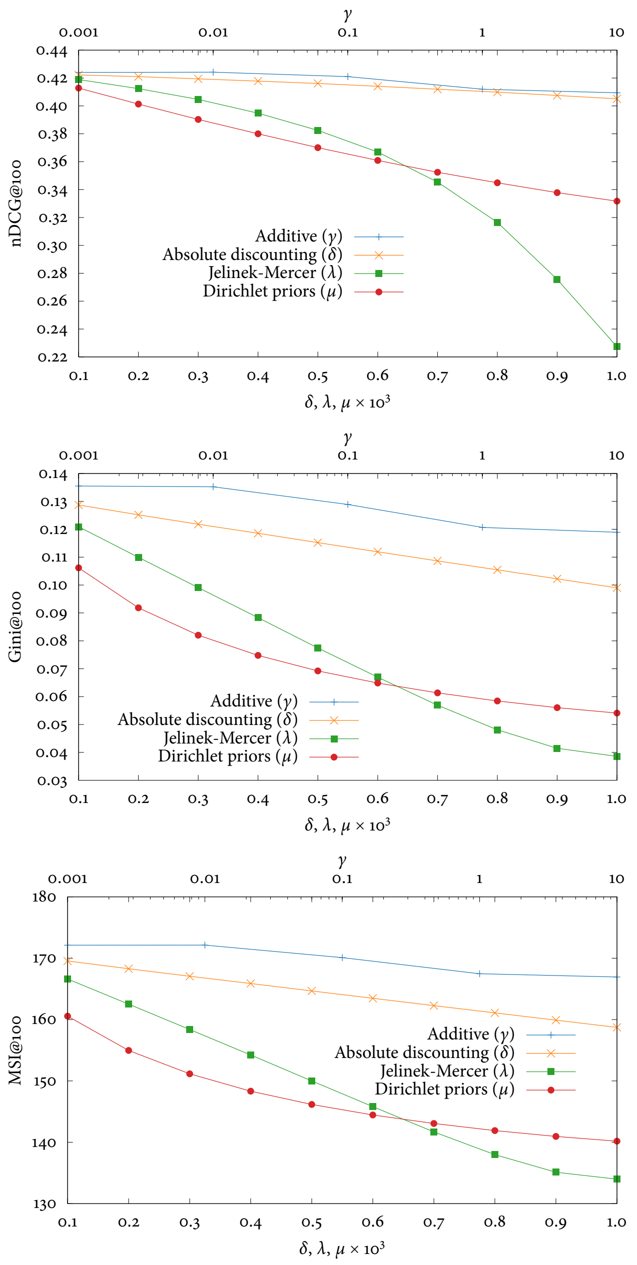


Figure 5.2: Values of nDCG@100 (top), Gini@100 (middle) and MSI@100 (bottom) for RM2 using AS, ADS, JMS and DPS methods on the MovieLens 1M dataset varying the smoothing parameters. Neighborhoods are computed taking the 75 closest users according to cosine similarity.

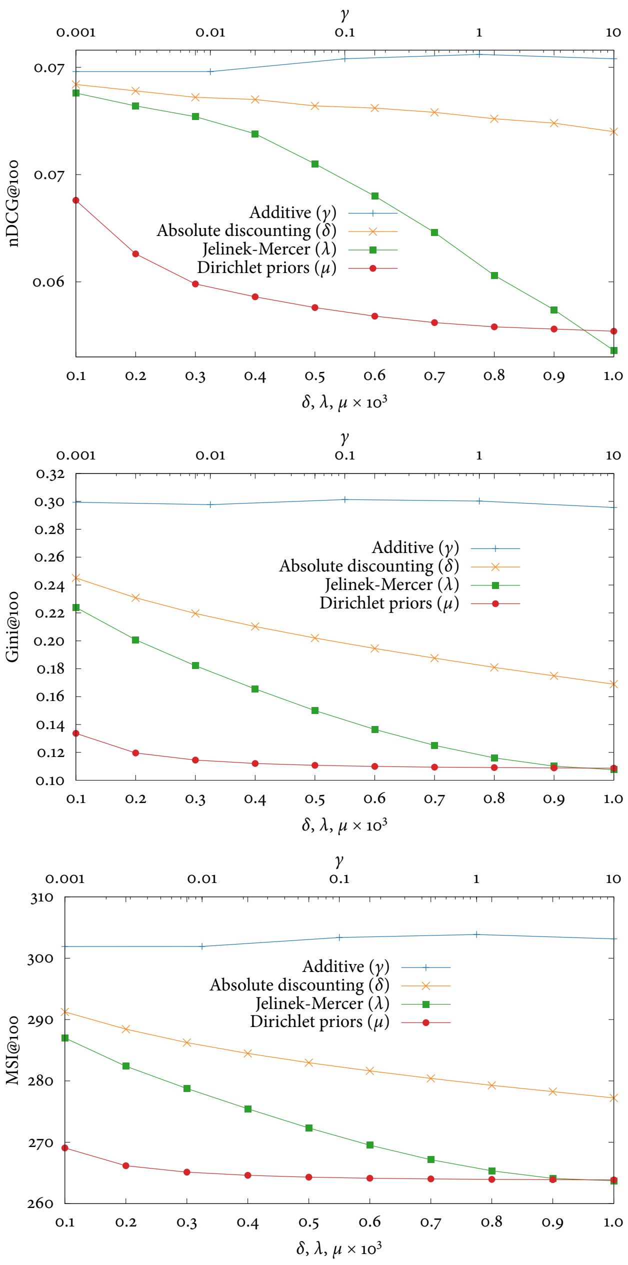


Figure 5.3: Values of nDCG@100 (top), Gini@100 (middle) and MSI@100 (bottom) for RM2 using AS, ADS, JMS and DPS methods on the R3-Yahoo dataset varying the smoothing parameters. Neighborhoods are computed taking the 150 closest users according to cosine similarity.

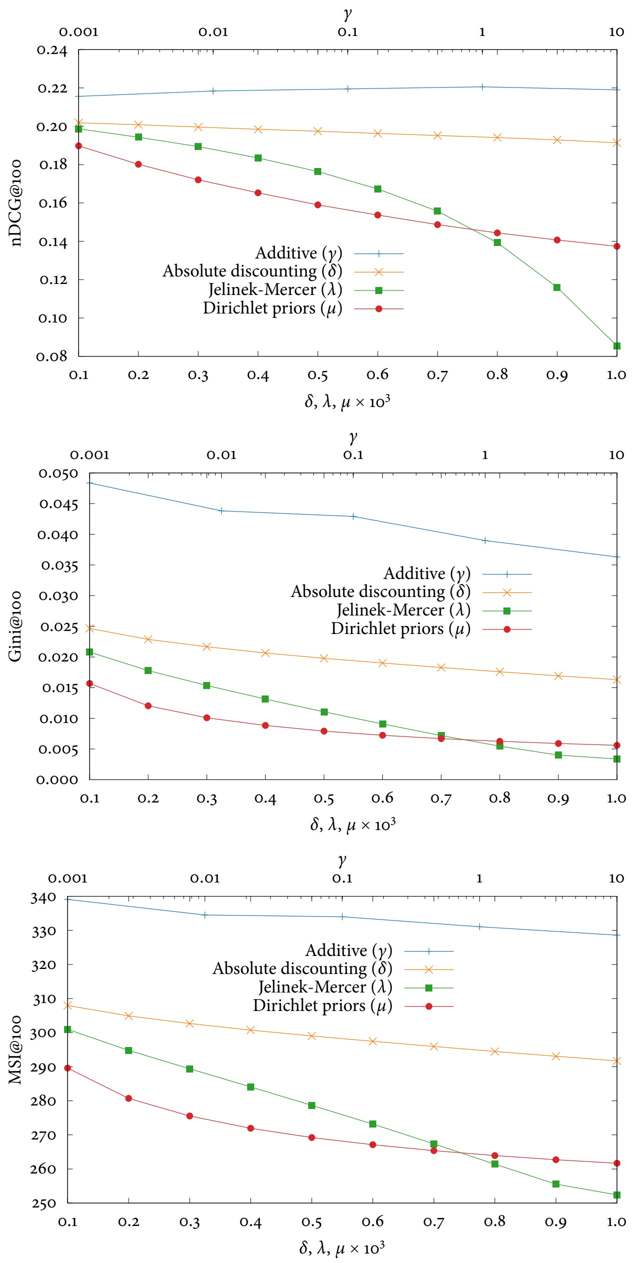


Figure 5.4: Values of nDCG@100 (top), Gini@100 (middle) and MSI@100 (bottom) for RM2 using AS, ADS, JMS and DPS methods on the LibraryThing dataset varying the smoothing parameters. Neighborhoods are computed taking the 50 closest users according to cosine similarity.

In contrast, the performance of Jelinek-Mercer and Dirichlet priors is far lower than the rest. Additionally, the computational complexity of the three collection-based methods is the same. Thus, there is no reason to consider using these smoothing methods with RM2 for recommendation.

Additive smoothing also presents another advantage over ADS (and also over JMS and DPS): it does not depend on collection statistics. This fact not only preserves the IDF effect but also reduces the memory consumption since we do not require to maintain global statistics of the collection.

To conclude our study of smoothing methods, we wish to emphasize that these empirical results agree with the axiomatic analysis: additive smoothing is the preferred method for smoothing the MLE in RM2.

5.3 STUDY OF PRIOR ESTIMATORS

Information retrieval algorithms often include the notion of a *document prior* which encodes the importance of a document independently of the user's query. These priors can be used for improving the document ranking effectiveness (Blanco and Barreiro 2008; Brin and Page 1998; Kraaij et al. 2002; Peng and Ounis 2007). Probably, the most famous example of a document prior in IR is PageRank, a web-based document prior that measures the relative importance of a website (Brin and Page 1998).

A prior is a probability distribution that expresses the beliefs about a quantity before considering evidence.

The use of probabilistic models such as RM for recommendation provides several advantages. One of them is the possibility of introducing prior probabilities into the recommendation process. In fact, the most effective estimation of relevance models in recommendation is RM2, which includes a user prior and an item prior. Previous works on the relevance modeling of recommender systems considered those priors uniform (Bellogín et al. 2013b; Parapar et al. 2013; Valcarce et al. 2015c, 2016a,d) leaving open the possibility of further studying this aspect. For this reason, we analyze here the effects of the user and the item priors in the RM2 model. First, we adapt two effective document length priors from IR to RS (Blanco and Barreiro 2008; Kraaij et al. 2002). Then, we propose two new variants of the probabilistic length prior devised by Blanco and Barreiro (2008). Finally, we conduct a series of experiments that show that the use of a linear length prior for the users and a probabilistic length prior based on Dirichlet smoothing for the items leads to significant improvements in terms of ranking accuracy.

5.3.1 *Prior estimators*

Equation (5.2) shows the recommendation formula that results from adapting RM2 to user-based recommendation. This probabilistic model involves a user prior for each neighbor and an item prior for each candidate item to be recommended. In the following, we study the different prior estimators.

5.3.1.1 *Uniform prior (U)*

This prior is drawn from a uniform distribution. That is to say, every user/item in the population has the same prior probability. We use this prior as our baseline.

$$p_U(u) = \frac{1}{|\mathcal{U}|} \tag{5.16}$$

$$p_U(i) = \frac{1}{|\mathcal{I}|} \tag{5.17}$$

5.3.1.2 *Linear prior (L)*

The linear document length prior was previously used in information retrieval (Blanco and Barreiro 2008; Kraaij et al. 2002). Its adaptation to recommendation boosts those users/items with larger profiles. In this way, we are promoting the recommendations that came from the power users of the system or items with high popularity.

$$p_L(u) = p(u|\mathcal{C}) = \frac{\sum_{i\in\mathcal{I}_u} r(u,i)}{\sum_{v\in\mathcal{U}} \sum_{j\in\mathcal{I}_v} r(v,j)} \tag{5.18}$$

$$p_L(i) = p(i|\mathcal{C}) = \frac{\sum_{u\in\mathcal{U}_i} r(u,i)}{\sum_{j\in\mathcal{I}} \sum_{v\in\mathcal{U}_j} r(v,j)} \tag{5.19}$$

5.3.1.3 *Probabilistic prior using Jelinek-Mercer smoothing (PJMS)*

An effective IR prior is the probabilistic document length prior proposed by (Blanco and Barreiro 2008). It is indirectly based on the document length. In recommendation, this estimator computes the prior probability of the users as a function of the statistics of the items they rated. Likewise, the prior probability of an item is a function of the statistics of the users who rated them. The original formulation of this prior employs Jelinek-Mercer smoothing:

$$\begin{aligned} p_{PJMS}(u) &\propto \sum_{i \in \mathcal{I}_u} p_\lambda(i|u) \\ &= \sum_{i \in \mathcal{I}_u} \left[(1-\lambda) \frac{r(u,i)}{\sum_{j \in \mathcal{I}_u} r(u,j)} + \lambda p(i|\mathcal{C}) \right] \\ &= (1-\lambda) + \lambda \sum_{i \in \mathcal{I}_u} p(i|\mathcal{C}) \end{aligned} \tag{5.20}$$

$$\begin{aligned} p_{PJMS}(i) &\propto \sum_{u \in \mathcal{U}_i} p_\lambda(u|i) \\ &= \sum_{u \in \mathcal{U}_i} \left[(1-\lambda) \frac{r(u,i)}{\sum_{v \in \mathcal{U}_i} r(v,i)} + \lambda p(u|\mathcal{C}) \right] \\ &= (1-\lambda) + \lambda \sum_{u \in \mathcal{U}_i} p(u|\mathcal{C}) \end{aligned} \tag{5.21}$$

5.3.1.4 *Probabilistic prior using Dirichlet priors smoothing (PDPS)*

In this work, we also propose to explore the previous probabilistic prior using Dirichlet smoothing:

$$\begin{aligned} p_{PDPS}(u) &\propto \sum_{i \in \mathcal{I}_u} p_\mu(i|u) \\ &= \sum_{i \in \mathcal{I}_u} \frac{r(u,i) + \mu p(i|\mathcal{C})}{\mu + \sum_{j \in \mathcal{I}_u} r(u,j)} \\ &= \frac{\sum_{i \in \mathcal{I}_u} r(u,i) + \mu \sum_{i \in \mathcal{I}_u} p(i|\mathcal{C})}{\mu + \sum_{i \in \mathcal{I}_u} r(u,i)} \end{aligned} \tag{5.22}$$

$$\begin{aligned} p_{PDPS}(i) &\propto \sum_{u \in \mathcal{U}_i} p_\mu(u|i) \\ &= \sum_{u \in \mathcal{U}_i} \frac{r(u,i) + \mu p(u|\mathcal{C})}{\mu + \sum_{v \in \mathcal{U}_i} r(v,i)} \\ &= \frac{\sum_{u \in \mathcal{U}_i} r(u,i) + \mu \sum_{u \in \mathcal{U}_i} p(u|\mathcal{C})}{\mu + \sum_{u \in \mathcal{U}_i} r(u,i)} \end{aligned} \tag{5.23}$$

5.3.1.5 *Probabilistic prior using absolute discounting smoothing (PADS)*

The same prior as before but using absolute discounting smoothing:

$$\begin{aligned} p_{PADS}(u) &\propto \sum_{i \in \mathcal{I}_u} p_\delta(i|u) \\ &= \sum_{i \in \mathcal{I}_u} \frac{\max(r(u,i) - \delta, 0) + \delta |\mathcal{I}_u| p(i|\mathcal{C})}{\sum_{j \in \mathcal{I}_u} r(u,j)} \\ &= \frac{\sum_{i \in \mathcal{I}_u} \max(r(u,i) - \delta, 0) + \delta |\mathcal{I}_u| \sum_{i \in \mathcal{I}_u} p(i|\mathcal{C})}{\sum_{j \in \mathcal{I}_u} r(u,j)} \end{aligned} \tag{5.24}$$

$$\begin{aligned} p_{PADS}(i) &\propto \sum_{u \in \mathcal{U}_i} p_\delta(u|i) \\ &= \sum_{u \in \mathcal{U}_i} \frac{\max(r(u,i) - \delta, 0) + \delta |\mathcal{U}_i| p(u|\mathcal{C})}{\sum_{v \in \mathcal{U}_i} r(v,i)} \\ &= \frac{\sum_{u \in \mathcal{U}_i} \max(r(u,i) - \delta, 0) + \delta |\mathcal{U}_i| \sum_{u \in \mathcal{U}_i} p(u|\mathcal{C})}{\sum_{v \in \mathcal{U}_i} r(v,i)} \end{aligned} \tag{5.25}$$

5.3.1.6 *Probabilistic prior using additive smoothing (PAS)*

Finally, we also test additive smoothing in the probabilistic prior:

$$\begin{aligned} p_{PAS}(u) &\propto \sum_{i \in \mathcal{I}_u} p_\gamma(i|u) \\ &= \sum_{i \in \mathcal{I}_u} \frac{r(u,i) + \gamma}{\sum_{j \in \mathcal{I}_u} r(u,j) + \gamma |\mathcal{I}|} \\ &= \frac{\sum_{i \in \mathcal{I}_u} r(u,i) + \gamma |\mathcal{I}_u|}{\sum_{j \in \mathcal{I}_u} r(u,j) + \gamma |\mathcal{I}|} \end{aligned} \tag{5.26}$$

$$\begin{aligned} p_{PAS}(i) &\propto \sum_{u \in \mathcal{U}_i} p(u|i) \\ &= \sum_{u \in \mathcal{U}_i} \frac{r(u,i) + \gamma}{\sum_{v \in \mathcal{U}_i} r(v,i) + \gamma |\mathcal{U}|} \\ &= \frac{\sum_{u \in \mathcal{U}_i} r(u,i) + \gamma |\mathcal{U}_i|}{\sum_{v \in \mathcal{U}_i} r(v,i) + \gamma |\mathcal{U}|} \end{aligned} \tag{5.27}$$

5.3.2 *Experimental evaluation of prior estimators*

In these experiments, we use RM2 with the optimal hyperparameters found in Section 5.2.3. In particular, we use the *k*NN algorithm with co-

RM2	Metric	ML 100K	ML 1M	R3-Yahoo	LibraryThing
	nDCG	0.4936	0.4242	0.0706	0.2206
U-U	Gini	0.2470	0.1352	0.3006	0.0390
	MSI	175.94	172.14	303.87	331.05
	nDCG	0.4953*	0.4296*	0.0717*	0.2385*
U-PJMS	Gini	0.2637	0.1637	0.4769	0.0319
	MSI	180.45*	182.75*	339.65*	417.57*

Table 5.2: Comparison of RM2 method using uniform user and item priors (U-U) or a uniform user prior and a probabilistic item prior estimate with Jelinek-Mercer smoothing (U-PJMS) on MovieLens 100k and 1M, R3-Yahoo and LibraryThing. Statistically significant improvements in nDCG@100 and MSI@100 according to permutation test ($p < 0.05$) are indicated with a star.

sine similarity to compute the user neighborhoods and additive smoothing to smooth the maximum likelihood estimates. We run the experiments on the MovieLens 100k and 1M, the R3-Yahoo and the LibraryThing datasets and we optimize for nDCG@100.

Our experiments show that the uniform estimator gives the best estimate of the user prior probability. Thus, giving more importance to some neighbors and demoting others in the recommendation formula does not improve the effectiveness of the recommendations. This result indicates that the probabilistic modeling of RM2 is able to tackle the importance of the feedback of each user with the conditional probability estimates.

Regarding the item prior, we found that the probabilistic item prior estimate using Jelinek-Mercer smoothing provides the best values of nDCG@100 on all the datasets. We plot in Figure 5.5, the values of nDCG@100, Gini@100 and MSI@100 when varying the parameter λ of the probabilistic item prior. We observe that the optimal λ in terms of nDCG@100 is around 0.5 and 0.7 which also provides good figures of diversity and novelty.

We compare in Table 5.2 the values of nDCG@100, Gini@100 and MSI@100 between using uniform estimators for the user and item priors or using a uniform user prior and a probabilistic item prior with Jelinek-Mercer smoothing. According to the permutation test ($p < 0.05$), the improvements in ranking accuracy and novelty are statistically significant on all datasets. We cannot perform paired statistical tests on Gini because it is not a pairwise metric, but a global one. Accuracy, diversity and

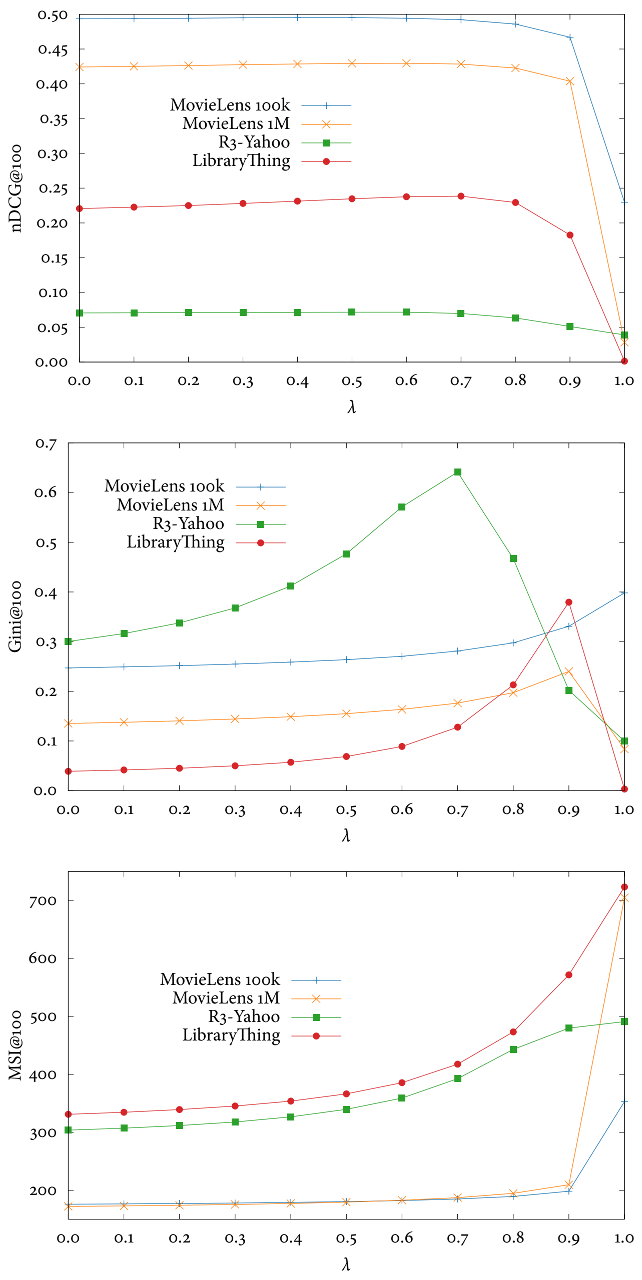


Figure 5.5: Values of nDCG@100 (top), Gini@100 (middle) and MSI@100 (bottom) for RM2 using a uniform user prior and a probabilistic item prior using Jelinek-Mercer smoothing on MovieLens 100k and 1M, R3-Yahoo and LibraryThing datasets. We vary the parameter λ of the item prior probability estimator.

novelty figures increase when using a probabilistic item prior except on the LibraryThing dataset where diversity is slightly reduced.

In the light of the results, we can expect small but significant improvements in ranking accuracy, diversity and novelty when using a probabilistic item prior using Jelinek-Mercer smoothing.

5.4 CONCLUSIONS

We studied the impact of smoothing methods and prior probability estimators in the ranking accuracy, diversity and novelty of relevance models for recommender systems. In particular, we established a connection between the IDF effect from information retrieval to the concept of novelty in recommendation. Then, we analyzed axiomatically how different smoothing methods affect the IDF effect on RM2. We found that collection-based methods penalize this effect while additive smoothing neither promotes nor demotes this property. Our experiments confirmed that additive smoothing provides better results than collection-based smoothing methods improving accuracy, diversity and novelty figures.

We also explored different user and item prior probability estimates. We identified the uniform estimator as the optimal one for modeling neighbors prior probabilities. Additionally, the probabilistic estimator based on Jelinek-Mercer smoothing is an excelent choice for computing the item prior probability. We found that the use of this prior estimator can significantly improve the ranking accuracy, diversity and novelty in most scenarios.

In summary, in this chapter, we studied thoroughly the effectiveness of relevance models to address the top-N recommendation task and proposed improvements to the original proposal. In the next chapter, we introduce a different adaptation of relevance models to deal with a novel recommendation problem.

6 ITEM-BASED RELEVANCE MODELS FOR LONG TAIL LIQUIDATION

Typically, recommenders tend to promote certain products or services of a company that are kind of popular among their users. An important research concern is how to formulate recommender systems centered on those items that are not very popular, that is, around the long tail products. A special case of those items are the ones that are the result of overstocking by the vendor. The excess of inventory or overstock is a source of revenue loss.

In the previous chapter, we studied the adaptation of relevance models as a user-based collaborative filtering technique proposed by Parapar et al. (2013). This adaptation provides high effectiveness figures in the top-N recommendation task. We propose a different adaptation of relevance models to deal with a novel recommendation task. We claim that recommender systems can also be used to liquidate long tail products and maximize the business profit. Thus, we formalize this recommendation task and propose an evaluation protocol. Then, we present a method to tackle this task based on relevance models. Instead of creating user-based models as in the previous chapter, we build item relevance models that are able to tackle the long tail liquidation task effectively.

The formulation of the long tail liquidation task and the solution based on item relevance models presented in this chapter have been published (Valcarce et al. 2016d). In this chapter, we present the results of this article and improve our proposal by using additive smoothing instead of absolute discounting smoothing.

6.1 INTRODUCTION

Many e-commerce companies have started to use recommender systems with the intention of increasing the number of sales. From the business perspective, recommenders are effective tools to improve user satisfaction and, thus, sales revenue. There exist several aspects of RS that impact sales. For instance, Fleder and Hosanagar (2009) analyzed the effect of recommender systems in sales diversity thoroughly. These authors concluded that recommenders could increase sales if they discount popularity appropriately generating more diverse suggestions. The key idea is that recommender systems should not focus solely on popular products but also on long tail items. Anderson (2008) coined the term long tail to refer to those less popular products that have a low demand in large catalogs. In the current context of e-commerce, he declared that promoting long tail items is crucial for both the user and the business: customers may discover new and unexpected relevant products while the companies may sell the majority of their stock. He claimed that a retailing strategy based on selling a large number of products in small quantities is more profitable than a business centered on selling large amounts of a small set of popular products.

Traditionally, recommendation approaches in the long tail have explored ways of promoting long tail items in the recommendation process (Park and Tuzhilin 2008; Yin et al. 2012). However, addressing the long tail problem in that way is not particularly novel because a high-performance recommender should recommend both popular and long tail items according to what fits best to the users' tastes. A growing body of literature has focused on improving the diversity of recommender systems (Adomavicius and Kwon 2012; Vargas and Castells 2014). Since more diverse recommendations are expected to lead to larger catalog coverage, more long tail products are likely to be recommended. In the same way, many studies have analyzed the importance of novelty in RS. A recommendation is considered novel when the recommended item is unpopular—long tail items are, by definition, unpopular. Although enhancing diversity and novelty may increase the number of recommended long tail items, we argue that there will still be items that vendors will be unable to sell.

Very different factors can produce the long tail nature of a product: it is rarely sold and therefore has very few ratings, it is sold but it has almost no ratings given its delicate nature (e. g., sex toys) or the existing recommender barely recommends it. All these aspects make recommending in the long tail a challenging task. For this reason, effectively suggesting all the products of companies' catalogs is almost impossible following a unified approach.

In business terminology, the items from the inventory of a company that cannot be successfully sold are called excessive stock or *overstock*. Overstock can be the result of poor prediction of product demand, but it can also be a consequence of market fluctuations. Thus, an effective process management and a diverse recommender may minimize overstock effects. However, as we stated before, despite the efforts oriented towards improving sales of long tail products, there will be situations when companies will not be able to sell some items. In this case, we think that a recommender tailored to the specific task of getting rid of long tail products may provide interesting benefits. Therefore, in this paper, we develop a formal recommender model for dealing with these products which may help companies to liquidate their excessive stock.

First, we state the overstock clearance problem formally, we study how to evaluate this task and how this process differs from the classic recommendation problem. Next, we propose three methodologies for estimating which items are part of the excessive stock of businesses given a standard recommendation dataset. Then, we propose to use an item-based adaptation of relevance models to tackle the overstock liquidation problem. This approach builds a statistical relevance model of each of the long tail items which enables the identification of target users for selling those particular items. Finally, we conduct a series of thorough experiments to analyze and compare the performance of our proposal with other collaborative filtering algorithms. The results confirm our intuition that a probabilistic item-based relevance model enables to build an effective recommender to get rid of the long tail products.

6.2 LONG TAIL LIQUIDATION

We propose a novel recommendation task applicable to e-commerce.

The objective of a recommender system is to elaborate personalized rankings of products for each user. Every recommendation task involves a set of products or items (we will use these two terms interchangeably) and a set of possible customers or users. Recommendation in the long tail refers to the generation of item recommendations to users including not only popular products but also long tail ones. However, here we propose a different approach: for those items in the long tail we want to get rid of, we want to identify those potential users that will buy the product, even when that item is not on the top of the users' preferences. Of course, this approach does not replace the classic top-N recommendation task. On the contrary, it is especially designed to address a business concern and should be used as a complement to current methods in an operational setting. More specifically, we claim that recommenders may help to get rid of long tail products which can be seen as a proxy representation of the overstock phenomena.

Content-based algorithms are usually preferred for improving novelty because they find similar items based on content, not on popularity (Mooney and Roy 2000). However, sometimes the available information about items is not adequate for using this family of methods. Additionally, content-based recommenders may lead to over-specialization suggesting only items that are very similar to those rated by the target user (Ning et al. 2015). Therefore, we propose a collaborative filtering approach specially designed for the task of getting rid of the long tail.

6.2.1 *Problem formulation*

We propose a novel recommendation problem: instead of generating the best item suggestions for each user, we aim to find the best users for each long tail product. The inversion of the classic recommendation task (recommending users to items in the place of suggesting items to users) was recently studied to improve sales diversity. Vargas and Castells (2014) explored the inverted task and proposed a probabilistic approach that enhances sales diversity. Still, in the end, they intended to improve the original recommendation problem (suggesting items to users). In contrast, our intention is to address a very different problem: how to get rid of the excessive stock suggesting the most suitable users for each item.

The traditional top-N recommendation task can be expressed as finding a scoring function $s : \mathcal{U} \times \mathcal{I} \rightarrow \mathbb{R}$ such that, for each user u, we can generate a ranked list of n items, $L_u^n \in \mathcal{I}^n$, sorted by decreasing score order. In contrast, we define the long tail liquidation task as follows:

Definition (Long tail liquidation)**.** Let $\mathcal{I}' \subset \mathcal{I}$ be the subset of items that are the part of the catalog that forms the products we want to liquidate. The objective of the long tail liquidation task is to find a scoring function $s' : \mathcal{I}' \times \mathcal{U} \rightarrow \mathbb{R}$ such that, for each item $i \in \mathcal{I}'$, we can build to a ranked list of n users, $L_i^n \in \mathcal{U}^n$, that are most likely interested in such item i.

Note that in the long tail liquidation task the recommender may use the ratings emitted to all the items, but we are only interested in generating recommendations for the overstock products.

Sales on e-commerce sites depend on multiple factors. Seasonality is especially relevant because some products become outdated while new items appear. Additionally, other factors such as price fluctuations or changes in the users' needs may affect sales numbers. To avoid these issues which are out of the scope of this chapter, we propose to liquidate stock at the end of the season. Therefore, the recommendation algorithms should use the available data of the season to generate stock liquidation suggestions at the end of this period. The interests and needs of the users should not change extremely during the season and price should be managed according to the liquidation policy of the company. In this way, we can effectively apply a collaborative filtering approach without using outdated information from past seasons.

6.2.2 *Long tail estimation*

We propose three strategies to estimate overstock items from a rating a dataset.

To the best of our knowledge, there is no public dataset available for research that specifies excessive stock. Since we lack of datasets with such kind of information, we designed three approaches to estimate the subset of overstock items. We should note that, for this purpose, we use all the information in the datasets (without training and testing splits) because we are merely designating which items are excessive stock. The splits of the collections will be used to train and evaluate the recommenders. Next, we present three strategies to discriminate long tail products.

6.2.2.1 *Least Rated Products*

We can argue that the least rated items conform the set of overstock products we want to get rid off because few users have shown interest in them. This is the most common approach used in recent studies (Cremonesi et al. 2010; Yin et al. 2012). Thus, given the threshold c_1 we can select a subset of items that have less than c_1 ratings:

$$\mathcal{I}' = \left\{ i \in \mathcal{I} \;\middle|\; |\mathcal{U}_i| < c_1 \right\} \tag{6.1}$$

We use a fixed threshold instead of relying on a percentage (e. g., taking the 5% of least rated products) because in this way we can assure that the selected items are rarely sold (if we choose a proper threshold). In contrast, there will always be items in the bottom percentiles, but we cannot assure that those are long tail items, only that they are the least sold in the dataset.

6.2.2.2 *Lowest Rated Products*

Another approach to the estimation of the overstock products is to suppose that the items with the lowest ratings are the ones that cannot be sold by the company. We only need to choose a value for c_2 to specify the threshold of what is a low rating.

$$\mathcal{I}' = \left\{ i \in \mathcal{I} \;\middle|\; \frac{\sum_{u \in \mathcal{U}_i} r_{u,i}}{|\mathcal{U}_i|} < c_2 \right\} \tag{6.2}$$

Again, the same motivation as in the previous strategy favors the use of a fixed threshold instead of a percentage. Note that it is not advisable for a traditional recommender system to recommend low-quality items (probably those with low ratings) to their users. However, in this paper, we are tackling a very different problem: we want to get rid of long tail items, no matter the cost. Thus, we are devising a recommendation technique that is able to select which users are the potential buyers of a long tail item. We leave up to the manager to decide whether liquidating items with very low ratings is a good idea (perhaps offering a substantial discount). In this work, our objective is to present a recommendation algorithm that is capable of doing so if needed.

6.2.2.3 *Least Recommended Products*

Finally, according to Fleder and Hosanagar (2009) who have pointed out the effect of recommendations diversity in e-commerce sales, we can argue that those products that a standard recommender system does not suggest are not going to be sold. Thus, given a particular threshold c_3, we can construct a set of items that are not present in the top c_3 results of any user recommendation list:

$$\mathcal{I}' = \left\{ i \in \mathcal{I} \;\middle|\; i \notin L_u^{c_3},\ \forall u \in \mathcal{U} \right\} \tag{6.3}$$

6.3 ITEM LIQUIDATION USING RELEVANCE MODELS

Recently, Parapar et al. (2013) adapted relevance models to collaborative filtering achieving high accuracy figures. The key idea is to adapt the pseudo-relevance feedback paradigm to recommendation. This involves mapping a triadic space (queries, documents and terms) to a dyadic one (user and items). Queries and documents are both mapped to users while terms play the role of items. Thus, instead of expanding a query with new terms, user profiles are expanded with items from a pseudo-relevant set. We can compute this set by calculating the neighborhood of the target user.

In this section, we propose the construction of a relevance model for each long tail product. We intend to expand item profiles with relevant users using information from similar items. The objective is to estimate the probability of a user u under the relevance model of an item i, $p(u|R_i)$. Note that our approach to collaborative filtering recommendation using RM is not just the item-based version of the model proposed by Parapar et al. (2013). In that model, the authors built a relevance model for each user and estimated the probability of relevance of each item, $p(i|R_u)$. We cannot apply this model directly to the long tail liquidation task because probabilistic relevance estimates across users are not comparable. Given two users u and v and the long tail item i we cannot generate recommendations of users to liquidate that item sorting $p(i|R_u)$ and $p(i|R_v)$ since we are comparing estimates from different relevance models. Likewise, we cannot apply Bayes's Theorem to get the Bayesian inversion because the estimation of $p(R_i|u)$ o $p(R_j|u)$ does not make sense within this probabilistic framework. Therefore, we need to build a relevance model for each item if we want to model long tail items and choose the best users for them.

Lavrenko and Croft (2001) proposed two methods for approximating a relevance model: assuming independent and identically distributed sampling (RM1) or conditional sampling (RM2). In this paper, we focus on the latter because it has shown better results in our experiments. We refer to the adaptation of the Relevance Model 2 (RM2) model to the long tail liquidation problem as IRM2 (Item Relevance Modeling 2).

6.3.1 *IRM2 derivation*

IRM2 aims to build a relevance model R_i for each long tail product $i \in \mathcal{I}'$ of the collection. In this way, we can estimate the relevance of each user for a given long tail product. Thus, in our stock liquidation task, we can define an *item relevance model* as a formalism that enables to compute the probability that a user u rates a long tail item i, $p(u|R_i)$. We ignore which users will rate a particular item, but we can use the history of ratings of that item and other relevant items to estimate it.

We construct an item-based relevance probabilistic model.

We assume that the relevance model R_i generates the long tail item i but also a set of relevant items J_i. Figure 6.1 illustrates how the item i and the set of relevant items J_i are random samples from an unknown relevance model R_i. However, the sampling process for the target item i can be different from the sampling for the relevant products J_i. With these assumptions, we can estimate $p(u|R_i)$ for each user $u \in \mathcal{U}$.

First, we sample the item i from the underlying relevance model R_i. This process consists in repeatedly sampling k users $v_1 \dots v_k$ from R_i. These users correspond to the clients that bought the item: $\mathcal{U}_i = \{v_1 \dots v_k\}$. Since we sample k users for item i, $k = |\mathcal{U}_i|$. To estimate $p(u|R_i)$, which is unknown, we rely on what we observed before: the set of users $\mathcal{U}_i$. We need to answer what is the probability that the next

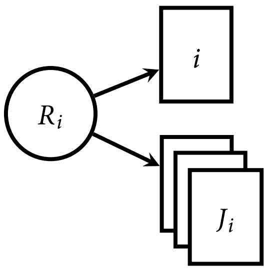


Figure 6.1: The item i and the set of relevant items to that item J_i are samples of the same model R_i (although the sampling process may vary).

user we sample from the relevance model will be u. Formally, we can formulate this question in the following way:

$$p(u|R_i) \approx p(u|v_1 \dots v_k) \tag{6.4}$$

Applying the definition of conditional probability, we can reformulate the previous equation in terms of the joint probability of observing the user u along with users $v_1 \dots v_k$ divided by the joint probability of observing users $v_1 \dots v_k$. Note that we can safely ignore the denominator because it remains constant for the same item i (it would not affect the final ranking):

$$p(u|R_i) \approx \frac{p(u, v_1 \dots v_k)}{p(v_1 \dots v_k)} \propto p(u, v_1 \dots v_k) \tag{6.5}$$

Following on the conditional sampling method proposed by Lavrenko and Croft (2001), we estimate the joint probability of observing the user u along with the users $v \in \mathcal{U}_i$ based on the ratings distribution of the relevant items $j \in J_i$. We present a diagram of this sampling scheme in Figure 6.2.

First, to estimate the relevance model of the item i, R_i, we pick a user u given the prior probability $p(u)$. This user u will condition the selection of item distributions from which we will sample the users $v_1 \dots v_k$:

$$p(u|R_i) \propto p(u, v_1 \dots v_k) = p(u) \prod_{v \in \mathcal{U}_i} p(v|u) \tag{6.6}$$

To estimate the conditional probability $p(v|u)$, we repeat the following process k times: we pick an item distribution j according to $p(j|u)$ and, then, we sample a user $v \in \mathcal{U}_i$ from the item distribution j with probability $p(v|j)$. Note that this sampling strategy considers that users $v_1 \dots v_k$

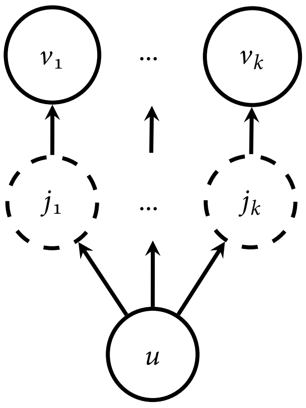


Figure 6.2: Conditional sampling used in IRM2.

are sampled independently of each other, but they are dependent on u. Applying the law of total probability we obtain the following:

$$p(v|u) = \sum_{j \in J_i} p(v|j, u)\, p(j|u) \tag{6.7}$$

If we assume that users $v \in \mathcal{U}_i$ become independent of u after choosing an item distribution $j \in J_i$, we can simplify Equation (6.7) obtaining the following estimate:

$$p(v|u) = \sum_{j \in J_i} p(v|j)\, p(j|u) \tag{6.8}$$

Applying Bayes' Theorem, we can estimate $p(j|u)$ as follows:

$$p(j|u) = \frac{p(u|j)\, p(j)}{p(u)} \tag{6.9}$$

Finally, plugging Equations (6.8) and (6.9) into Equation (6.6), we obtain the final IRM2 estimate:

$$p(u|R_i) \propto p(u) \prod_{v \in \mathcal{U}_i} \sum_{j \in J_i} p(v|j)\, \frac{p(u|j)\, p(j)}{p(u)} \tag{6.10}$$

To estimate the probability of user u under the relevance model R_i, IRM2 iterates over the users who rated that item, $\mathcal{U}_i$, and over the set of relevant items to that item, J_i. For the sake of simplicity, we consider prior probability estimates, $p(u)$ and $p(j)$, uniform. However, these priors open the door to explore new estimations that may improve the performance or even include business aspects.

With the estimate from Equation (6.10) we can generate the list of n recommendations L_i^n for each long tail item $i \in \mathcal{I}'$ by sorting users $u \in \mathcal{U}$ by decreasing estimated relevance $p(u|R_i)$.

Additionally, we need to provide two final estimation details. The first question is which items conforms the set of relevant items J_i. We explain this issue in Section 6.3.2. The other issue is how to calculate the probability of a user u given the item distribution j, $p(u|j)$. We use the maximum likelihood estimate of a multinomial distribution over the ratings:

$$p_{ml}(u|j) = \frac{r_{u,j}}{\sum_{v \in \mathcal{U}_j} r_{v,j}} \tag{6.11}$$

Since this estimate suffers from data sparsity, we need to use a smoothing method. We discuss smoothing strategies in Section 6.3.3.

6.3.2 *Computing the set of pseudo-relevant items*

In Equation (6.10), we can observe that IRM2 leverages the information from a set of relevant items to estimate the users' relevance. Since we ignore which items are relevant to some item i and we do not have explicit relevance feedback information, we need to compute approximation using the available data. We assumed that the similar items to the item i are the *pseudo-relevant* items for the relevance model R_i. We use the term pseudo-relevant because we want to emphasize that these items are an approximation of the real relevant items.

We employ clustering techniques for computing the set of pseudo-relevant items based on the ratings. In our case, we used the k-NN algorithm with cosine similarity. The cosine similarity s between two items i and j is given by the following equation:

$$\operatorname{cosine}(i,j) = \frac{\sum_{u \in \mathcal{U}_i \cap \mathcal{U}_j} r(u,i)\, r(u,j)}{\sqrt{\sum_{u \in \mathcal{U}_i} r(u,i)^2}\sqrt{\sum_{u \in \mathcal{U}_j} r(u,j)^2}} \tag{6.12}$$

6.3.3 *Smoothing method*

To avoid zeros in Equation (6.11), we smooth the maximum likelihood estimate with a background model which, in our case, is the user probability in the collection:

$$p(u|\mathcal{C}) = \frac{\sum_{j \in \mathcal{I}} r_{u,j}}{\sum_{v \in \mathcal{U},\, j \in \mathcal{I}} r_{v,j}} \tag{6.13}$$

The impact of smoothing methods for RM has been studied before in the context of the top-N recommendation task Valcarce et al. 2015c, 2016a. Among collection-based methods, absolute discounting smoothing tends to work best in top-N recommendation (Valcarce et al. 2015c); however, additive smoothing (a collection-agnostic method) provides better results because of its theoretical properties (Valcarce et al. 2016a). In previous work, we used absolute discounting smoothing to deal with the long tail liquidation task (Valcarce et al. 2016d). In contrast, in this chapter, we use additive smoothing which provides better effectiveness figures in the long tail liquidation task.

$$p_\gamma(u|i) = \frac{r(u,i) + \gamma}{\sum_{v \in \mathcal{U}_i} r(v,i) + \gamma\,|\mathcal{U}|} \tag{6.14}$$

We use the above estimator to calculate $p(u|j)$ and $p(v|j)$ in the computation of the item relevance model in Equation (6.10).

6.4 EXPERIMENTS

We conducted a series of experiments to analyze the performance of our proposal, IRM2, for the novel task of liquidating long tail items. In this section, we describe the set-up of the experiments and present and discuss the results.

6.4.1 *Datasets*

We conducted our experiments on three different collections. On the one hand, we used the MovieLens 1M and the LibraryThing datasets with explicit feedback in the form of ratings. On the other hand, we employed the Ta-Feng dataset with purchase information over a four-month period (from November 2000 to February 2001 inclusive).

Each collection was divided into training and test subsets. For the MovieLens and LibraryThing datasets, we included for each item 80% of its ratings in the training subset. This partition scheme ensures that each item has ratings both in the training and the test sets enabling to evaluate the performance of the recommendation algorithms on all the long tail items. For the Ta-Feng dataset, we used the data from the three first months to train the recommendation algorithms and the last month as the test set.

6.4.2 *Baselines*

We adapt top-N recommenders to the long tail item liquidation task as baselines.

To assess the performance of our proposed algorithm, we must compare it with a set of representative baseline recommender systems. To the best of our knowledge, no specific algorithm exists for the novel task we are proposing. Therefore, we use standard collaborative filtering algorithms from the state of the art. Since these methods are designed for computing item recommendations for users, we adapt them to the task of long tail liquidation. We describe these adaptations together with the method. Recommending long tail items is difficult, and the collections we built are estimations of the excessive stock. Therefore, facing this novel task, it is important to use a great variety of algorithms which employ different strategies in order to make a complete comparison between our proposal and very diverse approaches. We describe the baselines below.

6.4.2.1 *Random*

This strategy is not a recommender, but a basic baseline since it should yield the worst performance. This algorithm simply recommends random users to long tail items.

6.4.2.2 *Popularity*

Again, this is not a proper recommender algorithm, but a naïve approach to compare more sophisticated methods. This strategy chooses the most popular users for all the items. In other words, for each item, we recommend the same set of users: the ones who have more ratings in the training set.

6.4.2.3 *User-based and item-based kNN*

A classic collaborative filtering technique consists in computing a set of k nearest neighbors for each user or item (Ning et al. 2015). These neighborhood relationships are computed using pairwise similarities. We used Pearson's correlation coefficient for this purpose because it is the most common similarity for this method Ning et al. 2015. Once we have calculated the neighborhood, the recommender aims to predict the rating that the target user would emit based on the ratings of the neighbors. User-based (kNN-UB) and item-based (kNN-IB) versions are presented in Equations (6.15) and (6.16), respectively.

$$\hat{r}(u,i) = \frac{\sum_{v \in V_u} \rho(u,v)\, r(v,i)}{\sum_{v \in V_u} |\rho(u,v)|} \tag{6.15}$$

$$\hat{r}(u,i) = \frac{\sum_{j \in J_i} \rho(i,j)\, r(u,j)}{\sum_{j \in J_i} |\rho(i,j)|} \tag{6.16}$$

where $\rho(i,j)$ is the Pearson's correlation coefficient between items i and j and J_i indicates the neighborhood of item i. Likewise, $\rho(u,v)$ is the Pearson's similarity between users u and v and V_u represents the neighborhood of user u.

To recommending users to long tail items, we generate a recommendation list L_i for each $i \in \mathcal{I}'$. This list contains those users $u \in \mathcal{U}$ with the largest predicted rating, $\hat{r}(u,i)$.

6.4.2.4 *User item relevance (UIR)*

UIR is a probabilistic recommendation technique (Wang et al. 2006). Although it was proposed for implicit feedback datasets, their use with explicit feedback is straightforward. We used the item-based version (UIR-IB) because it outperformed the user-based counterpart in all our experiments. Despite being an item-based approach, UIR-IB still computes an estimate of the relevance of an item given a user model. The formula for estimating the score of the item i for the user u under the UIR-IB model is given by:

$$\text{score}(u,i) \propto \sum_{\substack{j \in \mathcal{I}_u \\ \mathcal{U}_i \cap \mathcal{U}_j \neq \varnothing}} \log\left(1 + \frac{(1-\lambda)\, p_{ml}(j|i,r)}{\lambda\, p(j|r)}\right) + \log p(i|r) \tag{6.17}$$

where the sum is over the items rated by the target user that were rated by other users who also rated the target item i. The probability of an item i assuming relevance is computed as the number of users that rated that item:

$$p(i|r) \propto |\mathcal{U}_i| \tag{6.18}$$

and the maximum likelihood estimate of an item j given the target item i and assuming relevance is proportional to the number of users that rated both items:

$$p(j|i,r) \propto \frac{|\mathcal{U}_i \cap \mathcal{U}_j|}{|\mathcal{U}_i|} \tag{6.19}$$

The list of recommendations for each long tail item $i \in \mathcal{I}'$, L_i, is generated sorting all the users $u \in \mathcal{U}$ by decreasing value of score(u,i).

6.4.2.5 *Hitting time (HT)*

Yin et al. (2012) designed this recommender system to deal with the long tail. The authors argued that most collaborative filtering algorithms are not able to recommend long tail items due to data sparsity. The authors overcame this problem modeling the recommendation task as a random walk in a graph. HT builds an edge-weighted undirected graph where the nodes are items and users. Each rating is a weight connecting two nodes (the corresponding user and item). Given that graph, the authors compute the hitting time from the item i to the target user q, $H(q|i)$, which is the average number of steps that a random walker starting from the node i will take to reach the node q.

Given the target user q, hitting time is initialized for each node x with $HT_0(q|x) = 0$. Then, for each user node u and each item node i, the following is computed iteratively (τ iterations):

$$HT_{t+1}(q|i) = 1 + \sum_{u \in \mathcal{U}_i} HT_t(q|u)\, p_{i,u} \tag{6.20}$$

$$HT_{t+1}(q|u) = 1 + \sum_{i \in \mathcal{I}_u} HT_t(q|i)\, p(u,i) \tag{6.21}$$

where:

$$p(u,i) = \frac{r(u,i)}{\sum_{j \in \mathcal{I}_u} r(u,j)} \tag{6.22}$$

$$p_{i,u} = \frac{r(u,i)}{\sum_{v \in \mathcal{U}_i} r(v,i)} \tag{6.23}$$

Those items whose hitting time with respect to q is the smallest are the candidates for recommendation for the target user q. In the long tail liquidation task, we build the list of recommendations with those users with the smallest hitting times to the target long tail item.

6.4.2.6 *PureSVD*

Matrix factorization techniques are a very fertile area of research. Multiple approaches to collaborative filtering based on low-rank approximations have been developed (Koren and Bell 2015). Here, we chose PureSVD as a representative method of this family because it a simple technique specially oriented to the top-N recommendation problem which has achieved high values of accuracy (Cremonesi et al. 2010). Also, it has demonstrated to be capable of recommending long tail items effectively (Cremonesi et al. 2010). PureSVD computes the standard singular value decomposition of the ratings matrix. We estimate the score for a user u and an item i as the dot product between the u-th user latent vector and the i-th item latent vector. With these scores, we can build recommendation lists for the long tail liquidation task in the same way as we have done for the neighborhood methods. Section 12.1.2.1 describes this technique in more detail.

6.4.2.7 *Sparse linear methods (SLIM)*

Finally, we selected a method that learns an item-item similarity matrix to generate recommendations. More specifically, SLIM computes a sparse aggregation coefficient matrix (Ning and Karypis 2011). The score of the item i for the user u is estimated as the dot product between the user vector of ratings and the i-th column vector of the item-item similarity matrix. For the long tail liquidation task, the predicted score $\hat{r}(u, i)$ can be used for generating recommendation lists in the same way as we have done for the neighborhood and PureSVD methods. Section 12.1.1.1 contains the details of this recommendation method.

6.4.3 *Evaluation protocol*

Since in this paper we are dealing with a novel recommendation task, we need to discuss the evaluation protocol. In this task, we focus on ranking-oriented metrics for assessing the performance of recommenders because our primary concern is to liquidate excess inventories. We consider that a recommendation is accurate if and only if it leads to a purchase. The evaluation in this scenario demands data about sales and stocking information that unfortunately is not always available. Therefore, on the ratings-based datasets, we derive purchase information assuming that each rating represents an acquisition: the user u bought the item i if there exists a rating $r(u, i)$.

We propose the TestUsers protocol stemming from the TestItems approach (Bellogín et al. 2011). TestUsers consists in recommending, for each long tail item, all the users who have rated some item in the test set but have not rated the target long tail item.

In the long tail liquidation task, we are concerned about precision-oriented metrics. Therefore, in the next experiments, we report the values

Dataset	Least rated	Lowest rated	Least recommended
MovieLens 1M	329 (8.88%)	502 (13.55%)	1003 (27.06%)
LibraryThing	6865 (18.44%)	165 (0.44%)	16496 (44.31%)

Table 6.1: Number and ratio of items identified as long tail products on the MovieLens 1M and LibraryThing datasets with the three proposed strategies.

of nDCG@100. Since the trends are the same, for the sake of brevity we do not report the values of precision at 100.

6.4.4 *Experiment with ratings*

We first estimate overstock items from ratings-based datasets.

We evaluate IRM2 against the baselines on the MovieLens 1M and LibraryThing datasets. We choose the long tail items following the three approaches described in Section 6.2.2. In particular, we take those items with less than 6 ratings (least rated strategy), those with less than a 2.5 average rating in a scale from 1 to 5 (lowest rated strategy) and those who do not appear in the top 50 list of the user-based neighborhood recommender using 100 nearest neighbors according to Pearson's correlation coefficient (least recommended strategy). Table 6.1 shows the number of items selected in each approach. We consider that this selection of items is reasonable as well as diverse which may be useful to assess the recommendation approaches in the stock liquidation task.

We optimize the parameters of all the recommendation algorithms with respect to nDCG@100. For the neighborhoods approaches (kNN-UB, kNN-IB and IRM2), we tuned k from 50 to 500 in steps of 50. The parameter λ in UIR-IB was tuned from 0.0 to 1.0 in steps of 0.1. The number of iterations τ in the HT algorithm was tuned from 5 to 35 in steps of 10. The number of dimensions d in PureSVD was tuned from 50 to 500 in steps of 50. The parameters α and β in SLIM were tuned from 0 to 5 in steps of 0.25. Finally, for IRM2, we selected the best smoothing parameter γ from $\{0.001, 0.01, 0.1, 1\}$.

Tables 6.2 and 6.4 show the nDCG@100 values following the three long tail strategies on the MovieLens and LibraryThing datasets, respectively. We use the permutation test to determine whether the improvements in terms of nDCG@100 were statistically significant at a significance level of 0.05. Additionally, for the sake of reproducibility, Tables 6.3 and 6.5 gather the best values of the parameters of the different recommendation approaches.

Method	Least rated	Lowest rated	Least recommended
Random	0.0025	0.0065	0.0045
Pop	0.1335^{acd}	0.1216^{acd}	0.1331^{acd}
*k*NN-UB	0.0000	0.0072	0.0067
*k*NN-IB	0.0000	0.0134^{ac}	0.0083^{a}
UIR-IB	0.1418^{acd}	0.1234^{acd}	0.1348^{abcd}
HT	0.1473^{acd}	0.1406^{abcde}	0.1479^{abcde}
PureSVD	0.1043^{acd}	0.2573^{abcdef}	0.2070^{abcdef}
SLIM	0.1948^{acdg}	**$0.3231^{abcdefg}$**	$0.2468^{abcdefg}$
IRM2	**$0.2355^{abcdefg}$**	$0.3151^{abcdefg}$	**$0.2524^{abcdefg}$**

Table 6.2: Values of nDCG@100 for each recommender approach on the MovieLens 1M dataset considering the long tail items those products with less than 6 ratings (least rated), those with an average rating less than 2.5 (lowest rated) and those that are not recommended in the top 50 (least recommended). Statistically significant improvements according to Wilcoxon signed-rank test ($p < 0.05$) with respect to Random, Popularity, *k*NN-UB, *k*NN-IB, UIR-IB, HT, PureSVD, SLIM and IRM2 are superscripted with a, b, c, d, e, f, g, h and i, respectively. Best value for each strategy is printed in bold.

Even though the results vary among strategies and datasets, we can find some general patterns. Although all the tested baselines are designed for dealing with the traditional recommendation task—suggesting items to users—the results show that some techniques are good approaches to the task of recommending users to long tail items. However, our proposed method IRM2 is, in general, the best option for the long tail liquidation task.

As it was expected, Random and Popularity methods behave poorly in this task. Nevertheless, the classic neighborhood methods (*k*NN-UB and *k*NN-IB) perform worse than these naïve strategies in some experiments. This result may indicate that traditional neighborhood algorithms are not suitable for the task we are proposing in this chapter. The reason lies in the fact that computing neighborhoods for long tail items is difficult because they have very few ratings. Pairwise similarities, such as Pearson's correlation coefficient, provide bad results when only a few co-occurrences between vectors are available. The item-based approach (*k*NN-IB) perform equal or better than the user-based counterpart (*k*NN-UB) in all the tested scenarios. Thus, not only finding neighborhoods for long tail items is challenging in this scenario; finding user neighborhoods who have information about long tail items is even more problematic. It has

Method	Least rated	Lowest rated	Least recommended
Random	–	–	–
Pop	–	–	–
*k*NN-UB	$k = –$	$k = 500$	$k = 450$
*k*NN-IB	$k = –$	$k = 50$	$k = 50$
UIR-IB	$\lambda = 0.1$	$\lambda = 0.1$	$\lambda = 0.1$
HT	$\tau = 15$	$\tau = 15$	$\tau = 15$
PureSVD	$l = 50$	$l = 150$	$l = 100$
SLIM	$\beta = 5, \alpha = 0.25$	$\beta = 5, \alpha = 1$	$\beta = 5, \alpha = 0.25$
IRM2	$k = 150, \gamma = 10^{-3}$	$k = 50, \gamma = 10^{-3}$	$k = 150, \gamma = 10^{-3}$

Table 6.3: Hyperparameters of each recommender system reported in Table 6.2 on MovieLens 1M.

Method	Least rated	Lowest rated	Least recommended
Random	0.0024	0.0002	0.0030
Pop	0.0408^{acd}	0.0499^{acd}	0.0455^{acd}
*k*NN-UB	0.0018	0.0039	0.0026
*k*NN-IB	0.0255^{ac}	0.0061	0.0169^{ac}
UIR-IB	0.0890^{abcd}	0.0894^{abcd}	0.0876^{abcd}
HT	0.1431^{abcdeg}	0.1451^{abcdeg}	0.1477^{abcdeg}
PureSVD	0.0879^{abcd}	0.0919^{abcd}	0.1065^{abcde}
SLIM	$0.2004^{abcdefg}$	$0.2029^{abcdefg}$	$0.2495^{abcdefg}$
IRM2	$\mathbf{0.2120}^{abcdefgh}$	$\mathbf{0.2108}^{abcdefg}$	$\mathbf{0.2522}^{abcdefg}$

Table 6.4: Values of nDCG@100 for each recommender approach on the LibraryThing dataset with the same notation used in Table 6.2.

been acknowledged that for the traditional recommendation task, item-based approaches tend to achieve better accuracy figures (Cremonesi et al. 2010; Ning et al. 2015). Under this new paradigm, where we want to recommend users to items, we observed that the item-based approaches are also desirable.

UIR-IB performance is acceptable compared to the aforementioned baselines. This may happen because UIR is based on the probability ranking principle, as RM and our method IRM2. Additionally, UIR also uses smoothing to deal with data sparsity which may explain the good results in the tested scenarios. However, their performance is much lower than IRM2. This result was expected because UIR ignores the value of the

Method	Least rated	Lowest rated	Least recommended
Random	–	–	–
Pop	–	–	–
*k*NN-UB	$k = 50$	$k = 350$	$k = 400$
*k*NN-IB	$k = 450$	$k = 100$	$k = 50$
UIR-IB	$\lambda = 0.1$	$\lambda = 0.1$	$\lambda = 0.1$
HT	$\tau = 15$	$\tau = 15$	$\tau = 15$
PureSVD	$l = 500$	$l = 300$	$l = 500$
SLIM	$\beta = 2.5, \alpha = 1$	$\beta = 2.5, \alpha = 0.5$	$\beta = 2.5, \alpha = 1$
IRM2	$k = 150, \gamma = 10^{-3}$	$k = 150, \gamma = 0.1$	$k = 150, \gamma = 10^{-3}$

Table 6.5: Hyperparameters of each recommender system reported in Table 6.4 on LibraryThing.

ratings since it was originally designed for dealing with implicit feedback modeling the co-occurrences of ratings. We also tested the user-based version of UIR (Wang et al. 2006), but the results were unsatisfactory, as with *k*NN-UB.

Hitting time shows good values of nDCG@100 in the experiments. We think that this is motivated by the fact that HT computes the average number of steps that a random walker needs to go from one node to other in the graph (Yin et al. 2012). This algorithm can generate recommendations for both users and items because all of them are nodes of the same graph connected by ratings—this model does not establish any difference between users and items. The symmetry of this algorithm between both entities is crucial in the quality of the recommendations for this novel task.

Even though HT outperforms PureSVD in four out of the six scenarios, the MF method shows high figures of nDCG@100. This approach computes a full singular value decomposition of the user-item matrix (Cremonesi et al. 2010). If we transpose this matrix and calculate a new decomposition, we will obtain the same user and item latent factors, but in switched places. Thus, this technique is also symmetric with respect to users and items. We consider that this property is responsible for the good results of this method for the stock liquidation task.

The strongest baseline is SLIM. In fact, in one scenario (MovieLens 1M considering long tail items those with a low average rating), SLIM outperformed IRM2. This method produces scores for user and items. Thus, the creation of a recommendation list in this task is done by sorting users by decreasing score according to one item. We think that SLIM produces very good recommendations since the method needs no adaptation to the stock liquidation task. Additionally, we can consider SLIM as an item-based recommender because, in the end, it is based on computing an item-item similarity matrix (Ning and Karypis 2011). Again, item-based approaches demonstrate superior performance compared to user-based ones. Although each column of the item-item similarity matrix can be computed independently, the learning process uses the global information of the user-item rating matrix. In contrast, IRM2 employs local information relying solely on the ratings of the item neighborhoods. This difference becomes crucial when scalability is a necessity.

Finally, IRM2 shows the best results in all the experiments except for the lowest rated scenario on the MovieLens dataset. Still, in this case, it was the second best method for the long tail liquidation task. We observed that the smoothing parameter γ is very stable within collections while the number of neighbors k is more dependent on the recommendation situation. However, the perfect optimization of these parameters is not crucial because the differences in terms of nDCG@100 are very slight with nearly-optimal parameters.

The notable figures of IRM2 can be explained studying the roots of the algorithm. For each long tail item, this method computes an item relevance model based on the item neighbors. Then, we can estimate the relevance of each user for a given item model. Within this Bayesian framework, the relevance of an item given a user is different of the relevance of a user given an item. The best baselines were those that are item-based methods or symmetric between users and items (i. e., they produce the same result if we switch users and items). IRM2 yields better results than these methods because it copes with the new task by building a relevance model in the item space.

6.4.5 *Experiment with purchases*

We now use a dataset of purchases.

In the previous experiment, we treated user ratings as purchases. This may raise doubts about the applicability of our approach. Thus, we also tested IRM2 and the baselines on a sparser dataset that contains purchase information: the Ta-Feng dataset. We used the data from the three first months to train the recommendation algorithms and the last month as the test set. This temporal split also helps to model a more realistic scenario: we use the information from a short period to liquidate the least sold products at the end of this period.

Since we have sales information, we designed our experiment with the objective of liquidating those products with a small number of sales. We

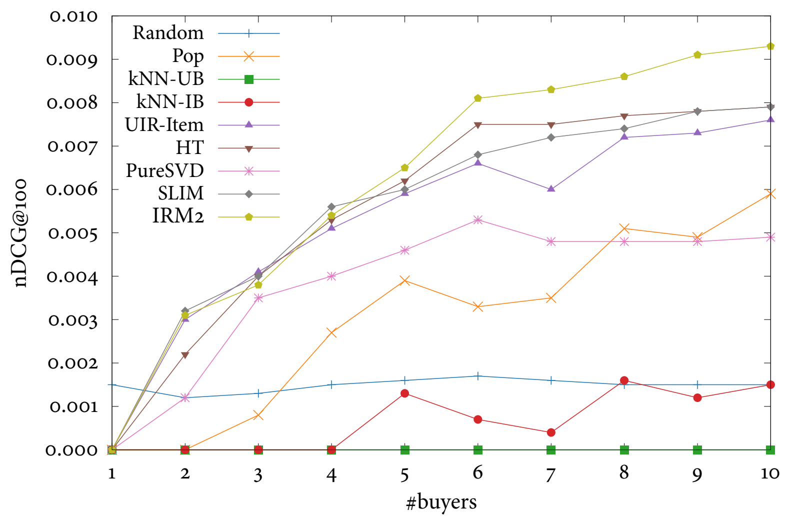


Figure 6.3: Values of nDCG@100 on the Ta-Feng dataset for liquidating long tail items using Random, Popularity, *k*NN-UB, *k*NN-IB, UIR-IB, HT, PureSVD, SLIM and IRM2 algorithms. Long tail items are those with no more than n buyers with $n \in [1, 10]$.

took those items with no more than n purchases where n ranged from 1 to 10. Instead of using user-item ratings, the recommendation algorithms used the number of purchases as training data.

Figure 6.3 shows the nDCG@100 values of IRM2 and the baselines on this dataset for liquidating those items that have no more than n buyers ($n \in [1, 10]$). To avoid overfitting, we used the optimal parameters for $n = 10$. We report their values in Table 6.6.

No algorithm provided valid recommendations for those items that were bought by only one customer except for the Random approach. The reason is that collaborative filtering techniques need training data of an item to be able to recommend it. In the case of products bought once, we either have a purchase in the training set or the test set. If the purchase is in the training set, there is no relevance judgment left to evaluate the recommendation. In contrast, if the purchase is in the test set, we have no training data to generate recommendations. Thus, only the Random technique was able to generate recommendations in this scenario. The Popularity approach might also have worked because it is not a collaborative filtering technique; however, in this very sparse dataset popularity was not a useful approach.

Disregarding the singular point of only one buyer, IRM2 outperforms the rest of baselines consistently. The relative performance of some recom-

METHOD	TA-FENG
Random	–
Pop	–
kNN-UB	k = –
kNN-IB	$k = 50$
UIR-IB	$\lambda = 0.1$
HT	$\tau = 15$
PureSVD	$l = 300$
SLIM	$\beta = 2.5, \alpha = 0.25$
IRM2	$k = 300, \gamma = 0.1$

Table 6.6: Hyperparameters of each recommender system reported in Figure 6.3 on the Ta-Feng dataset.

menders varies when we change the threshold of the number of buyers. For instance, SLIM and HT are the next best algorithms. However, IRM2 can deal effectively with either a high sparsity scenario or a more uncomplicated one. This is even more remarkable given the fact that in this scenario we do not have ratings. Therefore, our proposal is not in optimal conditions because IRM2 exploits this graded information meanwhile strong baselines such as SLIM and UIR-IB ignore that.

6.5 RELATED WORK

While previous research has studied the effect of recommender systems in sales, to the best of our knowledge, there is no previous work on the explicit task of recommending users to long tail items. For addressing the long tail phenomenon, some research efforts have focused on developing effective recommendation algorithms that are capable of suggesting long tail items to users (Cremonesi et al. 2010; Yin et al. 2012). However, there is no previous work on how to deal with those items that recommender systems are unable to sell. These long tail items are key to increase revenue (Anderson 2008; Yin et al. 2012). In this chapter, we have formulated the problem of stock liquidation within a recommendation framework with the aim of maximizing income. A considerable amount of literature has focused on improving revenue from different perspectives. We describe some of the most representative works in this area below.

Several authors have insisted on the importance of sales diversity for maximizing profit (Anderson 2008; Fleder and Hosanagar 2009; Vargas and Castells 2014) and this is currently a very active topic of research within the field of recommender systems. Fleder and Hosanagar (2009) simulated the effects of recommenders and concluded that classic systems reduce sales diversity reinforcing the "blockbuster nature" of media (i.e., promoting popular products) . They also studied whether personalization may create segregation and they find out that recommenders can help users widen their interests and create commonality (Hosanagar et al. 2014). Although users can discover new products thanks to recommenders, they tend to find the same popular items. This fact is especially present in collaborative filtering approaches where the system cannot recommend items for which it has no information (Mooney and Roy 2000). These algorithms find popular items in the user neighborhood (or popular users across similar items). Thus, niche products or new items are hard to recommend.

In the last years, several authors have focused on improving sales diversity as well as novelty in recommender systems. The probabilistic framework proposed by Vargas and Castells (2014) is especially relevant to our paper because they also examined the inversion of the recommendation task although with a different goal in mind. Other approaches reorder the output ranking of standard recommenders taking into account diversity at the expense of a reduced loss in accuracy (Adomavicius and Kwon 2012). However, for the proposed task, diversity in the recommendations will not help to increase the success rate in the suggestion of users to items. In this task, we are interested in searching for users that would buy surfeit products (i.e., producing high precision recommendations) because we are interested in selling those specific items.

Instead of boosting diversity and novelty in the recommenders trying to improve sales, Azaria et al. (2013) developed techniques for maximizing profit directly. These methods modify the original ranking of any recommender system improving the business revenue. Also, Azaria et al. (2013) discovered that the percentage of users that wanted to watch a film again is surprisingly high. This may lead to changing the way recommendation is performed because, in some scenarios, it would be acceptable to suggest products that users have already bought.

6.6 CONCLUSIONS

We proposed a new unstudied problem in the field of recommender systems: how to liquidate long tail items. Vendors usually have remaining products in their catalog they wish to get rid of. We described this task formally and designed an item-based adaptation of relevance models, IRM2, to cope with this problem. This model builds a relevance model for each long tail item. We also proposed three strategies to estimate the long tail items from a recommendation dataset based on the number of ratings, on the average value of the ratings or the recommendation frequency. Additionally, we used a dataset with purchase information. IRM2 outperformed a set of representative state-of-the-art recommendation algorithms in our experiments. We also found that traditional item-based approaches worked better than user-based ones.

This chapter presented a different adaptation of relevance models to recommendation. Parapar et al. (2013) showed how RM could be used as effective top-N recommenders, and we show here that they can also be adapted to tackle novel recommendation tasks such as long tail liquidation. In the next chapter, we use this item-based adaptation of relevance models to address another novel recommendation problem.

7 ITEM-BASED RELEVANCE MODELS FOR GROUP FORMATION

In the previous chapter, we presented IRM2, an item-based adaptation of relevance models for recommendation. We have seen how this model can address the long tail item liquidation task effectively using uniform prior probability estimates. In contrast, in this chapter, we exploit different user prior estimators to address a recent recommendation task: the user-item group formation problem.

Several daily activities are better enjoyed with a group of friends. Recommender systems can assist in finding the best companions for a given item. However, this task involves two objectives: we need to maximize the relations among the group as well as their interest in the proposed item. This task is called the item-driven group formation (UIGF) problem. Given a target user and a recommended item, we aim to find the best group of friends of the target user with whom to enjoy such item.

In this chapter, we present a collaborative filtering solution based on item-based relevance models. In this scenario, we design specific user prior probability estimators to tackle the UIGF problem effectively. Brilhante et al. (2016) presented the UIGF task and a graph-based approach to address it. Valcarce et al. (2018d) extended that work by proposing the adaptation of IRM2 to the UIGF task. In this chapter, we describe the user-item group formation problem and present the adaptation of IRM2 to this task.

7.1 INTRODUCTION

Recommender systems are a pervasive technology supporting several daily activities. In some domains, recommended items are better enjoyed with travel companions. Traditional recommender systems focus on identifying relevant items to single individuals. When the recommendation targets groups of users, it is referred to as *group recommendation*, whose goal consists in identifying items that a given group of users may like (Masthoff 2015). The group recommendation problem is hard to solve as users have diverse preferences and finding a trade-off among these preferences may bring to unsatisfactory or even unsettling recommendations for some of the users involved.

In this chapter, we address a complementary and even more challenging problem: given a user and a recommended item, we want to suggest the best group of friends with whom to enjoy the item. This task is known as the *user-item group formation* (UIGF) problem (Brilhante et al. 2016). In this task, we need to take into account both the social relationships and the preferences of the user and the group. This task is particularly relevant in the context of location-based social networks (LBSN). In these networks, users can establish bidirectional friendship relations as well as check-in in different venues and emit ratings. Venues can be places such as restaurants, cinemas or tourist attractions among others. Brilhante et al. (2016) formalized the UIGF problem and proposed a graph-based solution. More specifically, they reduced the UIGF problem to the problem of finding the densest k-subgraph in a graph obtained by enriching the user social network with item relevance information. In contrast, we propose a probabilistic collaborative filtering method based on relevance models. In particular, we use the IRM2 model with specific prior probability estimators to introduce the constraints of the UIGF problem.

7.2 USER-ITEM GROUP FORMATION

We address a new recommendation task: the user-item group formation problem.

The user-item group formation problem comprises a set of users $\mathcal{U}$ and a set of items $\mathcal{I}$. In this scenario, we may have user-item interactions in the form of ratings or check-ins. Additionally, we model the social network connecting users as a graph $\mathcal{S} = \{\mathcal{U}, E\}$ where $\mathcal{U}$ is the set of users and E is the set of undirected edges representing the friendship relationship between pairs of users in $\mathcal{U}$. We assume that each edge $e_{uv} \in E$ has a weight $w(u, v)$ indicating the *strength* of the friendship between u and v. Given the target user u, we call $\mathcal{S}_u = \{F_u, E_u\}$ the subgraph of $\mathcal{S}$ representing the social network of u. The nodes $F_u \subseteq \mathcal{U}$ constitute the set of friends of u and $E_u \subseteq E$ are the edges modeling the friendship relationships between these users.

UIGF is a recent recommendation problem that takes a user u and an item i as input and aims to find the best group of friends of the user

u for enjoying the item i. This task considers two different dimensions: friendship and item relevance in the group. On the one hand, the best group to enjoy an item should be formed by people that have strong ties among them. On the other hand, the item should be interesting for all the members of the proposed group individually. More formally, Brilhante et al. (2016) defined the UIGF problem as follows:

Definition (User-item group formation)**.** Given a user u, the social network $\mathcal{S}_u$ and an item i relevant to u, the UIGF problem seeks to find the group of k friends of u, $F_u^k \subseteq F_u$, that maximizes their "satisfaction", i. e., a measure that takes into account both the relevance of item i for all the members of the group and the intra-group friendship.

Brilhante et al. (2016) formulated the UIGF task as a densest k-subgraph problem over an enriched graph built from $\mathcal{S}_u$ and propose two algorithms to address it: one based on a greedy approach and another based on kNN algorithm. Next, we present our approach to the UIGF task using item-based relevance models (Valcarce et al. 2018d).

7.3 ITEM-BASED RELEVANCE MODELS FOR UIGF

We formulate the user-item group formation as an item relevance modeling task. We employ the IRM2 model presented in the previous chapter.

Since we assess the proposed solutions in the context of LBSN, venues play the role of items. Some of these social networks allow users to emit a rating for the venue; others only admit check-ins. When ratings are available, $r_{u,i}$ represents the rating that the user u gave to an item i. If not, we rely on the normalized count of check-ins to estimate $r_{u,i}$.

Relevance models are a state-of-the-art PRF technique (Lavrenko and Croft 2001). Even though these methods have originated in IR, Parapar et al. (2013) adapted the relevance modeling framework to the collaborative filtering scenario. Recently, we proposed an item-based relevance modeling framework for collaborative filtering to deal with a novel recommendation task: the liquidation of long tail items (Valcarce et al. 2016d). This task consists in identifying the most suitable users for offering them a given long tail product. Our proposal, IRM2, creates a relevance model for every long tail item and estimates the probability of relevance of each user under these models. This technique, which achieved excellent results in the task of liquidating long tail items, can also be used to tackle the UIGF problem.

We propose to use IRM2 to solve the UIGF problem because, both in the long tail liquidation and in the group formation tasks, we aim to recommend the most appropriate users for a target item. However, its use is not straightforward because both tasks possess their own peculiarities. In particular, when addressing the UIGF problem, we have to deal with all types of items, not only with long tail ones. This is not a difficulty since recommending for long tail items is, in principle, harder than recommending for regular ones. The main difference between the long tail liquidation task and the UIGF problem is that the latter exploits the friendship relationships among users whereas the former does not deal with this kind of information.

The recommendation for the target user u and the recommended item i can be addressed by estimating the probability of relevance of each friend $v \in F_u$ under the relevance model of the target item R_i using Equation (6.10). The recommended group consists of the k users with the highest estimated relevance. Formally, UIGF can be defined as follows:

Definition (UIGF as an Item Relevance Modeling problem)**.** Given the target user $u \in \mathcal{U}$, the recommended item $i \in \mathcal{I}$ and an integer k, the user-item group formation problem asks to find the set $F^G_{u,i} \subseteq \mathcal{U}$ where $\left|F^G_{u,i}\right| = k$ whose users $v \in F^G_{u,i}$ maximize the probability of relevance under the model of the recommended item i:

$$\begin{aligned} F^G_{u,i} = \underset{F^*_u}{\arg\max} & \quad \sum_{v \in F^*} p\left(v | R_i\right) \\ \text{s.t.} & \quad F^* \subseteq \mathcal{U},\ \left|F^*\right| = k \end{aligned} \tag{7.1}$$

We consider F_u as the candidate set of users which consists of only those users who are friends of the target user u. We build the set of most similar items, J_i, by taking the most similar items to i according to cosine similarity. We also use the additive smoothing to smooth the maximum likelihood estimate. To introduce the social relationships into IRM2, we extend the model by defining novel prior probability estimators that take into account the social information.

7.3.1 *Prior estimators*

We encode information about group dynamics in the priors.

One of the advantages of this relevance modeling framework is its sound statistical foundation which enables us to introduce different types of information in the probability estimates.

For the UIGF task, we use a uniform prior estimator for items while we explore different priors for users because we want to consider also the social graph generated by the friendship relationships to maximize the group satisfaction. With the proposed user priors, IRM2 is able to model a satisfaction function that takes into account both the strength of the relationship between users and the relevance of the target item for those users.

7.3.1.1 *Uniform prior (U)*

As a baseline, we studied the uniform estimator for the user prior. Since the set of candidate users of the group recommendation task is F_u (the friends of the target user u), the formulation of this prior is the following:

$$p_U(v) = \frac{1}{|F_u|} \tag{7.2}$$

7.3.1.2 *Common Friends (CF)*

This prior promotes users who share many common friends with the target user. Since the user prior is in the denominator of Equation (6.10), we formulate a prior which is inversely proportional to the number of common friends.

$$p_{CF}(v) \propto \frac{1}{|F_u \cap F_v|} \tag{7.3}$$

7.3.1.3 *Common group friends (CGF)*

This estimator boosts those users who have more common friends with the members of the current group $G_{u,i}$. Initially, the group is constituted by the target user, and this prior behaves as the CF prior. However, as more users are added to the candidate group, the estimate varies.

$$p_{CGF}(v) \propto \frac{1}{\left|\left(\bigcup_{w \in F^G_{u,i}} F_w\right) \cap F_v\right|} \tag{7.4}$$

7.3.1.4 *Group closeness (GC)*

This estimator pushes those users who have more friends in the current group $G_{u,i}$.

$$p_{GC}(v) \propto \frac{1}{\left|F^G_{u,i} \cap F_v\right|} \tag{7.5}$$

7.4 EXPERIMENTS

We use five publicly available datasets collected from four popular LBSN: FS, FS-NYC, Brightkite, Gowalla and Weeplaces. Table 3.3) shows detailed statistics about these collections. These datasets record information about the users registered in these social networks and the venues where the users checked-in. All datasets contain entertainment places such as restaurants, cinemas or tourist attractions among other venues. The social links between users are bidirectional friendship relationships. Foursquare-based datasets (FS and FS-NYC) contain not only check-ins but also user-item ratings.

All the datasets are extremely sparse in terms of check-ins and ratings. The rating/check-in density is below 0.01%. This poses a challenge for any recommender system. In particular, Foursquare collections have an especially low density of ratings and check-ins while the other datasets have very sparse social networks. These particularities affect the performance of our proposals as reported in Section 7.4.3.

Since we are dealing with a novel problem, we propose a new evaluation protocol based on ground truth groups. Next, we detail the baselines and the metrics used for evaluation. Finally, we describe and discuss the results of the experiments.

7.4.1 *Evaluation protocol and metric*

We used ground truth groups to evaluate UIGF approaches.

To assess the quality of the groups produced by UIGF models, we build ground truth groups (i. e., groups of friends that enjoyed a specific venue together). We extracted these ground truth groups from the five datasets. We looked for sets of users who checked in the same place within a fixed temporal window. We considered a user to be a member of a group only if this person is a friend of at least one of the other group members. In this way, we obtained groups of users who enjoyed the place where they checked-in, together with their friends.

As previous work (Brilhante et al. 2016; Valcarce et al. 2018d), we set the temporal window to 4 hours. Different values of the temporal window affect the number (and the size) of the ground truth groups mined. In our experiments we consider only groups with at least four members. The 4-hour window produces 1495 ground truth groups on FS, 258 on FS-NYC, 24 996 on Brightkite, 27 997 on Gowalla and 39 148 on Weeplaces. Weeplaces has the largest number of ground truth groups as it also has the largest number of check-ins.

Our evaluation protocol uses these ground truth groups in the following way: for each of these groups, we select a random member as the target user and the venue where the group registered as an item. Then, we ask the UIGF model to form a group of k friends for this specific user and venue. The members of the ground truth group are those whom we would like to find in the group suggested by the algorithmic solution.

We evaluate our proposals by exploiting these ground truth groups. We denote the ground truth group for user u and venue i by $\hat{F}_{u,i}$ and the recommended group $F^G_{u,i}$. We evaluate the quality of the recommendations using precision because we have binary relevance judgments. We averaged the metric over all the ground truth groups in each dataset. In this scenario, precision is computed as the fraction of members in $F^G_{u,i}$ that also appear in the ground truth group $\hat{F}_{u,i}$:

$$P\left(F^G_{u,i}\right) = \frac{\left|\hat{F}_{u,i} \cap F^G_{u,i}\right|}{\left|F^G_{u,i}\right|} \tag{7.6}$$

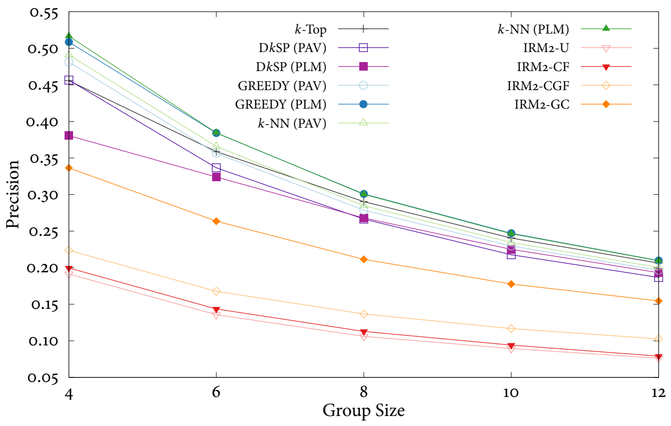
7.4.2 *Baselines*

We compare the performance of our solution with four graph-based approaches presented by Brilhante et al. (2016). The simplest baseline is k-Top which is a heuristic that computes a dense k-subgraph without considering the relationships among the users. We also use DkSP which is a well-known heuristic that aims at approximating the densest k-subgraph of a graph (Feige et al. 2001). We also employ Greedy and kNN, two approaches that exploit a user-item ego network (Brilhante et al. 2016). DkSP, Greedy and kNN approaches use different pairwise satisfaction metrics. We use pairwise aggregated voting (PAV) and pairwise least misery (PLM) as Brilhante et al. (2016) proposes.

7.4.3 *Results*

We show in Figures 7.1 and 7.2 the results in terms of precision of IRM2 with the different prior estimators and the baselines on the five datasets. We varied the group size of the solution from 4 to 12 people. We tuned the hyperparameters of all methods. For IRM2, we set the number of similar items to 400 and the additive smoothing parameter to 0.001.

Regarding the baselines, both Greedy and kNN outperform k-Top and DkSP in terms of precision for both PAV and PLM metrics. On average Greedy achieves better results on Foursquare datasets, while



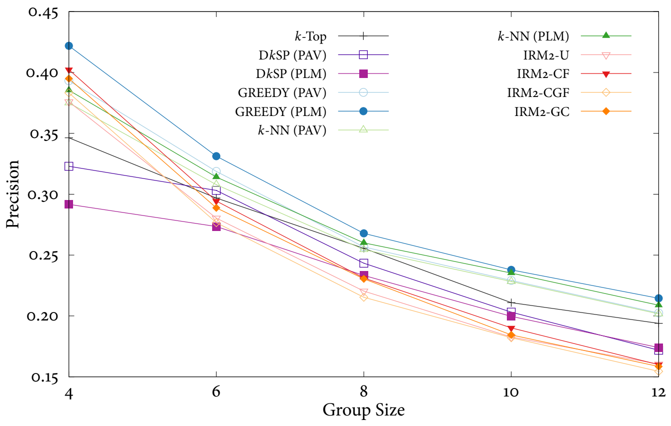


Figure 7.1: Values of precision of the different algorithms for the UIGF problem on FS (top) and FS-NYC (bottom) datasets.

*k*NN demonstrates a better performance on the Brightkite, Gowalla and Weeplaces datasets. It is worth highlighting that the improvement is higher for smaller values of k, while for larger groups the difference decreases. Moreover, Greedy and *k*NN are able to suggest more precise groups when using the PLM user-item relevance.

On the other hand, IRM2 outperforms all the algorithms on the Brightkite and Weeplaces datasets using any prior. The proposed priors demonstrate better performance than the original uniform prior. In particular, group closeness constitutes the best estimate and also outperforms all the algorithms on the Gowalla dataset. Also, it provides a significant improvement in performance on the Foursquare datasets. Nevertheless, on the Foursquare datasets, Greedy is a better option.

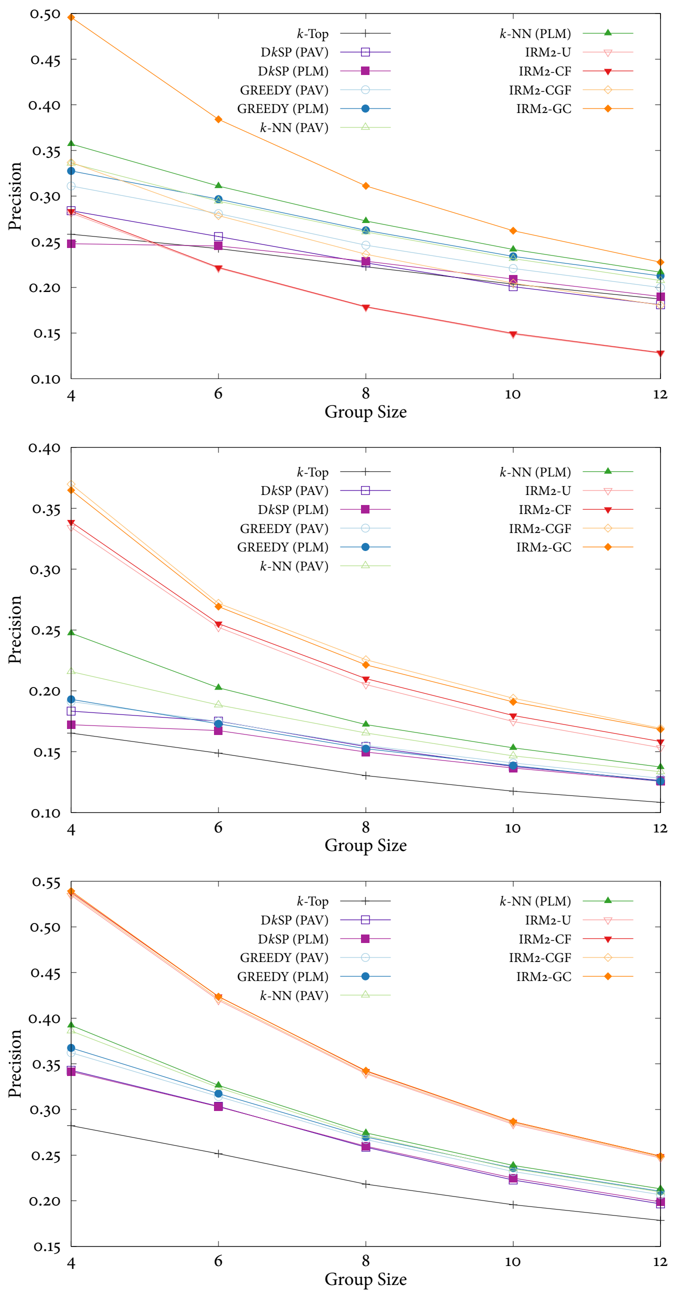


Figure 7.2: Values of precision of the different algorithms for the UIGF problem on Gowalla (top), Brightkite (middle) and Weeplaces (bottom) datasets.

By relating these results to the properties of the datasets, we can argue that IRM2 works better on datasets with sparser social networks but with denser check-in data while graph-based approaches such as Greedy and *k*NN perform very well on Foursquare datasets which present high sparsity on the ratings and check-ins but a higher number of links among users. A possible explanation of this phenomenon relies on the robustness of graph-based approaches in capturing group dynamics analyzing the user-item ego network. In contrast, the original formulation of IRM2 does not consider social relationships among users (Valcarce et al. 2016d). We introduced this information into the model by defining novel user prior estimators. Additionally, Greedy and *k*NN exploit the user-item relevance scores computed by a content-based technique meanwhile IRM2 is a collaborative filtering approach. This result is consistent with RS literature: content-based approaches tend to work better on sparser collections whereas collaborative filtering algorithms perform very well on less sparse datasets (Ricci et al. 2015).

7.5 RELATED WORK

Basu Roy et al. (2015) discussed the problem of group formation from a group recommendation perspective. They indeed consider a problem that is complementary to UIGF: how to build groups such that their members are mostly satisfied with the top-N provided recommendations. The problem consists in building non-overlapping groups of users by considering the similarity between their top-N recommended items. Different methods are proposed to measure group satisfaction. Although groups are built by considering items recommendations, this proposal ignores the social relationships between the users and do not restrict the size of the group which might lead to very large groups.

Some other relevant research topics are related to this work. In particular, group recommendation, team formation, community discovery and spatial social networks. Next, we summarize some results in these fields.

7.5.1 *Group recommendation*

This task consists in recommending a tailored list of items to a group of users considering the interests of each member of the group (Cantador and Castells 2012; Masthoff 2015). Cantador and Castells (2012) classified group recommendation techniques in model aggregation, prediction aggregation, group formation and cooperative consensus.

Hu et al. (2014) proposed a group recommender system that accommodates both individual choices and group decisions in a joint model through a model built with collective deep belief networks and dual-wing restricted Boltzmann machines. The authors claimed that traditional methods aggregating users preferences or predictions are very sensitive to noise in the data and that they might fail to learn group preferences when the data are slightly inconsistent due to strict aggregation assumptions.

Garcia et al. (2011) introduced a recommender system for tourism able to provide suggestions to groups. Authors designed a recommender system taking into account the tastes of the users, their demographic classification and the places they have visited on former trips. The group recommendation was built from individual recommendations through the application o aggregation and intersection mechanisms. While intersection considers the user preferences that are shared by all the members in the group, aggregation takes into account the union of preferences of users in the group, weighted by average user-interest.

Anagnostopoulos et al. (2017) studied the algorithmic implications of suggesting the best set of places that a group of people could perform together in the city. The authors addressed the problem by providing several formulations that take into account the overall group preferences as well as the individual satisfaction and the length of the tour recommended. They studied the computational complexity of these formulations and proposed solutions that were evaluated on datasets constructed from real city data.

In group recommendation, the group of users is assumed to be known in advance. This task deals with recommending a list of items to that group. In contrast, we address a different scenario where given a recommended item and a user, we have to compute the group that maximizes the intra-group social relationships and the relevance of the recommended item for each group member.

7.5.2 *Team formation*

The team formation problem asks to build a group offering an optimal match between its members and a set of functional requirements. Chen and Lin (2004) proposed a model to build multifunctional design teams in concurrent engineering. Their approach is based on representing the multifunctional knowledge of team members; their teamwork capability by taking their experience, communication skills and flexibility and their collegiality as it directly affects team performance. On the other hand, Lappas et al. (2009) formulated the team formation problem as follows. Given a social graph where nodes are labeled with a set of skills that each node possesses and given a task that requires a certain set of skills to be satisfied, the objective is to find a subgraph in which all skills are present and the communication cost is small.

Although both team formation and UIGF exploit a weighted social graph and the selection process requires group members to be socially close, the team formation problem deals with the finding of a set of experts that satisfies certain skills.

7.5.3 *Community discovery*

The community discovery problem aims at finding, at the global level, groups (communities) of users with greater ties internally than to the rest of the network. In contrast, our approach focuses on finding the group that maximizes the relevance of the recommended item for every member of the group and the intra-group social relationships, based on social network.

Sozio and Gionis (2010) studied a query-dependent variant of the community discovery problem, which they call the *community search* problem: given a graph and a set of query nodes in the graph, the authors proposed to find a subgraph that contains the query nodes and is densely connected. However, this problem does not consider information about items as it only relies on the network structure of the graph.

Coscia et al. (2011) classified community discovery methods based on different definitions of communities in the literature. Communities may involve several features like overlapping, weighted and directed links and social dynamics.

Communities have been exploited in the recommendation process. For example, Lee and Brusilovsky (2017) presented a recommendation technique that leverage community membership of the users as a useful information source for dealing with cold-start users, i. e., users for whom the system do not have enough personal information to provide useful recommendations. However, the authors focused on regular user-item recommendations and do not explore group recommendations.

7.5.4 *Spatial social networks*

Some approaches from the spatial social networks literature are also related to UIGF. Those approaches try to find groups of users with social relations among them that satisfy a given spatial constraint. In contrast, in this work, we model social networks with relevance information about items. Nevertheless, in some cases, we can argue that we can substitute the spatial distance with a metric based on item relevance to tackle a similar problem to UIGF.

Yang et al. (2012) proposed a socio-spatial group query to select a group of nearby people with tight social relations. They showed that the problem is NP-hard and designed an efficient algorithm to solve it. Although we can replace the spatial distance with a notion of item relevance, the approach is different from the one proposed here. First, their model allows specifying the average number of unfamiliar people an invitee may have. In our proposal, the notion of familiarity is explicitly enforced by a probability model that takes into account both the social relationship and item relevance for the group members. Second, Yang et al. (2012) aimed at minimizing the total spatial distance while we address the problem from a user-item relevance point of view.

Liu et al. (2012) proposed another similar socio-spatial approach to allow finding a group of people that are close to the target user in terms of physical distance and in terms of social distance. The authors showed that the problem is NP-Hard and propose an ϵ-approximation for that. However, in contrast to our proposal, Liu et al. (2012) aimed to minimize the maximum distance between every two vertices of the group. Moreover, they used as the distance the weighted average between the geographical distance and the closeness, in terms of social information while we maximize the density of the formed group. As they try to minimize a different function, this may lead to important differences in the resulting groups formed by the two approaches.

7.6 CONCLUSIONS

In this chapter, we proposed to use item-based relevance models to address the UIGF problem, i. e., finding the best group of companions with whom to enjoy an item. We introduced the concept of group satisfaction into IRM2 through different user prior probability estimators. Our experiments showed that our proposal provides better solutions than state-of-the-art techniques for the UIGF task in denser datasets. This shows the potential of probabilistic modeling in tackling unconventional recommendation tasks.

In the previous chapter, we have seen how item-based relevance models can address the long tail item liquidation task, and here, in this chapter, we have seen how these models can also tackle the item-driven group formation task by exploiting the potential of user priors. In the next chapter, we explore simpler pseudo-relevance feedback techniques based on Rocchio's algorithm to address the classic top-N recommendation problem.

8 THE ROCCHIO FRAMEWORK FOR USER-BASED RECOMMENDATION

We thoroughly studied the use of relevance models in different recommendations tasks in the previous chapters. Nevertheless, relevance models can be computationally expensive and difficult to adapt to large scale scenarios. In fact, in earlier work, we developed a distributed implementation of RM2 to alleviate efficiency issues (Valcarce et al. 2018a). Moreover, relevance models may use complex probability estimates with smoothing parameters that we need to tune.

In this chapter, we return to the top-N recommendation problem. We adapt term scoring functions used within the Rocchio framework, the standard query expansion method in the vector space model, to user-based recommendation.

The adaption of the term scoring functions presented in this chapter has been previously published (Valcarce et al. 2016b,e). However, instead of using Pearson's correlation for computing neighborhoods with *k*NN algorithm, we update our previous work by using cosine similarity which provides better results.

8.1 INTRODUCTION

An effective approach to collaborative filtering is the adaptation of relevance models, a pseudo-relevance feedback technique (Lavrenko and Croft 2001), to user-based recommendation (Parapar et al. 2013). The effectiveness of relevance models can be understood if we look at their sound statistical foundations since they are designed for generating a ranking of terms (or items in the top-N recommendation task) following the probability ranking principle.

Nevertheless, relevance models can be a computationally demanding probabilistic framework. Therefore, in this chapter, we aim to find techniques with a better trade-off between effectiveness and efficiency. Additionally, we seek parameter-free approaches that do not require hyperparameter tuning. In particular, we explore four term scoring functions used within the Rocchio framework: Rocchio weights (Rocchio 1971), Robertson selection value (Robertson 1990), Chi-square (Carpineto et al. 2001) and Kullback-Leibler divergence (Carpineto et al. 2001).

We follow the analogy between PRF and user-based collaborative filtering proposed by Parapar et al. (2013). Target users play the role of queries, and their ratings act as query terms. The neighborhoods of the target users play the role of pseudo-relevant sets. Therefore, similar users are used to extract items that are candidates to expand users profiles. These candidate items conform the recommendation list.

8.2 SCORING FUNCTIONS IN THE ROCCHIO FRAMEWORK

We adapt four information-theoretic term scoring functions used within the Rocchio framework for query expansion (Carpineto et al. 2001). This adaptation results in four simple and efficient neighborhood-based approaches.

8.2.1 *Neighborhoods*

Neighborhood-based approaches are collaborative filtering techniques that exploit the interactions from a set of similar users or items (Ning et al. 2015). In particular, user-based approaches rely on a set of similar users, called user neighborhood. The adaptation of term scoring functions to collaborative filtering requires the computation of these user neighborhoods.

We denote the neighborhood of the user u by V_u and its size by $|V_u|$. In previous work, we used the kNN algorithm with Pearson's correlation coefficient (Valcarce et al. 2016e). In this work, we use cosine similarity as it provides better results. In the following, we use $\mathcal{C}$ to denote the collection.

8.2.2 *Scoring functions*

Next, we present the adaptation of Rocchio weights (RW), Robertson selection value (RSV), Chi-square (CHI2) and Kullback-Leibler divergence (KLD) term scoring functions to user-based collaborative filtering. The complexity of these parameter-free methods is notably smaller than RM2.

These item ranking functions (except Rocchio Weights) use probability estimates such as $p(i|V_u)$ and $p(i|\mathcal{C})$. We compute these probabilities using the maximum likelihood estimate (MLE) of a multinomial distribution of ratings. We represent by $\mathcal{U}_X$ the set of users that rated the items from the set X. Likewise, $\mathcal{I}_X$ denotes the set of items that were rated by the users of the set X. In this way, the MLE is computed as follows:

$$p_{mle}(i|X) = \frac{\sum_{u \in \mathcal{U}_X} r(u,i)}{\sum_{u \in \mathcal{U}_X} \sum_{j \in \mathcal{I}_X} r(u,j)} \tag{8.1}$$

8.2.2.1 *Rocchio weights (RW)*

This method is based on the Rocchio's formula (Rocchio 1971). The score is computed as the sum of the weights for each term of the pseudo-relevant set. In recommender systems, this approach promotes highly rated items in the neighborhood.

$$p_{RW}(i|u) = \sum_{v \in V_u} \frac{r(v,i)}{|V_u|} \tag{8.2}$$

8.2.2.2 *Robertson selection value (RSV)*

The Robertson selection value technique computes a weighted sum of the item probabilities in the neighborhood (Robertson 1990).

$$p_{RSV}(i|u) = p(i|V_u) \sum_{v \in V_u} \frac{r(v,i)}{|V_u|} \tag{8.3}$$

8.2.2.3 *Chi-square (CHI2)*

This method roots in the Chi-square statistic (Carpineto et al. 2001). The probability in the neighborhood plays the role of the observed frequency, and the probability in the collection is the expected frequency.

$$p_{CHI2}(i|u) = \frac{(p(i|V_u) - p(i|\mathcal{C}))^2}{p(i|\mathcal{C})} \tag{8.4}$$

8.2.2.4 *Kullback-Leibler divergence (KLD)*

KLD is a non-symmetric measure for assessing the relative entropy between two probability distributions. Carpineto et al. (2001) proposed its use for PRF obtaining good results. The idea behind this method is to choose those terms of the pseudo-relevant set which diverge more from the collection in terms of entropy.

$$p_{KLD}(i|u) = p(i|V_u) \log \frac{p(i|V_u)}{p(i|\mathcal{C})} \tag{8.5}$$

8.3 NEIGHBORHOOD SIZE NORMALIZATION

Neighborhoods size can be a good indicator of users' uncommonness.

When we use a hard clustering algorithm to compute neighborhoods, the number of users in each cluster is variable. Even algorithms such as kNN can lead to neighborhoods with different sizes: a similarity measure based on the common occurrences among users may not be able to find k neighbors for all users when k is too high or when the collection is very sparse. In these cases, the information provided by the neighborhood is even more important since the user differs strongly from the collection. In IR, this situation would be associated with difficult queries that returned a very limited amount of documents. Therefore, the information of the relevant set should be promoted while the global collection information should be demoted.

We incorporate this intuition into the recommendation framework by biasing the probability estimate. We can normalize the MLE by dividing the estimate by the number of users in the population (neighborhood or collection) as follows:

$$p_{nmle}(i|X) \overset{\text{rank}}{=} \frac{1}{|\mathcal{U}_X|} \frac{\sum_{u \in \mathcal{U}_X} r(u,i)}{\sum_{u \in \mathcal{U}_X,\, j \in \mathcal{I}_X} r(u,j)} \tag{8.6}$$

This normalized maximum likelihood estimate (NMLE) does not affect RSV because this scoring function does not rely on the probability in the collection. Therefore, the resulting score would only be rescaled by a constant and the ranking would remain the same. However, we argue that NMLE can improve the effectiveness of CHI2 and KLD functions as we experimentally see in the next section.

8.4 EXPERIMENTS

We first compare the efficiency of the techniques proposed in this chapter with the relevance modeling approach to recommender systems. Then, we evaluate the effectiveness in terms of ranking accuracy, diversity and novelty.

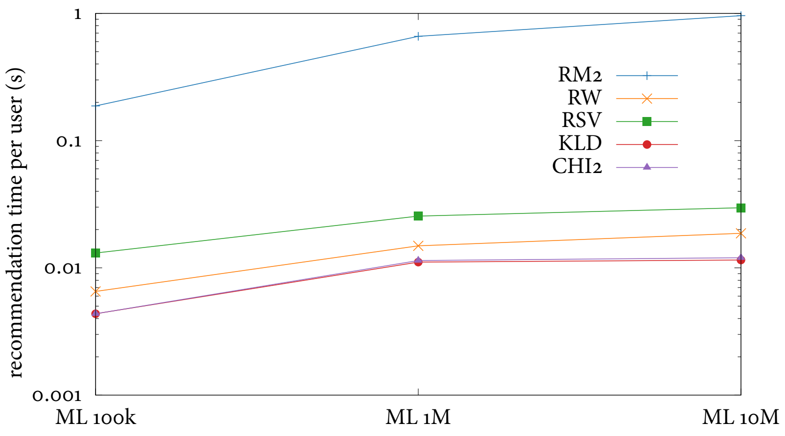


Figure 8.1: Recommendation time per user (in logarithmic scale) using RM2, RW, RSV, CHI2 and KLD algorithms on the MovieLens 100k, 1M and 10M datasets.

8.4.1 *Efficiency experiment*

The principal motivation for this work was to propose more efficient recommendation techniques based on PRF methods than relevance models (RM2). To assess the efficiency of our proposals, we measured the user recommendation times on the MovieLens 100k, 1M and 10M datasets. The neighborhoods are precomputed using the kNN algorithm with cosine similarity ($k = 100$). Since computing the neighborhoods is a common phase for each method, we disregard this stage. We measured the running time of the algorithms in a desktop computer with an Intel i7-4790 @3.60GHz and 16 GB DDR3 1600 MHz.

Our proposal are orders of magnitude faster than RM2.

Figure 8.1 illustrates the recommendation times on the three datasets. We report times (in logarithmic scale) for RM2, RSV, RW, CHI2 and KLD. These results demonstrate that the proposed approaches are dramatically faster than RM2 (our proposals obtain speed-ups up to 100x) meanwhile the variations in time among our proposed methods are small. We reported the running times for the normalized maximum likelihood estimate, but the differences in time between both probability estimates (MLE and NMLE) were insignificant.

Method	ML 100k	ML 1M	R3-Yahoo	LibraryThing
RM2	50	75	150	50
RW	50	50	150	50
RSV	50	50	150	50
CHI2-MLE	100	225	150	600
CHI2-NMLE	175	550	500	300
KLD-MLE	350	200	575	175
KLD-NMLE	50	75	300	50

Table 8.1: Optimal number of neighbors for the *k*NN algorithm with cosine similarity to maximize the value of nDCG@100 on the MovieLens 100k and 1M, R3-Yahoo and LibraryThing datasets for the RM2, RW, RSV, CHI2-MLE, CHI2-NMLE, KLD-NMLE and KLD-NMLE recommenders.

8.4.2 *Effectiveness experiment*

We present now the results of our methods on the MovieLens 100k, MovieLens 1M, R3-Yahoo and LibraryThing collections. We use the *k*NN algorithm with cosine similarity to compute the neighborhoods and tuned *k* from 25 to 700 neighbors (in steps of 25) to maximize the value of nDCG@100. For the sake of reproducibility, we report the optimal value of *k* in Table 8.1. As the baseline, we use the optimal version of RM2 with tuned smoothing and prior parameters (see Table 5.2).

The values of nDCG@100, Gini@100 and MSI@100 are reported in Table 8.2. Regarding ranking accuracy, on the MovieLens datasets, RM2 is the best recommender algorithm. However, on the R3-Yahoo and LibraryThing datasets, CHI2-NMLE outperforms RM2. On the R3-Yahoo dataset, RM2 is not statistically better than CHI2-NMLE, KLD-MLE and KLD-NMLE. On LibraryThing, CHI2-MLE, CHI2-NMLE and KLD-MLE are statistically better than RM2. These results may indicate that RM2 tends to work better in denser collections while the proposed scoring functions work better in sparser scenarios. However, among the proposed techniques, there is no clear winner and experiments should be carried out to select the best scoring function in a particular scenario.

Regarding the proposed neighborhood size normalization, the experiments show that NMLE outperforms the regular MLE in accuracy except for the KLD technique on the LibraryThing dataset. For CHI2, this improvement is statistically significant on three out of four datasets while, for KLD, it is significant on the MovieLens datasets. This empirical evi-

Method	Metric	ML 100k	ML 1M	R3-Yahoo	LibraryThing
RM2	nDCG	**0.4953**^{bcdefg}	**0.4296**^{bcdefg}	0.0717^{bcd}	0.2385^{bcg}
	Gini	0.2637	0.1637	0.4769	0.0319
	MSI	180.45	182.75	339.65	417.57
RW	nDCG	0.4827^{cdef}	0.4114^{cdef}	0.0704^{d}	0.2182^{c}
	Gini	0.2341	0.1331	0.2937	0.0348
	MSI	172.72	171.87	302.82	326.95
RSV	nDCG	0.4825^{def}	0.4112^{def}	0.0703^{d}	0.2180
	Gini	0.2338	0.1329	0.2940	0.0346
	MSI	172.60	171.80	302.91	326.69
CHI2 MLE	nDCG	0.2916	0.2775	0.0628	0.2605^{abcfg}
	Gini	0.3745	0.3895	0.4429	0.1496
	MSI	233.63	262.21	333.12	442.55
CHI2 NMLE	nDCG	0.4639^{df}	0.3966^{df}	**0.0726**^{bcdf}	**0.2610**^{abcfg}
	Gini	0.2947	0.1677	0.4136	0.1128
	MSI	190.77	188.34	327.74	400.18
KLD MLE	nDCG	0.4207^{d}	0.3393^{d}	0.0709^{d}	0.2543^{abcg}
	Gini	0.3168	0.3190	0.6064	0.0891
	MSI	199.23	237.88	371.56	396.31
KLD NMLE	nDCG	0.4839^{def}	0.4195^{bcdef}	0.0715^{bcd}	0.2337^{bc}
	Gini	0.2806	0.1540	0.3037	0.0669
	MSI	185.27	179.59	306.48	359.25

Table 8.2: Values of nDCG@100, Gini@100 and MSI@100 for each recommender approach on the MovieLens 100k and 1M, R3-Yahoo and LibraryThing datasets. Statistically significant improvements in nDCG@100 according to permutation test ($p < 0.05$) with respect to RM2, RW, RSV, CHI2-MLE, CHI2-NMLE, KLD-NMLE and KLD-NMLE are superscripted with a, b, c, d, e, f and g, respectively. Highest value of nDCG@100 for each dataset is indicated in bold.

dence supports the idea that the size of the neighborhoods is an important factor to model in recommender systems.

We also find that the recommendation approaches with the highest values of nDCG present lower figures of diversity and novelty than the other techniques on each dataset except on the LibraryThing collection. This result is in line with the accuracy-diversity trade-off (Zhou et al. 2010b).

8.5 CONCLUSIONS

Since relevance models are an effective tool for top-N recommendation, this work aimed to assess whether we could adapt other simpler PRF methods to the same task. The results of this investigation revealed that, indeed, more efficient PRF techniques constitute effective recommender approaches. The experiments showed that the proposed recommendation algorithms (RW, RSV, KLD and CHI2) are orders of magnitude faster than RM2 for recommendation. These alternatives offer important improvements in terms of computing time while incurring, in some cases, in a modest decrease of accuracy.

Since these methods lack parameters and only rely on the neighborhood information, their effectiveness depends heavily on the quality of the neighborhoods. In previous work, we used the *k*NN algorithm with Pearson's correlation similarity (Valcarce et al. 2016b,e). In this chapter, we used cosine similarity which resulted in improved accuracy. Therefore, exploring other techniques for building neighborhoods may lead to important improvements. In the next part, we address how to compute better neighborhoods in a collaborative filtering scenario.

Part IV

IMPROVING NEIGHBORHOODS

There is nothing like looking, if you want to find something. You certainly usually find something, if you look, but it is not always quite the something you were after.

— J. R. R. Tolkien,
The Hobbit

9 FINDING AND ANALYZING NEIGHBORHOODS

In the previous part, we studied the adaptation of pseudo-relevance feedback techniques to different recommendation tasks which led us to the formulation of neighborhood-based collaborative filtering techniques. This family of recommender systems (also known as memory-based recommenders) tends to be simple, effective and interpretable, but their performance is strongly tied to the clustering strategies used to compute the neighborhoods.

In this chapter, we show that there is room for improvement in neighborhood computation process. We first propose a very efficient memory-based algorithm called weighted sum recommender (WSR) after studying a state-of-the-art neighborhood-based approach. We then build an oracle for WSR which yields approximately optimal neighborhoods for this technique and find that there is a large gap in effectiveness. By considering the output of the oracle as ground truth data, we perform an analytical study of these neighborhoods to characterize them. Our goal is to find those properties that the ground truth neighborhoods satisfy to improve the recommendations produced by WSR.

Neighborhood computation is an area with potential for improvement.

As a result of our analysis, we propose to change the user profile size normalization that cosine similarity uses to improve the neighborhoods computed with the *k*NN algorithm. In addition, we present a simpler oracle which leads us to include the IDF effect on the cosine formulation. This work sheds light on the benefits of this type of analysis and paves the way for future research in the characterization of good neighborhoods for collaborative filtering.

The contributions presented in this chapter have been previously published. On the one hand, we derived the formulation WSR and studied its effectiveness (Valcarce et al. 2016e). On the other hand, we also designed the oracle strategies and proposed improvements in cosine similarity (Valcarce et al. 2018c).

9.1 INTRODUCTION

Neighborhood-based techniques are straightforward and efficient and their output is more easily explainable than the one from model-based recommenders (Ning et al. 2015). Nevertheless, model-based techniques are regarded as the most effective ones (Koren and Bell 2015). Although these model-based recommenders tend to attract more attention nowadays, we believe that there is still room for improvement in neighborhood-based systems. We devote this chapter to investigating this claim further.

While obvious, it is important to remark that the effectiveness of neighborhood-based methods depends largely on the quality of neighborhoods.

The performance of a neighborhood-based recommender depends heavily on the quality of the neighborhood (Bellogín and Parapar 2012; Cremonesi et al. 2010; Ning et al. 2015). Therefore, we seek to study how much we can increase the quality of the recommendations by improving the neighborhoods. To study the scope for improvement of these recommenders, we propose a simple neighborhood-based recommender called weighted sum recommender (WSR). Our proposal provides better results than NNCosNgbr in top-N recommendation which has been regarded as an effective neighborhood-based approach (Cremonesi et al. 2010).

To obtain an upper bound of the performance of WSR, we can build an oracle that produces ideal neighborhoods. Then, we can measure the maximum scope for improvement by comparing the neighborhoods computed with a state-of-the-art technique against those calculated by the oracle. We choose the kNN algorithm with cosine similarity as the neighborhood technique. However, creating this oracle poses a challenge since obtaining the optimal neighborhood is an NP-hard task. To tackle this problem, we propose a greedy heuristic which can find an approximation of the ground truth neighborhoods. We also propose a second oracle that, although it shows a more modest performance, is more appropriate for current grouping strategies which are based principally on co-occurrences of ratings.

The large gap in the quality of the recommendations motivates the development of an analytical methodology to characterize the neighborhoods produced by the oracles. We aim to find those features that cosine similarity is missing or incorrectly exploiting. Our methodology provides with the tools to design two variants of cosine similarity that improve the original formulation of this similarity measure. Extensive experimentation on four datasets confirms that our new formulations of cosine similarity generate neighborhoods that produce better recommendations regarding ranking quality, diversity and novelty.

9.2 NEIGHBORHOOD-BASED RECOMMENDER SYSTEMS

Neighborhood-based collaborative filtering techniques are also known as memory-based recommenders. Henceforth, we will use both terms interchangeably. We can distinguish either user-based or item-based approaches within the neighborhood-based techniques. User-based systems emit recommendations using the feedback from like-minded users (known as *neighbors*) while item-based approaches recommend items that are similar to those the target user liked (Ning et al. 2015). Since both are collaborative filtering approaches, the similarity among users or items is always computed in terms of the user-item interactions.

There is no silver bullet: item-based and user-based approaches have their own advantages.

Item-based approaches are usually preferred (Cremonesi et al. 2010; Deshpande and Karypis 2004; Ning et al. 2015) because the number of items is usually smaller than the users. This enables the efficient computation of the neighborhoods. Also, they have been shown to report better results in terms of accuracy than user-based approaches (Deshpande and Karypis 2004; Ning et al. 2015). Also, item-based recommendations are easy to justify with explanations such as "you would like item B because you liked item A". However, item-based methods may generate less serendipitous recommendations because they tend to recommender similar items to those rated by the user (Ning et al. 2015). In contrast, user-based approaches recommend items that similar users enjoyed. In fact, it is possible to suggest items that strongly differ from the ones rated by the target user.

Neighborhoods are computed by means of clustering algorithms.

We can usually differentiate two phases in neighborhood-based recommenders: first, the construction of the neighborhoods (i. e., the set of neighbors) and, second, the computation of recommendations employing those neighborhoods. Previous works have emphasized the importance of the first phase in the quality of the recommendations (Cremonesi et al. 2010; Ning et al. 2015; Valcarce et al. 2016e). Although strict partitioning clustering techniques such as k-means (Xue et al. 2005), posterior probability clustering (Parapar et al. 2013) or normalized cut (Bellogín and Parapar 2012) have been used with memory-based recommenders, the k nearest neighbors (kNN) algorithm is the dominant technique for computing neighborhoods in a collaborative scenario (Ning et al. 2015). In the case of user-based recommenders, kNN computes a specific neighborhood for each user consisting of the top k users with higher similarity to the target user. In contrast to partitioning clustering algorithms, kNN is an overlapping technique because a user may appear in more than one neighborhood.

In the following, we present non-normalized cosine neighborhood (NNCosNgbr), a popular neighborhood-based approach, and our proposal weighted sum recommender (WSR).

9.2.1 *Non-normalized cosine neighborhood*

Cremonesi et al. (2010) proposed non-normalized cosine neighborhood (NNCosNgbr), an effective item-based neighborhood technique. To compute the k nearest neighbors, this method uses cosine similarity instead of Pearson's correlation coefficient because the former is computed over all the ratings while the latter relies only on the shared ratings. Moreover, Cremonesi et al. (2010) introduced a shrinking factor based on common ratings into the similarity metric (Koren 2008). This shrunk similarity penalizes very sparse vectors. Additionally, NNCosNgbr removes user-item biases according to the definition in (Koren 2008). The resulting formulation to predict the score $\hat{r}(u,i)$ for the user u and the item i is given by the following expression:

$$\hat{r}(u,i) = b(u,i) + \sum_{j \in J_i} \mathrm{sim}(i,j)\ [r(u,j) - b(u,j)] \tag{9.1}$$

where $b(u,i)$ denotes the bias for the user u and the item i (Koren 2008); $\mathrm{sim}(i,j)$, the shrunk cosine similarity between items i and j (Cremonesi et al. 2010), and J_i, the neighborhood of the item i.

The major difference between this method and the standard neighborhood approach (Ning et al. 2015) is the absence of the normalizing denominator. Since we are not interested in predicting ratings, we do not worry about getting scores in a fixed range. On the contrary, this method fosters those items with high ratings by many neighbors (Cremonesi et al. 2010; Deshpande and Karypis 2004; Koren 2008).

9.2.2 *Weighted sum recommender*

WSR stems from NNCosNgbr.

Our recommendation algorithm stems from NNCosNgbr. First, we decide to keep the biases, instead of eliminating them. Removing user-item biases produces notable improvements in rating prediction because it provides more accurate rating estimates (Koren 2008; Ning et al. 2015). Nevertheless, on the top-N recommendation, this is ineffective and may decrease the quality of the rankings that top-N recommenders are concerned about. Moreover, this process adds an extra hyperparameter to the model (Koren 2008).

Next, we focus on the similarity metric. Cremonesi et al. (2010) introduced a shrinking factor into the cosine metric to promote those similarities that are based on many shared ratings. This shrinkage procedure has shown good results in previous studies based on error metrics (Koren 2008; Ning et al. 2015) at the expense of adding a hyperparameter to the model. However, we found that its inclusion is detrimental and plain cosine similarity works better. This is reasonable because the main advantage of cosine similarity over other metrics such as Pearson's correlation coefficient is that it considers non-rated values as zeros. In this way, cosine already takes into account the amount of co-occurrence between vectors of ratings which makes unnecessary the use of a shrunk similarity.

Finally, we also propose a user-based variant of this algorithm. Depending on the characteristics of the dataset, a user-based or an item-based version may work better than the other. In the end, the final formulation of our proposal is a weighted sum of the ratings of the neighbors, which we call *weighted sum recommender* (WSR). Next, we present the user-based and item-based variants:

$$\hat{r}(u,i) = \sum_{v \in V_u} \cos(u,v)\, r(v,i) \tag{9.2}$$

$$\hat{r}(u,i) = \sum_{j \in J_i} \cos(i,j)\, r(u,j) \tag{9.3}$$

where $\cos(\cdot,\cdot)$ is the cosine similarity between a pair of users or items. The cosine similarity between two users is given by Equation (5.3); the item-based similarity is calculated as in Equation (6.12). V_u denotes the neighborhood of user u as J_i represents the neighborhood of item i.

We can use the kNN algorithm with cosine similarity to compute the neighborhoods. Note, in this case, two different uses of cosine similarity. On the one hand, we can use this similarity measure in the kNN algorithm to compute the neighborhoods. On the other hand, this similarity is the weight in the scoring function of WSR.

9.2.3 *Comparing WSR with NNCosNgbr*

Now we compare the effectiveness of WSR and NNCosNgbr algorithms on the MovieLens 100k and 1M, R3-Yahoo and LibraryThing datasets. We compare NNCosNgbr against the user-based and the item-based variant of WSR in Table 9.1. We tuned the number of neighbors from 25 to 200 is steps of 25 and the shrinking factor from 25 to 150 in steps of 25.

Method	Metric	ML 100k	ML 1M	R3-Yahoo	LibraryThing
NNCosNgbr	nDCG	0.2227	0.1980	0.0567	0.0852
	Gini	0.3438	0.2407	0.2341	0.0659
	MSI	230.14	228.00	386.78	546.47
WSR-UB	nDCG	**0.4857***	**0.4138***	0.0705*	0.2213*
	Gini	0.2375	0.1356	0.3208	0.0768
	MSI	173.86	172.76	309.52	364.70
WSR-IB	nDCG	0.4833*	0.4035*	**0.0727***	**0.3085***
	Gini	0.2560	0.1516	0.3356	0.2768
	MSI	177.34	178.95	315.05	461.73

Table 9.1: Values of nDCG@100, Gini@100 and MSI@100 for the neighborhood-based recommenders on MovieLens 100k and 1M, R3-Yahoo and LibraryThing. Statistically significant improvements in nDCG@100 according to permutation test ($p < 0.05$) with respect to NNCosNgbr are indicated with a star. Highest value of nDCG@100 for each dataset is indicated in bold.

WSR is a simple yet effective recommender.

Both user-based and item-based versions of WSR significantly outperform NNCosNgbr on all datasets in ranking accuracy. The user-based approach reported the best figures on the MovieLens datasets while the item-based algorithm yielded the best results on the sparser songs and books collections. This result agrees with the literature about neighborhoods methods (Cremonesi et al. 2010; Deshpande and Karypis 2004; Ning et al. 2015): item-based approaches tend to work well on sparse datasets because they compute similarities among items which often contain denser information than users. Additionally, on the R3-Yahoo and LibraryThing datasets, WSR produces even more diverse recommendations than NNCosNgbr in spite of the accuracy improvement.

9.3 A GREEDY NEIGHBORHOOD ORACLE

This chapter is devoted to studying the room for improvement of neighborhoods. In particular, we aim to find to what extent we can increase the quality of the recommendations of memory-based approaches by improving the neighborhoods. We take WSR-UB as a neighborhood-based recommender and seek to improve the similarity measure used in finding neighborhoods using the *k*NN algorithm. Nonetheless, the proposed analysis can be performed to improve other neighborhood computation approach. We focus on user-based methods and leave the exploration of item-based models as future work. Nevertheless, the presented approach can be easily extended to the item-based scenario.

We propose to build an oracle that produces the optimal neighborhoods for each user to measure how much we can improve the selection of neighbors. This oracle would provide an upper bound of the maximum performance that we can achieve. Nevertheless, the construction of this oracle is challenging: there exist 2^{n-1} possible neighborhoods for each user where n represents the number of users in the dataset. A brute force approach for finding the best neighborhood for a particular user consists in testing all the possible neighborhoods and return the one that produces the recommendations with the highest nDCG@100 value. This algorithm would require $\Theta(2^n)$ evaluations. In fact, finding the best neighborhood is analogous to the *subset selection* problem for regression. Subset selection is the problem of finding the subset of variables for learning a linear regression model that minimizes the prediction error (Miller 2002). Unfortunately, subset selection is an NP-hard problem in general (Welch 1982) and so is building our oracle.

Building a perfect oracle is infeasible, but we can approximate one.

Since computing an exact solution is infeasible, we propose to use a greedy heuristic to find an approximated solution. We took inspiration from *forward selection*, a strategy that is used to deal with the subset selection problem (Miller 2002). Our greedy oracle computes the best neighborhood for a user u up to a size k. The algorithm begins by testing all the possible neighborhoods of the user u of size 1. It chooses the neighborhood that gives the best performance in terms of an objective function we call *eval* (more details in Section 9.3.1). This function takes the input user u and the candidate neighborhood V and returns the goodness of the given neighborhood. Then, we take the best neighborhood of size 1 as input for building the neighborhood of size 2 by selecting a user such that when added to the previous neighborhood obtains the best value for the *eval* function. We repeat this procedure until we reach neighborhoods of size k. We use a greedy approach in each step because we stem from the best neighborhood of size k to obtain the best neighborhood of size $k + 1$. In the same way that forward selection chooses the best feature in each step, we choose the best neighbor to be added to the neighborhood in each step. At this point, given a user u, we have the best neighborhoods of sizes 1 up to k. Finally, we return the smallest neighborhood that achieves the highest score.

This algorithm follows a greedy approach since it searches the best neighborhood of size k from the solution for $k - 1$ and thus returns a suboptimal solution. The oracle requires $\Theta(nk)$ calls to the *eval* function where n represents the number of users and k the size of the neighborhood. Next, we define the evaluation procedure of the *eval* function.

9.3.1 *Evaluation function*

To measure the goodness of a neighborhood for a particular user, we use the *eval* function which assesses the quality of the recommendations produced for a particular user with the given neighborhood. To assess the quality of the recommendations, we employed normalized discounted cumulative gain with a cut-off at 100 (nDCG@100).

In spite of the decrease in computing complexity from the brute force approach to the greedy heuristic, the greedy oracle algorithm is still computationally expensive because each call to the *eval* function requires ranking almost all items in the collection. Additionally, the oracle algorithm should be executed for each user of the collection. For this reason, we introduced the parameter k into the algorithm that limits the maximum size of the neighborhoods.

9.3.2 *Upper Bound of Accuracy*

With the devised oracle, we can approximate the upper bound of the ranking accuracy of a neighborhood-based recommender system. We can only get an approximated upper bound because the oracle follows a greedy heuristic. Due to its computational cost, we employed a small collection, the MovieLens 100k dataset, to test this oracle.

We compared the accuracy of the recommendations produced by WSR when using the greedy oracle strategy and the kNN algorithm with cosine similarity. We varied the value of k from 10 to 300. In the case of the oracle, k denotes the maximum amount of users in the neighborhood and, in the case of kNN, the number of neighbors. Figure 9.1 shows the results of this experiment. We observe a huge gap in terms of nDCG@100 between the oracle and kNN with cosine similarity. Since our oracle strategy provides only an approximation of the optimal neighborhood, the actual gap will be even higher than the one shown in Figure 9.1.

The optimal value of k for kNN with cosine similarity is around 50. As we increase the size of the neighborhoods, the quality of the recommendations decreases because we are introducing more noise into the model. In contrast, as expected, the performance of the oracle is always increasing with k because this strategy selects the best neighborhood up to size k. We can explain now the reason why the greedy oracle does not

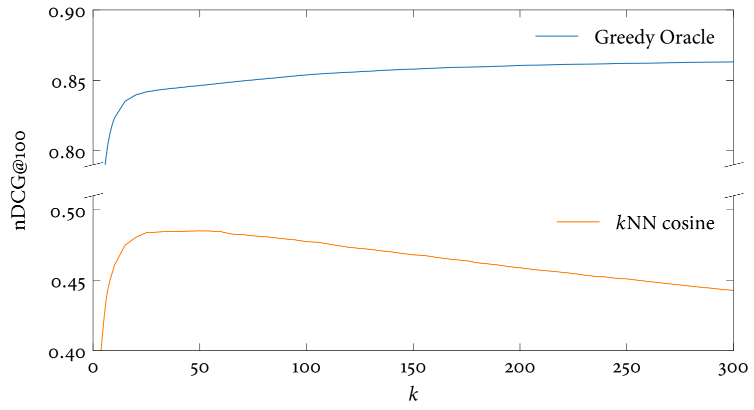


Figure 9.1: Values of nDCG@100 of the recommendations generated by WSR using the neighborhoods produced by the greedy oracle and by k-NN using cosine similarity when varying the parameter k.

METHOD	k	NDCG@100	GINI@100	MSI@100
kNN Cosine	50	0.4857	0.2375	173.86
Greedy Oracle	300	0.8631	0.2664	168.08

Table 9.2: Values of nDCG@100, Gini@100 and MSI@100 using WSR with the greedy oracle and k-NN with cosine similarity on MovieLens 100k.

fix the size of the neighborhoods to be exactly k. The rationale is that it might produce undesired results. For example, let k be the forced size of the neighborhood and l the optimal size when $k > l$ the oracle will produce l good neighbors and $k - l$ spurious neighbors. To avoid punishing the recommendation after finding l good neighbors, the oracle selects users whose ratings differ from those of the rest of the neighborhood. Those spurious neighbors are chosen not because they are valuable to the recommendation but because they do not hurt or hurt the quality of the recommendations minimally.

As we can see in Figure 9.1, although the accuracy of the oracle increases with higher values of k, the improvements are very slight. Due to computational resources, we set k to 300 to establish an approximated upper bound. We also measure diversity and novelty (see Table 9.2), but the oracle shows that there is only a small room for improvement in terms of diversity.

In the next section, we present an analysis of the neighborhoods that the oracle produces with the objective of designing more effective techniques for finding neighborhoods.

9.4 NEIGHBORHOOD ANALYSIS

We can gain important insights from studying ground truth neighborhoods.

We argue that the oracle designed in the previous section not only provides an upper bound of the maximum performance of a neighborhood-based recommender but also provides ground truth neighborhoods. In this section, we seek to characterize those ideal neighborhoods through an analytical study of their features. To the best of our knowledge, this is the first exploratory analysis of its kind. To allow comparisons, we compared the ground truth neighborhoods with those obtained using kNN with cosine similarity. We aim to gain insights into what features a good neighbor possesses. In this way, we may be able to improve the original formulation of cosine similarity.

We compared several features between the neighbors produced by the oracle and by kNN with cosine similarity such as the sum of ratings, the number of rated items, the average rating, the overlap in rated items between the target user and the neighbor or the popularity of the rated items. In particular, we found the size of the user profiles an interesting feature to analyze carefully.

9.4.1 *User Profile Size*

In a collaborative filtering scenario, the users' profiles are made of ratings. However, there is high diversity among them. It is common to find highly active users and users with little involvement due to several reasons (such as age, seniority or interest in the service). We argue that the size of the user profiles is an influential variable in the selection of neighbors. We can measure the size of a user profile as the sum or as the number of the ratings emitted by that user. We can use either metric because both of them are strongly correlated (Pearson's $\rho > 0.97$ in the MovieLens 100k dataset). We studied the distributions of the sizes of the user profiles of neighborhoods provided by the oracle and the ones produced by kNN with cosine similarity and discovered that those distributions are substantially different. We illustrate them in Figure 9.2. The oracle selects neighbors with a notably smaller profile than kNN with cosine similarity. Large user profiles may contain more noise and according to Figure 9.2, cosine similarity does not penalize them properly.

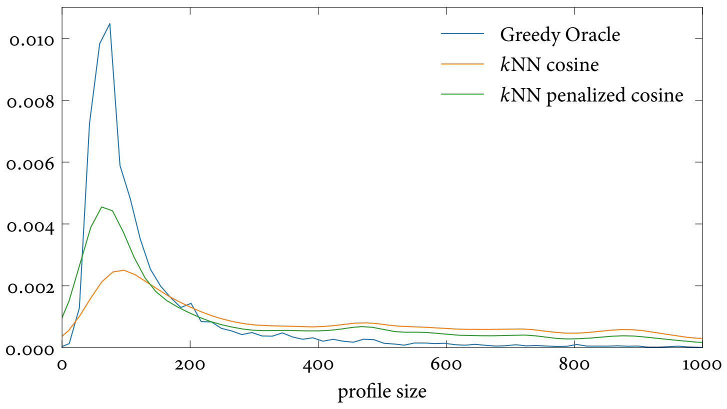


Figure 9.2: Distribution of the sizes of the user profiles (measured as the sum of ratings) of the ground truth neighborhoods produced by the oracle and the ones provided by k-NN with cosine and penalized cosine similarities when $k = 50$ and $\alpha = 1.3$ on the MovieLens 100k dataset.

9.4.2 *Penalized Cosine*

In light of these findings, we seek to adapt cosine similarity to penalize users with large profiles. As the formulation of this measure shows, cosine already penalizes the similarity by the product of the norms of both profiles. If we represent a user profile as a vector of ratings, the ℓ_2-norm of a user profile is given by the square root of the sum of the squared ratings. When computing the neighbors for a particular user u, we can simplify the formulation of cosine similarity removing the norm of the target user u because all the similarities would be scaled by a constant which do not affect the ranking procedure of kNN. The formula of this rank-equivalent cosine similarity is the following:

$$\cos'(u, v) = \frac{\sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} r(u, i)\, r(v, i)}{\sqrt{\sum_{i \in \mathcal{I}_v} r(v, i)^2}} \tag{9.4}$$

To penalize more those users with larger profiles, we must increase the normalization in the denominator of cosine similarity. To this end, we draw inspiration from information retrieval techniques. The process of computing neighbors using cosine similarity is analogous to the document ranking procedure in the vector space model (Salton et al. 1975) where the target user plays the role of the query, and the rest of the users are the documents in the collection. In this way, choosing the k nearest neighbors is the same as taking the top k results using the user as the query. Following this analogy, we apply the pivoted length normalization procedure (Singhal et al. 1996).

Pivoted length normalization is a state-of-the-art retrieval model.

In ad hoc retrieval, Singhal et al. (1996) found that cosine similarity demotes large document too much. Therefore, they modified the normalization scheme of cosine similarity to reduce the penalty controlled by a parameter α between zero and one. This document length normalization term led to notable effectiveness improvements in the vector space model (Zhai and Massung 2016). In this normalization approach, a value of zero would imply no penalty at all while a value of 1 would yield the original cosine normalization. Although Singhal et al. (1996) did not consider increasing the cosine normalization (as we seek to do), we can set α to a value higher than 1 to produce this effect. The penalized formulation of cosine similarity is given by:

$$\text{penalized\_cos}(u,v) = \frac{\sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} r(u,i)\, r(v,i)}{(1-\alpha) + \alpha \frac{\sqrt{\sum_{i \in \mathcal{I}_v} r(v,i)^2}}{\frac{1}{|\mathcal{U}|} \sum_{w \in \mathcal{U}} \sqrt{\sum_{i \in \mathcal{I}_w} r_{w,i}^2}}} \tag{9.5}$$

We can easily see that when $\alpha = 1$, we obtain a similarity measure which is rank-equivalent to cosine similarity. The difference is that the values are scaled by a constant which is the average user profile size in the collection $\frac{1}{|\mathcal{U}|} \sum_{w \in \mathcal{U}} \sqrt{\sum_{i \in \mathcal{I}_w} r_{w,i}^2}$. Note that if we precompute the average user profile size, the computational complexity of the penalized cosine remains the same compared to the original formulation.

Figure 9.2 shows that the distribution of the sizes of the user profiles of penalized cosine is much more similar to the distribution of the oracle than cosine similarity on the MovieLens 100k dataset. Therefore, the proposed penalty for cosine similarity is an effective technique to promote neighbors with small profiles. Nevertheless, we still need to verify if this approach produces better recommendations than the original cosine formulation. To this end, we evaluated the recommendations produced by WSR with neighborhoods computed using kNN with cosine and penalized cosine similarities on four different datasets. In addition to MovieLens 100k, we use MovieLens 1M, R3-Yahoo and LibraryThing datasets. We tuned the values of k from 25 to 200 in steps of 25 and α from 0.5 to 2.0 in steps of 0.05 to maximize the value of nDCG@100. We report the values of nDCG@100, Gini@100 and MSI@100 in Table 9.3. For the sake of reproducibility, we also report the values of the parameters in Table 9.6. We used the permutations test ($p < 0.05$) to analyze whether the improvements in terms of nDCG@100 and MSI@100 are statistically significant. We cannot apply a paired test to Gini because it is a global metric.

Similarity	Metric	ML 100k	ML 1M	R3-Yahoo	LibraryThing
Cosine	nDCG	0.4857	0.4138	0.0705	0.2255
	Gini	0.2375	0.1356	0.3208	0.0417
	MSI	173.86	172.76	309.52*	333.50
Penalized Cosine	nDCG	0.4889*	0.4194*	0.0709	0.2266
	Gini	0.2516	0.1446	0.2863	0.0471
	MSI	177.97*	176.41*	302.39	339.05*

Table 9.3: Values of nDCG@100, Gini@100 and MSI@100 on MovieLens 100k, MovieLens 1M, R3-Yahoo and LibraryThing using different similarity measures with k-NN and WSR as recommender algorithm. Statistically significant improvements in nDCG@100 or MSI@100 using permutations test ($p < 0.05$) are indicated with a star.

The results show that penalized cosine outperforms cosine similarity in ranking accuracy, diversity and novelty on all the datasets except for diversity and novelty on R3-Yahoo. The improvements in novelty are statistically significant in the three collections although the improvements on accuracy are only significant on the MovieLens datasets. Table 9.6 shows that the value of α that maximizes nDCG@100 is always greater than 1 except for the R3-Yahoo dataset where it is strictly less than 1. These findings support the idea that profile size normalization may be beneficial in some recommendation scenarios. However, the amount of normalization (regulated by the hyperparameter α) is domain-dependent.

This finding has been brought to light as a result of characterizing the neighborhoods produced by our proposed oracle. Therefore, these results show that the analysis of the features of the ground truth neighborhoods computed with the greedy oracle is an effective tool for improving neighborhood techniques. In particular, we have found that profile size normalization can be tuned to improve the neighborhoods retrieved with cosine similarity. However, we can do better. In the next section, we propose a simpler oracle which leads us to another improvement of cosine similarity which does not require any extra parameter.

The analysis of the ground truth neighborhoods led us to the beneficial profile length normalization.

9.5 A COSINE-BASED ORACLE

The oracle presented in Section 9.3 is a useful tool for approximating an upper bound of a neighborhood-based recommender. Nevertheless, in practice, obtaining such neighborhoods may be infeasible with similarity techniques that solely exploit co-occurrence of ratings among

We develop a simpler and more realistic oracle based on cosine similarity.

Method	nDCG@10	Gini@10	MSI@10
*k*NN Cosine	0.4857	0.2375	173.86
Cosine-based Oracle	0.5298	0.2508	174.97

Table 9.4: Values of nDCG@100, Gini@100 and MSI@100 using WSR with cosine similarity and the cosine-based oracle on the MovieLens 100k dataset. We set k to 50 neighbors.

user profiles. Cosine similarity but also many other similarity measures (such as Pearson's correlation) or clustering algorithms different to *k*NN are based on the notion of co-occurrence. There is no guarantee that co-occurrence alone is sufficient to compute the ideal neighborhoods that the greedy oracle provides. The greedy oracle may be surpassing the so-called *magic barrier* (Bellogín et al. 2014b; Herlocker et al. 2004) for this kind of algorithms. For this reason, we propose a cosine-based oracle that simply outputs the best neighborhood that cosine similarity can provide by varying the size of the neighborhood up to value k for each user in the collection. As we did with the greedy oracle, we chose the neighborhood that maximizes an evaluation metric—nDCG@100 in our case. This cosine-based oracle can also be seen as an adaptive *k*NN algorithm which employs the optimal k for each user.

We compared the quality of the recommendations produced using WSR with cosine similarity using 50 nearest neighbors against this cosine-based oracle with up to 50 neighbors per user. We present the results in terms of nDCG@100, Gini@100 and MSI@100 in Table 9.4. The cosine-based oracle provided better results than the baselines; however, its effectiveness seems much more achievable compared to the greedy oracle (see Table 9.2). As we did before with the greedy oracle, we seek to characterize the neighborhoods obtained by the cosine-based oracle to develop new formulations of cosine similarity.

9.5.1 *User Profiles Overlap*

We measure the overlap between two users with the Jaccard index.

We studied some features related to the neighborhoods as we did before with the greedy oracle. We found the overlap between the target user profile and each of the neighbors to be an interesting feature as previous studies have shown (Liu et al. 2014). We define the overlap between the

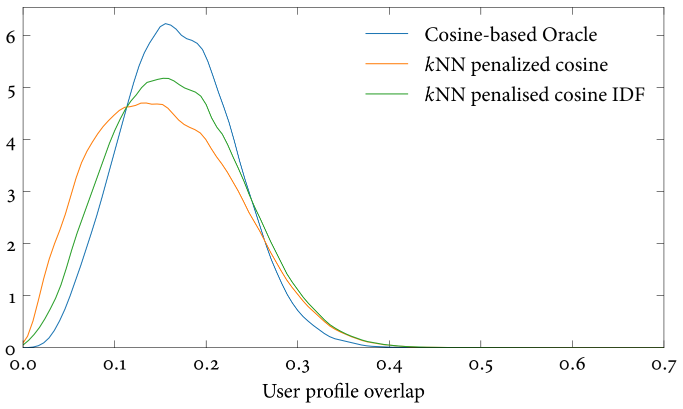


Figure 9.3: Distribution of user profiles overlap in the neighborhoods produced by the oracles and the ones provided by k-NN with penalized cosine and penalized cosine with IDF similarities when $k = 50$ on the MovieLens 100k dataset.

target user u and the neighbor v using the Jaccard index, i.e., the ratio between the intersection and the union of rated items by those users:

$$\text{Jaccard}(u, v) = \frac{|\mathcal{I}_u \cap \mathcal{I}_v|}{|\mathcal{I}_u \cup \mathcal{I}_v|} \tag{9.6}$$

Using the Jaccard index, we studied the distribution of the overlap among user and neighbors profiles. We did not employ the Jaccard index for computing neighborhoods directly because, in our experiments, it produced worse neighborhoods than cosine similarity. We used the cosine-based oracle and kNN with penalized cosine similarity and set k to 50 on the MovieLens 100k dataset. We illustrate the results in Figure 9.3. We observe that penalized cosine similarity chooses neighbors with a low overlap with the target user. The penalty introduced to favor users with short profiles provokes this effect. However, if the overlap between user and neighbor ratings is small, we may not be selecting like-minded users and thus producing bad recommendations. For this reason, we seek to modify penalized cosine similarity to improve the quality of the neighborhoods further.

9.5.2 *Penalized Cosine with IDF*

After analyzing the distribution of the sizes of the greedy oracle, we increased the profile size normalization of cosine to improve the plain

We improve the cosine similarity by introducing the IDF effect in its formulation.

cosine similarity. However, Figure 9.3 shows that this penalty can also hurt the overlap between the target user and the neighbors. A low overlap may drift the topics of the recommendations producing an undesirable outcome. To avoid selecting users with divergent interests, we propose to emphasize our focus on the items which are less popular. The rationale for this approach is based on the IDF effect, a well-known heuristic in information retrieval.

The inverse document frequency (IDF) measures the specificity of a term (Baeza-Yates and Ribeiro-Neto 2011). The IDF of a term is proportional to the inverse of the number of documents in the collection that contains that term. The idea is that terms that appear in a few documents are more informative than those that occur several times. Therefore, the IDF effect assumes that specific terms are more discriminative than generic ones. In Section 5.2.2.1, we established a connection between the IDF effect and item novelty. Promoting the IDF effect should increase the focus on less common items which are more discriminative than the popular ones. Since penalizing lengthy user profile reduces the overlap, we introduce the notion of IDF into the similarity measure to promote those users with more co-occurrence in unpopular items.

To introduce the IDF effect, we used the *inverse frequency smooth* weighting scheme as defined in (Baeza-Yates and Ribeiro-Neto 2011). We adapted its formulation from information retrieval to recommender systems. The resulting penalized cosine similarity with IDF is given by:

$$\text{penalized\_cos\_IDF}(u,v) = \frac{\sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} t(u,i)\, t(v,i)}{(1-\alpha) + \alpha \frac{\sqrt{\sum_{i \in \mathcal{I}_v} r(v,i)^2}}{\frac{1}{|\mathcal{U}|} \sum_{w \in \mathcal{U}} \sqrt{\sum_{i \in \mathcal{I}_w} r_{w,i}^2}}} \tag{9.7}$$

where:

$$t(u,i) = r(u,i)\ \text{idf}(i) = r(u,i) \log\left(1 + \frac{|\mathcal{U}|}{|\mathcal{U}_i|}\right) \tag{9.8}$$

where $|\mathcal{U}_i|$ refers to the number of users who rated the item i. Note that the computational complexity does not change after introducing the IDF heuristic. We need to compute this IDF term for each rating which can be done in constant time if we have cached the number of users that rated any item. Therefore, the scalability of our proposal is not compromised.

We compared the penalized cosine with IDF against the version without IDF. Figure 9.3 shows that the IDF version presents a higher overlap and is more similar to the cosine-based oracle.

To evaluate if the IDF effect also provides improvements in the quality of the recommendations, we tested penalized cosine with IDF on the four datasets. We compare against a representative set of neighborhood computation techniques as baselines:

PEARSON We use the *k*NN algorithm with Pearson's correlation coefficient as the pairwise similarity is the traditional approach to compute neighborhoods in the rating prediction task (Ning et al. 2015).

PPC Posterior probabilistic clustering (PPC) is a clustering technique based on non-negative matrix factorization (Ding et al. 2008) used by Parapar et al. (2013) in recommendation. It has two hyperparameters to tune: the number of clusters c from 25 to 200 in steps of 25 and the number of training epochs e. We found that 100 training epochs are enough for convergence on the four datasets.

NC Normalized cut is a spectral clustering algorithm which has been recognized as an effective technique for computing neighborhoods in user-based collaborative filtering (Bellogín and Parapar 2012). We tuned two hyperparameters of NC: the number of clusters c and the number of eigenvectors λ from 25 to 150 in steps of 25.

SHRUNK COSINE We use the *k*NN algorithm with the shrunk cosine similarity used in NNCosNgbr (Cremonesi et al. 2010). This shrinking factor is controlled by δ. We tuned this hyperparameter by grid search from 25 to 150 in steps of 25.

We measure the results with nDCG@100, Gini@100 and MSI@100 in Table 9.5. We also report the tuned values of the parameters in Table 9.6. The experiments showed that penalized cosine with IDF outperforms all baselines in terms of nDCG@100 on the four datasets. Additionally, these improvements are always statistically significant according to the permutations test ($p < 0.05$). Regarding diversity and novelty, only normalized cut provided better results than penalized cosine with IDF on R3-Yahoo and LibraryThing. Note that it is easy to improve diversity and novelty figures by reducing the accuracy due to the accuracy-diversity trade-off (Castells et al. 2015; Kunaver and Požrl 2017; Zhou et al. 2010b). Additionally, if we compare penalized cosine with and without IDF (see Table 9.3), we observe that the IDF effect provides better figures in all the metrics on the four datasets.

Therefore, the experiments support the introduction of the IDF heuristic into cosine similarity. The IDF effect is compatible with the profile size normalization added to the penalized version of cosine. As we can see in Table 9.6, the optimal value of α that maximizes nDCG@100 is not 1 on any dataset either including the IDF or not. This fact confirms

Method	Metric	ML 100k	ML 1M	R3-Yahoo	LibraryThing
Pearson	nDCG	0.4079	0.2252	0.0607	0.1518
	Gini	0.1189	0.0480	0.1642	0.0142
	MSI	136.57	139.20	273.81	281.33
PPC	nDCG	0.4049	0.3056^{a}	0.0652^{a}	0.1702^{ac}
	Gini	0.1859	0.0775	0.2764	0.0308
	MSI	155.18^{a}	150.60^{a}	297.91^{a}	315.08^{a}
NC	nDCG	0.4270^{ab}	0.3581^{ab}	0.0666^{a}	0.1662^{a}
	Gini	**0.2842**	**0.1821**	0.3318	0.0376
	MSI	**182.81**^{abdef}	**188.03**^{abdef}	309.04^{abe}	333.02^{ab}
Shrunk Cosine	nDCG	0.4717^{abc}	0.3891^{abc}	0.0705^{abc}	0.2206^{abc}
	Gini	0.2111	0.1098	0.3208	0.0503
	MSI	165.83^{ab}	163.73^{ab}	309.52^{abe}	342.54^{abce}
Cosine	nDCG	0.4857^{abcd}	0.4138^{abcd}	0.0704^{abc}	0.2255^{abcd}
	Gini	0.2375	0.1356	0.3107	0.0417
	MSI	173.86^{abd}	172.76^{abd}	305.26^{ab}	333.50^{ab}
Penalized Cosine	nDCG	0.4889	0.4194	0.0709	0.2266
	Gini	0.2516	0.1446	0.2863	0.0471
	MSI	177.97	176.41	302.39	339.05
Penalized Cosine with IDF	nDCG	**0.4927**^{abcde}	**0.4281**^{abcde}	**0.0721**^{abcde}	**0.2422**^{abcde}
	Gini	0.2517	0.1551	**0.3376**	**0.0596**
	MSI	178.65^{abde}	180.41^{abde}	**312.08**^{abcde}	**354.46**^{abcde}

Table 9.5: Values of nDCG@100, Gini@100 and MSI@100 on the MovieLens 100k, MovieLens 1M, R3-Yahoo and LibraryThing datasets using different neighborhood computation techniques and WSR as recommendation algorithm. Statistically significant improvements in nDCG@100 and MSI@100 according to permutation test ($p < 0.05$) with respect to Pearson, PPC, NC, shrunk cosine, cosine and penalized cosine with IDF are superscripted with a, b, c, d and e, respectively. Highest value for each metric on each dataset is indicated in bold.

that profile size normalization is important by itself. However, adding the IDF heuristic to the cosine formula provides a boost in ranking accuracy, diversity and novelty at no extra cost—we do not need to add any new parameter to the similarity measure.

Again, we observed that the optimal value of α is greater than 1 except on the R3-Yahoo where a value below the unit provides better recommen-

METHOD	ML 100K	ML 1M	R3-YAHOO	LIBRARYTHING
Pearson	$k = 475$	$k = 500$	$k = 475$	$k = 500$
PPC	$c = 25$	$c = 50$	$c = 150$	$c = 100$
NC	$c = 25$, $\lambda = 25$	$k = 100$, $\lambda = 25$	$c = 75$, $\lambda = 100$	$c = 75$, $\lambda = 150$
Shrunk Cosine	$k = 50$, $\delta = 25$	$k = 50$, $\delta = 25$	$k = 50$, $\delta = 50$	$k = 25$, $\delta = 25$
Cosine	$k = 50$	$k = 50$	$k = 125$	$k = 25$
Penalized Cosine	$k = 50$, $\alpha = 1.30$	$k = 75$, $\alpha = 1.3$	$k = 150$, $\alpha = 0.90$	$k = 25$, $\alpha = 1.15$
Penalized Cosine with IDF	$k = 75$, $\alpha = 1.45$	$k = 100$, $\alpha = 1.35$	$k = 150$, $\alpha = 0.95$	$k = 75$, $\alpha = 1.65$

Table 9.6: Parameters of neighborhood techniques when using WSR as recommender algorithm on the MovieLens 100k and 1M, R3-Yahoo and LibraryThing datasets.

dations. Therefore, this parameter should be appropriately tuned on the collection since it is domain-dependent.

This analysis of the cosine-based oracle has improved the previously penalized cosine similarity. However, the possibilities of improving neighborhood techniques are manifold. We think that exploring and characterizing the neighborhoods produced by these oracles regarding other features is a promising avenue for future work.

9.6 RELATED WORK

Traditionally, in collaborative filtering, it has been accepted that the more similar a neighbor is to the target user, the more her preferences can help in the recommendation process. Due to this rationale, much effort has been carried out in the field of neighborhood-based recommenders for improving the similarities and algorithms for clustering users or items. Probably, the most common clustering algorithm consists in using *k*NN with Pearson's correlation as the similarity measure (Resnick et al. 1994). However, many works have addressed the selection of both similarities and clustering algorithms.

Regarding similarities, different measures have been studied in collaborative filtering. As commented before, Pearson's correlation has been the most used. However, other correlations measures such as Spearman's (Ekstrand et al. 2011) or Kendall's (Su and Khoshgoftaar 2009) have also been explored. Studies showed different behaviors when evaluating rating prediction or top-N recommendation. In fact, Pearson's correlation produces effectiveness figures similar to a normalized version of the cosine similarity that operates on user-mean centered ratings (Ekstrand et al. 2011).

Of course, the well-known cosine similarity and adjusted variances of it (Sarwar et al. 2000, 2001) have also been tested for this job. Cremonesi et al. (2010) showed that non-normalized cosine similarity is a better similarity measure than Pearson's correlation for the top-N recommendation task. Other authors also explored other traditional distances in collaborative filtering. Goldberg et al. (2001) used the Euclidean distance for implementing some of their baselines. More recently, Wang et al. claimed that the utilization of the Hamming distance improves the mean average error figures of the k-means algorithm (Wang et al. 2015). In the nineties, Shardanand and Maes (1995) studied the use of mean squared differences and compared the performance of rating prediction with Pearson's correlation showing similar results but lower coverage values. Apart from traditional measures, some works also explored similarity measures that consider the ratings as probability distributions. In previous work, we also proposed the adaptation of term association measures from IR as alternatives to cosine similarity, but these measures did not consistently outperform cosine similarity on diverse datasets (Suárez-García et al. 2018). Deshpande and Karypis (2004) showed how, for item-based recommendations, the use of the normalized conditional probability could obtain small improvements over the normalized version of the cosine. It is important to note that the existing works always coupled the similarity metric with specific scoring methods. Hence, the generalization of the results is limited to the specific scoring formulas (typically a weighted sum).

Another important point to achieve better neighborhoods is the clustering algorithm itself. Regarding these grouping strategies, as previously commented those based on choosing the nearest neighbors are the more popular (Ning et al. 2015). Two main approaches are commonly employed for limiting the number of users to avoid noise: restricting the number of users to k neighbors or use a similarity threshold t and selecting all the users with similarity over t. Additionally, other approaches such as using the k furthest neighbors (Said et al. 2013), selecting k neighbors following a probabilistic distribution (Adamopoulos and Tuzhilin 2014) or building inverted neighborhoods (Vargas and Castells 2014) have been proposed to improve the diversity figures of memory-based collaborative filtering techniques. In previous work, we also compared some of these clustering algorithms with a focus on diversity and novelty metrics (Landin et al. 2018).

The well-known k-means (Xue et al. 2005), or its variants such as bisecting k-means (Sarwar et al. 2002), are probably the second clustering algorithm in popularity. Pitsilis et al. (2011) compared the use of the affinity propagation algorithm with traditional k-means, but, unfortunately, the results regarding mean average error and coverage were worse than previous alternatives. Truong et al. (2007) presented an item-based method for producing uniform clusters of items by minimizing the variance between the items within every cluster. George and Merugu (2005) proposed a weighted co-clustering algorithm that involves simultaneous clustering of users and items; however, the advantages are mainly centered on the efficiency of the algorithm. More recently, other strict partitioning clustering techniques have been tested. Bellogín and Parapar (2012) explored the use of a spectral clustering algorithm, normalized cut, for producing the neighbors showing important improvements regarding precision but lower coverage values than the nearest neighbor approach. Parapar et al. (2013) used posterior probabilistic clustering for the same task obtaining higher coverage values.

It is interesting to note that IDF heuristic was previously used in recommendation. Bellogín et al. (2013a) adapted the most prominent text retrieval models to collaborative filtering. The majority of the text retrieval models exploit the IDF effect. However, the authors used these models for computing directly the list of recommendations. In contrast, in our work, we introduce the IDF heuristic into the neighborhood selection phase.

Finally, some works addressed the goodness of the neighbors directly and how to predict that goodness or adapt the recommendation algorithm to some specific conditions. Herlocker et al. (2002) conducted the first thorough analysis of different choices for similarity metrics, weighting schemes and other collaborative filtering decisions. However, the analysis is limited to the study of the performance of different choices for every step without actually addressing which is the ideal performance at every one of those steps. Another line of research is to address this topic in a similar fashion to the performance prediction and adaptive (or selective) pseudo-relevance feedback in Information Retrieval. Bellogín and Castells (2009) and Bellogín et al. (2014a) adapted different query performance predictors from the text retrieval field to estimate how good is a neighbor. They modified the recommendation formula to weight the influence of every neighbor by its predicted goodness. In a similar line to this adaptive method, Baltrunas and Ricci (2008) presented an improved framework for neighborhood selection where they adjust the similarity between user to the sub-set of co-rated items more related to the target item achieving improvements regarding error metrics.

9.7 CONCLUSIONS

In this chapter, we showed that we are still far away from the maximum performance we can achieve with neighborhood-based recommenders. We designed a greedy strategy for building an oracle that yields nearly optimal neighborhoods in a particular collaborative filtering scenario. This oracle showed that there is a big scope for improvement regarding ranking quality for neighborhood techniques. We proposed an analytical approach to characterize these ground truth neighborhoods. This analysis showed that user profile size plays a key role in clustering users. We proposed a penalized version of cosine similarity inspired in the pivoted document length normalization model from IR. We studied empirically that tuning this normalization term leads to effectiveness gains.

We also presented a simpler oracle that selects the ideal number of neighbors for each user using cosine similarity. The characterization of the neighborhoods produced by this cosine-based oracle led us to introduce the IDF effect in the cosine formulation. Our experiments showed that this variant generates better neighborhoods than the original formulation of cosine similarity. These findings support our analytical approach as a useful approach to analyze and develop new techniques within the field of memory-based recommendation based on the ground truth neighborhoods.

In the next chapter, we propose a different approach to computing neighborhoods inspired in the language modeling framework, a state-of-the-art ad hoc retrieval technique.

10

LANGUAGE MODELS FOR NEIGHBORHOODS

We have shown that we can enhance the effectiveness of neighborhood-based recommenders by improving the neighborhood computation process. In the previous chapter, we characterized ground-truth neighborhoods built from two oracles and proposed improvements in the cosine similarity.

In this chapter, we follow a different approach. We propose to compute neighbors using the language modeling framework. Language models constitute one of the most successful models in information retrieval. Their sound statistical foundation and high ranking effectiveness in several retrieval tasks are key to their current success. Here, we explore how to adapt these language models to address the computation of user and item neighborhoods in a collaborative filtering scenario. Our experiments show that this approach is superior to cosine similarity. In addition to the empirical study, we perform an axiomatic analysis that shows that our proposal satisfies two desirable properties that cosine similarity does not.

We propose to use language models to compute neighborhoods.

We have previously published the contributions included in this chapter. On the one hand, we presented the adaptation of language models to compute neighborhoods (Valcarce et al. 2016b,e). On the other hand, we performed the axiomatic analysis of cosine similarity and language models for neighborhood computation (Valcarce et al. 2017a). We extend here these works by conducting a more exhaustive experimentation.

10.1 INTRODUCTION

Multiple approaches to generate neighborhoods exist in the RS literature because this phase is crucial for neighborhood-based approaches (Ning et al. 2015). The effectiveness of these type of recommenders depends largely on how we calculate the neighborhoods. A popular approach consists in computing the k nearest neighbors according to a pairwise similarity metric.

Ad hoc retrieval and neighborhood computation are analogous tasks.

Previous work has found that the cosine similarity yields better results than Pearson's correlation in terms of accuracy metrics in the neighborhood computation process (Cremonesi et al. 2010). Thinking about cosine similarity in terms of retrieval models, we can note that it is the basic distance measure used in the vector space model, a traditional ad hoc retrieval model (Salton et al. 1975). We can, thus, establish a parallelism between ad hoc retrieval and neighborhood computation. In the case of user-based recommendation, the target user would play the role of the query and the candidate user neighbors the role of documents. In an item-based approach, the target item would act as the query and the candidate item neighbors as documents.

Following this analogy between ad hoc retrieval and neighborhood computation, we can argue the following. If the cosine similarity is a great metric for computing neighborhoods as it is for computing query-document similarities in the VSM, it sounds reasonable to apply more sophisticated ad hoc retrieval models to the task of finding user and item neighborhoods. Therefore, in this chapter, we model the computation of user and item neighborhoods as an ad hoc retrieval task. In particular, we propose an adaptation of language models to compute neighborhoods. Our proposal leverages the advantages of this state-of-the-art retrieval technique to calculate neighborhoods in a collaborative filtering scenario.

10.2 LANGUAGE MODELING OF NEIGHBORHOODS

LM achieve state-of-the-art figures in several IR tasks.

Language models (LM) represent a successful framework in information retrieval. Ponte and Croft (1998) used language models for the first time in the ad hoc retrieval task. Nowadays, language models have become so popular in the field that they have been improved to address several IR tasks achieving state-of-the-art performance (Zhai 2008). Compared to previous techniques, the main contributions of these models are their solid statistical foundation and their interpretability (Ponte and Croft 1998; Zhai 2008).

We adapted the LM framework to the task of finding neighborhoods in a user or item-based manner. In the user-based scenario, we model the generation of ratings by users as a random process given by a probability distribution. In this way, we can see documents and queries as users and terms as items. Thus, the retrieval procedure results in finding the nearest neighbors of the target user (i.e., the query). Analogously, we can flip to the item-based approach. In this case, we model the generation of ratings by items as a random process where the query plays the role of the target item while the rest of items play the role of the documents. In this way, a retrieval returns the most similar items to the target item.

We have already used the aforementioned analogy between retrieval and user-based recommendation in the adaptation of pseudo-relevance feedback techniques to top-N recommendation (see Chapters 5 and 8). Additionally, we used the analogy between item-based recommendation and pseudo-relevance feedback in Chapters 6 and 7.

We adapt the query likelihood model to neighborhood computation.

Within the language modeling framework, we adapt the query likelihood model from Equation (2.3) to the computation of neighborhoods. In the user-based scenario, we estimate the probability of generating the target user u given the language model of the candidate neighbor v as follows:

$$p(u|v) = \prod_{i \in \mathcal{I}_u} p(i|v)^{r(u,i)} \tag{10.1}$$

The item-based counterpart for a target item i and a candidate item neighbor j is analogous:

$$p(i|j) = \prod_{u \in \mathcal{U}_i} p(u|j)^{r(u,i)} \tag{10.2}$$

We compute the conditional probabilities on the right side of Equations (10.1) and (10.2) by smoothing the MLE of a multinomial distribution of ratings. We explore the same smoothing methods we used with relevance models: Jelinek-Mercer (JMS), Dirichlet priors (DPS), absolute discounting (ADS) and additive smoothing (AS). We presented the user-based smoothed estimates in Section 5.2.1. The item-based estimates are analogous as shown in Section 6.3.3.

Language models provide document length normalization and incorporate the IDF effect.

We can use Equations (10.1) and (10.2) as user-based and item-based similarity metrics and plug these equations into kNN algorithm as a replacement of cosine similarity. In the previous chapter, we saw that the document length normalization and the IDF effect could improve the effectiveness of cosine similarity in the neighborhood computation process. Language models also incorporate document length normalization and the IDF effect (Zhai and Lafferty 2004).

10.3 EXPERIMENTS

We run our experiments on four datasets: MovieLens 100k and 1M, R3-Yahoo Music and LibraryThing. In this section, we compare our language modeling approach for computing neighborhoods against cosine similarity. We compute recommendations using the user-based and the item-based versions of WSR (see Section 9.2.2) and the user-based recommender RM2 (see Chapter 5).

10.3.1 *Comparing smoothing methods*

To compare the different methods for smoothing the query likelihood model when computing neighborhoods, we show in Figures 10.1 to 10.4 the values of nDCG@100, Gini@100 and MSI@100 of our proposals on each dataset. We tuned the smoothing parameters $\gamma \in \{0.001, 0.01, 0.1, 1, 10\}$, δ and λ from 0.1 to 1.0 in steps of 0.05, and μ from 400 to 4000 in steps of 200 of the AS, ADS, JMS and DPS methods, respectively. We also compare against cosine similarity as a baseline. In our previous work (Valcarce et al. 2016e), we also compared against RM1Sim (Bellogín et al. 2013b) and Pearson's correlation similarity. For brevity, we omit these baselines since they are weaker than cosine similarity.

Our previous experiments in Section 9.2.3 using cosine similarity with *k*NN algorithm showed that the user-based formulation of the weighted sum recommender (WSR-UB) works better on the MovieLens datasets than the item-based one (WSR-IB). In contrast, on the R3-Yahoo and LibraryThing collections, WSR-IB outperformed WSR-UB. We repeated the experiments with our language modeling approach instead of using cosine similarity obtaining the same trends: user-based approaches work better than item-based on the movie datasets and the other way around on the other two collections. For this reason, we run the user-based query likelihood model on the MovieLens datasets and the item-based version on the other two collections.

The experiments showed that DPS and JMS smoothing methods produced the best results on the MovieLens datasets. This also happens in ad hoc retrieval where DPS and JMS are the favorite methods (Zhai and Lafferty 2004). On R3-Yahoo and LibraryThing, JMS and AS were the best methods. However, only JMS can outperform cosine similarity in ranking accuracy, diversity and novelty on all datasets. Accuracy figures of JMS increase with a high amount of smoothing; however, we can observe a significant drop at $\lambda = 1$ which is expected because the estimate degenerates to the background model. Finally, ADS is not competitive on any dataset. In light of these results, we disregard the rest of methods and, from now on, only use Jelinek-Mercer for smoothing the query likelihood model when computing neighborhoods.

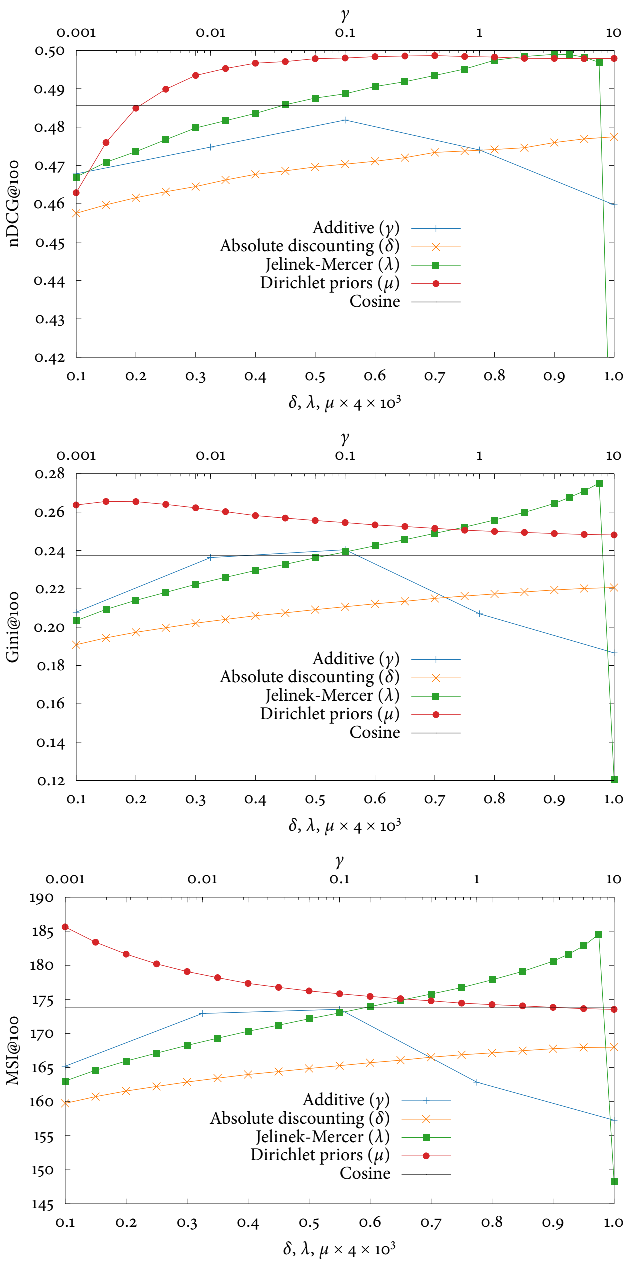


Figure 10.1: Values of nDCG@100 (top), Gini@100 (middle) and MSI@100 (bottom) for WSR-UB on MovieLens 100k varying the smoothing parameters. Neighborhoods are computed taking the 50 closest users according to the query likelihood model smoothed by AS, ADS, JMS and DPS methods as well as cosine similarity.

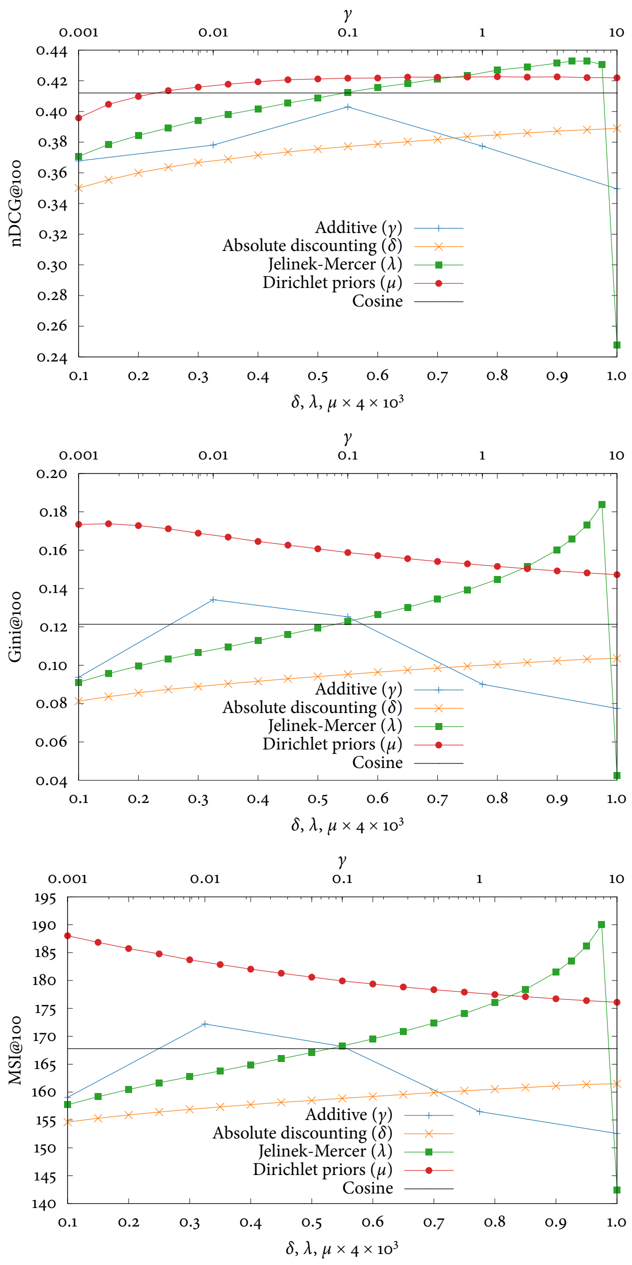


Figure 10.2: Values of nDCG@100 (top), Gini@100 (middle) and MSI@100 (bottom) for WSR-UB on the MovieLens 1M dataset varying the smoothing parameters. Neighborhoods are computed taking the 75 closest users according to the query likelihood model smoothed by AS, ADS, JMS and DPS methods as well as cosine similarity.

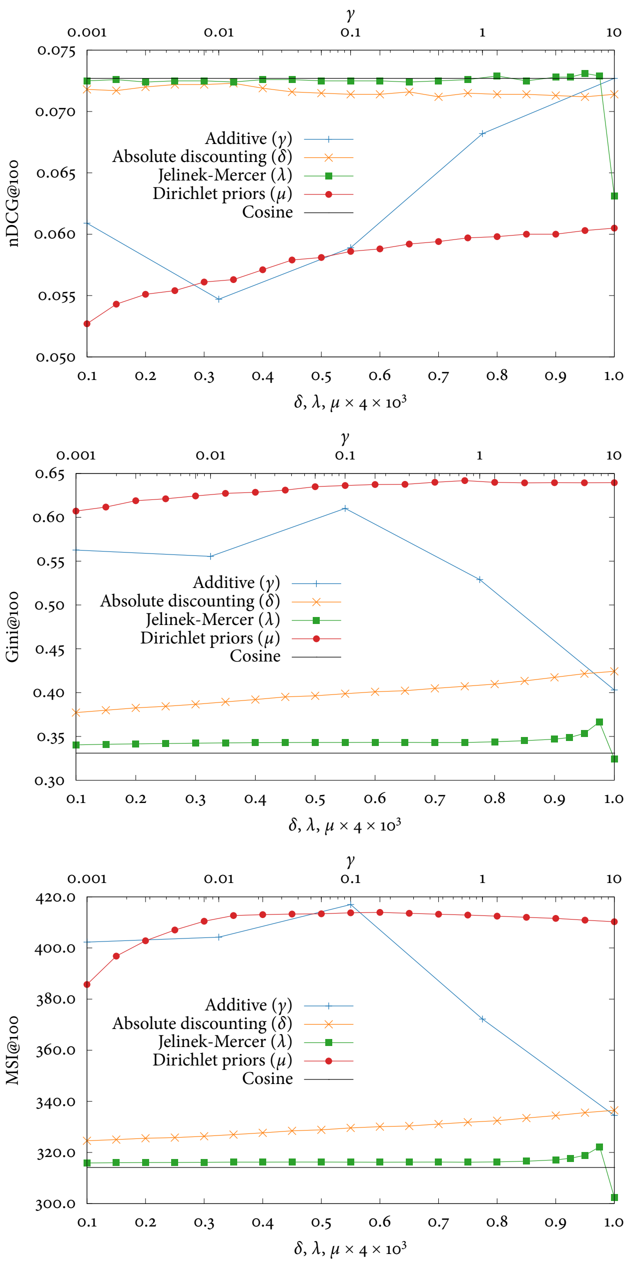


Figure 10.3: Values of nDCG@100 (top), Gini@100 (middle) and MSI@100 (bottom) for WSR-IB on the MovieLens r3 dataset varying the smoothing parameters. Neighborhoods are computed taking the 125 closest users according to the query likelihood model smoothed by AS, ADS, JMS and DPS methods as well as cosine similarity.

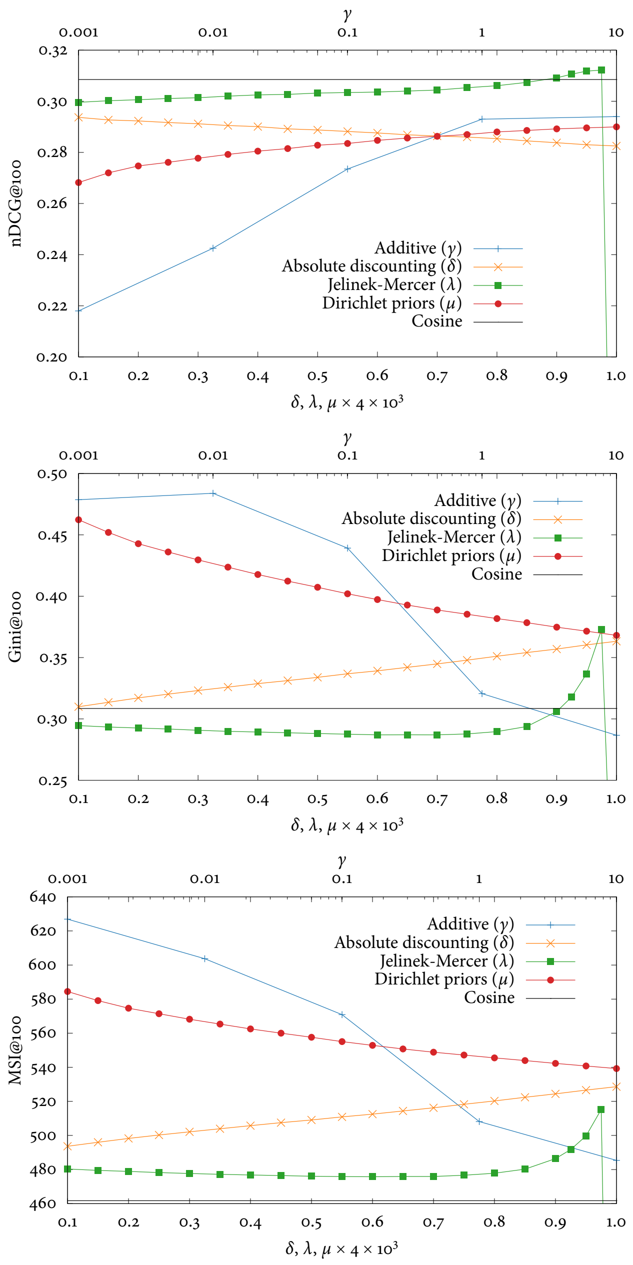


Figure 10.4: Values of nDCG@100 (top), Gini@100 (middle) and MSI@100 (bottom) for WSR-IB on the MovieLens 1M dataset varying the smoothing parameters. Neighborhoods are computed taking the 25 closest users according to the query likelihood model smoothed by AS, ADS, JMS and DPS methods as well as cosine similarity.

10.3.2 *Comparing neighborhood-based methods*

We now compare user-based and item-based WSR and RM2 with cosine similarity and the query likelihood model with Jelinek-Mercer smoothing. Table 10.1 shows the values of nDCG@100, Gini@100 and MSI@100 on the four datasets. For the sake of reproducibility, we show the optimal values of the hyperparameters in Table 10.2.

The language modeling approach improves the ranking accuracy of all the neighborhood-based recommenders compared to cosine similarity in all cases. Additionally, on all datasets except for RM2 on R3-Yahoo, our approach also improves the diversity and novelty figures. In many scenarios, these improvements are statistically significant. This a notable result considering the difficulty of improving these three aspects of the recommendations at the same time.

RM2 using the neighborhoods computed with language models shows the highest figures of ranking accuracy except on the LibraryThing dataset where the item-based version of WSR with the same neighborhoods is better. It is interesting to note that RM2 outperforms WSR-IB on R3-Yahoo although item-based techniques tend to work better than user-based models on this dataset. In general, there is no large difference between item-based and user-based methods in terms of ranking accuracy except on the LibraryThing dataset where WSR-IR significantly outperforms the rest of methods.

Regarding diversity and novelty, the item-based version of WSR using the neighborhoods produced by the query likelihood model obtains the best figures on three out of four datasets. Only on R3-Yahoo, RM2 with cosine similarity can outperform the rest of methods in terms of Gini and Mean Self-Information (MSI). Overall, we can see that item-based models tend to produce more diverse and novel recommendations than user-based recommender systems.

Table 10.2 shows that the optimal number of neighbors vary across datasets, but it is quite stable among methods on the same collection. This indicates that the optimal size of the neighborhoods depends more on the dataset than on the neighborhood computation model. Additionally, regarding the Jelinek-Mercer parameter λ of the query likelihood model, we can see that its optimal value, when combined with RM2, is slightly smaller than when using WSR. Therefore, this hyperparameter needs to be tuned when using different neighborhood-based approaches.

Method	Metric	ML 100k	ML 1M	R3-Yahoo	LibraryThing
Cosine WSR-UB	nDCG	0.4857^{b}	0.4138^{b}	0.0703	0.2255
	Gini	0.2375	0.1356	0.3107	0.0417
	MSI	173.86	172.76	305.26	333.50
Cosine WSR-IB	nDCG	0.4790	0.4035	0.0727^{a}	0.3085^{acdf}
	Gini	0.2738	0.1516	0.3309	0.2768
	MSI	181.59^{ac}	178.95^{a}	314.12^{a}	461.74^{acdf}
Cosine RM2	nDCG	0.4953^{ab}	0.4322^{abe}	0.0717^{a}	0.2384^{a}
	Gini	0.2637	0.1533	**0.4769**	0.1278
	MSI	180.45^{a}	179.39^{a}	**339.64**^{abdef}	417.56^{ad}
LM-JMS WSR-UB	nDCG	0.4990^{abc}	0.4329^{abe}	0.0719^{a}	0.2370^{a}
	Gini	0.2645	0.1731	0.3566	0.0570
	MSI	180.59^{a}	186.15^{abc}	314.23^{a}	352.80^{a}
LM-JMS WSR-IB	nDCG	0.4989^{abc}	0.4232^{ab}	**0.0731**^{a}	**0.3118**^{abcdf}
	Gini	**0.2952**	**0.1854**	0.3520	**0.3368**
	MSI	**190.23**^{abcdf}	**191.34**^{abcdf}	318.00^{abd}	**499.73**^{abcdf}
LM-JMS RM2	nDCG	**0.5021**^{abcd}	**0.4392**^{abcde}	**0.0731**^{acd}	0.2406^{ad}
	Gini	0.2794	0.1825	0.4281	0.1285
	MSI	184.29^{abcd}	189.27^{abcd}	332.49^{abde}	418.39^{ad}

Table 10.1: Values of nDCG@100, Gini@100 and MSI@100 for each recommender approach on the MovieLens 100k and 1M, R3-Yahoo and LibraryThing datasets. Statistically significant improvements in nDCG@100 and MSI@100 according to permutation test ($p < 0.05$) with respect to Cosine-WSR-UB, Cosine-WSR-IB, Cosine-RM2, LM-JMS-WSR-IB, LM-JMS-WSR-IB and LM-JMS-RM2 are superscripted with a, b, c, d, e and f, respectively. Highest value for each metric on each dataset is indicated in bold.

10.4 AXIOMATIC ANALYSIS OF LANGUAGE MODELS FOR NEIGHBORHOODS

To further study why the query likelihood model with Jelinek-Mercer smoothing outperforms cosine similarity, we perform an axiomatic analysis of these two models. Axiomatic analysis has proven to be a useful tool for studying language models formally (Fang et al. 2004; Hazimeh and Zhai 2015; Valcarce et al. 2016a). We enunciate two desirable properties that a good neighborhood technique should satisfy, namely user specificity and item specificity, and perform an axiomatic analysis to verify

Method	ML 100k	ML 1M	R3-Yahoo	LibraryThing
Cosine-WSR-UB	$k = 50$	$k = 50$	$k = 125$	$k = 50$
Cosine-WSR-IB	$k = 75$	$k = 100$	$k = 150$	$k = 25$
Cosine-RM2	$k = 50$	$k = 100$	$k = 150$	$k = 50$
LM-JMS WSR-UB	$k = 50$, $\lambda = 0.90$	$k = 75$, $\lambda = 0.95$	$k = 125$, $\lambda = 0.80$	$k = 50$, $\lambda = 0.95$
LM-JMS WSR-IB	$k = 50$, $\lambda = 0.95$	$k = 50$, $\lambda = 0.95$	$k = 125$, $\lambda = 0.90$	$k = 25$, $\lambda = 0.95$
LM-JMS RM2	$k = 50$, $\lambda = 0.80$	$k = 100$, $\lambda = 0.85$	$k = 150$, $\lambda = 0.50$	$k = 50$, $\lambda = 0.85$

Table 10.2: Optimal number of neighbors k for the kNN algorithm and optimal value of λ for the Jelinek-Mercer smoothing of the language model used in Table 10.1 on the MovieLens 100k and 1M, R3-Yahoo and LibraryThing datasets for the Cosine-WSR-UB, Cosine-WSR-IB, Cosine-RM2, LM-JMS-WSR-IB, LM-JMS-WSR-IB and LM-JMS-RM2 recommenders.

whether the query likelihood model and cosine similarity meet these properties.

10.4.1 *Neighborhood properties*

We propose user and item specificity properties as beneficial recommendation properties.

Next, we present the two proposed properties for neighborhood computation: user specificity and item specificity. We claim that a good neighbor similarity should enforce them. For the sake of simplicity, we use the notation $|u|$ to refer to the sum of ratings of user u: $|u| = \sum_{i \in \mathcal{I}} r(u,i)$. Likewise, $|i|$ denotes the sum of ratings of item i: $|i| = \sum_{u \in \mathcal{U}} r(u,i)$.

We analyze the user-based versions of cosine similarity and the query likelihood model with Jelinek-Mercer smoothing. Similar properties for the item-based formulations could be formulated, and the proofs would be analogous to those presented here.

10.4.1.1 *User specificity*

When calculating the similarity between users to compute a user neighborhood, we often find users with very broad tastes and many ratings. These users are very similar to many other users concerning standard similarity metrics because they have several items rated in common to almost every user. However, these users are not very informative—they like almost everything—and their contribution to the final recommendation is noisy. In other words, we prefer candidate neighbors that have in common with the target user a greater part of their profile. Formally, we can state this property as follows:

Definition. *User specificity*. Given three users u, v and w from the set of users $\mathcal{U}$ such that $\mathcal{I}_u \cap \mathcal{I}_v = \mathcal{I}_u \cap \mathcal{I}_w$, $r(u,i) = r(v,i) = r(w,i)$ for each $i \in \mathcal{I}_u \cap \mathcal{I}_v$ and $|v| < |w|$, the user specificity property enforces $\mathrm{sim}(u,v) > \mathrm{sim}(u,w)$.

10.4.1.2 *Item specificity*

The previous property gives the highest importance to those candidate users whose tastes mainly agree with the target user and avoids those users that have rated many other non-relevant items with respect to the target user. Now we address a different property related to the items that users have in common. If we have two candidate users v and w with the same common ratings with the target user u except for two items j and k which are rated by only one of these users, we prefer the user who has the most specific item. With this property, we seek to give more importance to highly specific items rather than general and popular items. We believe that the former kind of items is more informative than the latter. Additionally, we believe that this property may help to provide more diverse neighborhoods improving the novelty and diversity of recommendations. Formally, we can define our property as follows:

Definition. *Item specificity*. Let u be the target user and v and w be two candidate users from the set of users $\mathcal{U}$ such that $|v| = |w|$ and let j and k be two items from the set of items $\mathcal{I}$ such that $j \in \mathcal{I}_u \cap \mathcal{I}_v$ and $k \in \mathcal{I}_u \cap \mathcal{I}_w$. Given that $(\mathcal{I}_u \cap \mathcal{I}_v) \setminus \{j\} = (\mathcal{I}_u \cap \mathcal{I}_w) \setminus \{k\}$, $r(u,j) = r(v,j) = r(u,k) = r(w,k)$ and $r(u,i) = r(v,i) = r(w,i)$ for each $i \in \mathcal{I}_u \cap \mathcal{I}_v \cap \mathcal{I}_w$, if $|j| < |k|$, then the item specificity property enforces $\mathrm{sim}(u,v) > \mathrm{sim}(u,w)$.

10.4.2 *Theoretical analysis of cosine similarity*

Cosine similarity measures the cosine of the angle between two vectors. The user-based formulation is as follows:

$$\cos(u,v) \frac{\sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} r(u,i)\, r(v,i)}{\sqrt{\sum_{i \in \mathcal{I}_u} r(u,i)^2} \sqrt{\sum_{i \in \mathcal{I}_v} r(v,i)^2}} \tag{10.3}$$

10.4.2.1 *User specificity in cosine similarity*

Assuming the users u, v and w as in the definition of the user specificity property, if the sign of $\cos(u,v) - \cos(u,w)$ is positive, then cosine enforces this property. For simplicity, we refer to the root of the squared sum of the ratings of a user u by $\|u\| = \sqrt{\sum_{i\in\mathcal{I}_u} r(u,i)^2}$.

$$\begin{aligned}
&\cos(u,v) - \cos(u,w) = \\
&\quad = \frac{\sum_{i\in\mathcal{I}_u\cap\mathcal{I}_v} r(u,i)\,r(v,i)}{\|u\|\,\|v\|} - \frac{\sum_{i\in\mathcal{I}_u\cap\mathcal{I}_w} r(u,i)\,r(w,i)}{\|u\|\,\|w\|} \\
&\quad = \frac{\sum_{i\in\mathcal{I}_u\cap\mathcal{I}_v} r(u,i)\,r(v,i)\,(\|w\| - \|v\|)}{\|u\|\,\|v\|\,\|w\|} \\
&\quad > 0 \quad \text{if} \quad \|w\| > \|v\|
\end{aligned} \tag{10.4}$$

We can see that cosine similarity only enforces the user specificity property when $\|w\| > \|v\|$, but this is not guaranteed. We know that $|w| > |v|$, that is, $\sum_{i\in\mathcal{I}_w} r(w,i) > \sum_{i\in\mathcal{I}_v} r(v,i)$, but this does not imply $\sqrt{\sum_{i\in\mathcal{I}_w} r(w,i)^2} > \sqrt{\sum_{i\in\mathcal{I}_v} r(v,i)^2}$. Therefore, in general, we cannot say that cosine enforces this property.

10.4.2.2 *Item specificity in cosine similarity*

Assuming the users u, v and w as in the definition of the item specificity property, if the sign of $\cos(u,v) - \cos(u,w)$ is positive, then cosine enforces this property.

$$\begin{aligned}
&\cos(u,v) - \cos(u,w) = \\
&\quad = \frac{\sum_{i\in\mathcal{I}_u\cap\mathcal{I}_v} r(u,i)\,r(v,i)}{\|u\|\,\|v\|} - \frac{\sum_{i\in\mathcal{I}_u\cap\mathcal{I}_w} r(u,i)\,r(w,i)}{\|u\|\,\|w\|} \\
&\quad = \frac{\sum_{i\in\mathcal{I}_u\cap\mathcal{I}_v\cap\mathcal{I}_w} r(u,i)\,r(v,i)\,(\|w\| - \|v\|)}{\|u\|\,\|v\|\,\|w\|} \\
&\qquad + \frac{r(u,j)r(v,j)}{\|u\|\,\|v\|} - \frac{r(u,k)r(w,k)}{\|u\|\,\|w\|} \\
&\quad > 0 \quad \text{if} \quad \|w\| > \|v\|
\end{aligned} \tag{10.5}$$

Taking into account that $r(u,j) = r(v,j) = r(u,k) = r(w,k)$, we can see that cosine similarity only enforces the item specificity property when $\|w\| > \|v\|$ which is a condition based on the users' ratings and not on the specificity of the items. Therefore, in general, cosine similarity does not enforce this property.

10.4.3 *Theoretical analysis of the query likelihood model*

We take logarithms in the query likelihood model to ease the computation. Applying logarithms to Equation (10.1) and using Jelinek-Mercer smoothing from Equation (5.6), the log-likelihood is given by:

$$\mathrm{ql}_{\mathrm{jms}}(u,v) = \log p(u|v) = \sum_{i \in \mathcal{I}_u} r(u,i) \log\left((1-\lambda)\,\frac{r(v,i)}{|v|} + \lambda\,\frac{|i|}{|\mathcal{C}|}\right) \tag{10.6}$$

The above formula needs to be computed over all the items rated by the target user u. However, we can use a more convenient rank equivalent expression that only needs to be computed over the items rated by both users u and v (Zhai and Lafferty 2004):

$$\log p(u|v) = \sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} r(u,i) \log\left(1 + \frac{(1-\lambda)\,\frac{r(v,i)}{|v|}}{\lambda\,\frac{|i|}{|\mathcal{C}|}}\right) \tag{10.7}$$

10.4.3.1 *User specificity effect in the query likelihood model*

Assuming the users u, v and w as in the definition of the user specificity property, if the sign of $\log p(u|v) - \log p(u|w)$ is positive, then the query likelihood model (using Jelinek-Mercer smoothing) supports this property.

$$\begin{aligned}
&\log p(u|v) - \log p(u|w) = \\
&\quad = \sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} r(u,i) \log\left(1 + \frac{(1-\lambda)\, r(v,i)\, |\mathcal{C}|}{\lambda\, |i|\, |v|}\right) \\
&\quad\quad - \sum_{i \in \mathcal{I}_u \cap \mathcal{I}_w} r(u,i) \log\left(1 + \frac{(1-\lambda)\, r(w,i)\, |\mathcal{C}|}{\lambda\, |i|\, |w|}\right) \\
&\quad = \sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} r(u,i) \left[\log\left(1 + \frac{(1-\lambda)\, r(v,i)\, |\mathcal{C}|}{\lambda\, |i|\, |v|}\right)\right. \\
&\quad\quad \left. - \log\left(1 + \frac{(1-\lambda)\, r(w,i)\, |\mathcal{C}|}{\lambda\, |i|\, |w|}\right)\right] \\
&\quad > 0
\end{aligned} \tag{10.8}$$

We can see that $\log p(u|v) - \log p(u|w) > 0$ because $|v| < |w|$ and $r(v,i) = r(w,i)$ for each $i \in \mathcal{I}_v \cap \mathcal{I}_w$. Therefore, the query likelihood model using Jelinek-Mercer smoothing supports the user specificity effect.

10.4.3.2 *Item specificity effect in the query likelihood model*

Assuming the users u, v and w as in the definition of the item specificity property, if the sign of $\log p(u|v) - \log p(u|w)$ is positive, then the query likelihood model (using Jelinek-Mercer smoothing) supports this property.

$$\begin{aligned}
&\log p(u|v) - \log p(u|w) = \\
&\quad = \sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} r(u,i) \log\left(1 + \frac{(1-\lambda)\, r(v,i)\, |\mathcal{C}|}{\lambda\, |i|\, |v|}\right) \\
&\qquad - \sum_{i \in \mathcal{I}_u \cap \mathcal{I}_w} r(u,i) \log\left(1 + \frac{(1-\lambda)\, r(w,i)\, |\mathcal{C}|}{\lambda\, |i|\, |w|}\right) \\
&\quad = r(u,j) \log\left(1 + \frac{(1-\lambda)\, r(v,j)\, |\mathcal{C}|}{\lambda\, |j|\, |v|}\right) \\
&\qquad - r(u,k) \log\left(1 + \frac{(1-\lambda)\, r(w,k)\, |\mathcal{C}|}{\lambda\, |k|\, |w|}\right) \\
&\quad > 0
\end{aligned} \tag{10.9}$$

First, we get rid of the sums using $(\mathcal{I}_u \cap \mathcal{I}_v) \setminus \{j\} = (\mathcal{I}_u \cap \mathcal{I}_w) \setminus \{k\}$ and $r(u,i) = r(v,i) = r(w,i)$ for each $i \in \mathcal{I}_u \cap \mathcal{I}_v \cap \mathcal{I}_w$. Since $r(u,j) = r(u,k) = r(v,k) = r(w,k)$ and $|v| = |w|$, we can see that the difference is positive when $|j| < |k|$. Therefore, we conclude that the query likelihood model with Jelinek-Mercer smoothing enforces the item specificity property.

10.5 CONCLUSIONS

We presented a novel approach to find user or item neighborhoods based on the query likelihood model of the language modeling framework. This method, combined with neighborhood-based recommenders, such as WSR or RM2, yields highly accurate recommendations. We also proposed two properties (user specificity and item specificity) that may be useful for computing neighborhoods. Our axiomatic analysis of cosine similarity and the query likelihood model showed that the former does not enforce these properties while the latter does.

Moreover, in this chapter, we established a parallelism between ad hoc retrieval and neighborhood computation. This allowed us to adapt the query likelihood model to this recommendation task, but it also enables us to adapt any other retrieval model.

Part V

RECOMMENDER SYSTEMS MODELS FOR PSEUDO-RELEVANCE FEEDBACK

As if there could be true stories: things happen in one way, and we retell them in the opposite way.

— Jean-Paul Sartre,
Nausea

11 LINEAR METHODS FOR PSEUDO-RELEVANCE FEEDBACK

Thus far, we have adapted information retrieval models to different recommendation tasks. In this chapter, we explore the opposite direction: adapting recommendation models to retrieval tasks. In particular, we seek to adapt linear methods that are very effective recommender systems to pseudo-relevance feedback.

We turn the tables by adapting a recommendation model to pseudo-relevance feedback.

We propose two linear methods for pseudo-relevance feedback, one document-based and another term-based, that models the PRF task as a matrix decomposition problem. These factorizations involve the computation of an inter-document or inter-term similarity matrix which is used for expanding the original query. These decompositions can be computed by solving a least squares regression problem with regularization and a non-negativity constraint. We find that the term-based formulation outperforms the state of the art whereas the document-based model constitutes a cost-effective technique.

The pseudo-relevance feedback models presented in this chapter have been previously published. First, we proposed the PRF framework based on linear methods and the term-based approach (Valcarce et al. 2018a). Later, we extended that work by developing the document-based linear method for pseudo relevance feedback (Valcarce et al. 2018b).

11.1 INTRODUCTION

We propose a novel framework for the PRF task which is not based on language models, but in linear methods, which we call LiMe. In particular, we propose two modelings of the PRF task as matrix decomposition problems called DLiMe (document-based linear methods) and TLiMe (term-based linear methods). Zamani et al. (2016) proposed RFMF, the first adaptation of a matrix factorization technique to PRF. They com-

We propose a document-based and a term-based PRF models based on linear methods.

pute a latent factor representation of documents/queries and terms using non-negative matrix factorization. In contrast, in this chapter, we propose a different decomposition that stems from the computation of inter-document or inter-term similarities. Previous work on translation models has exploited this concept of inter-term similarities (Berger and Lafferty 1999; Lafferty and Zhai 2001); however, to the best of our knowledge, no state-of-the-art PRF approach directly leverages inter-document or inter-term similarities. Our matrix formulations enable to compute these similarities that yield within the query and the pseudo-relevant set. We use the information of these relationships between documents or terms to expand the original query.

Since producing a good rank of expansion terms is critical for a successful PRF technique, we claim that modeling inter-term similarities should be a desirable property. Additionally, computing good weights for those expansion terms is a critical factor in the performance of a PRF technique. We also think that modeling the relationship between pseudo-relevant documents can be a faster way to produce expansion terms because the number of documents is much smaller than the number of terms in the pseudo-relevant set. In fact, our experiments show that the computation of inter-term similarities produces high-quality rankings of expansion terms and weights. In contrast, our proposal based on inter-document similarities is computationally very cheap at the expense of slightly worse expansion terms.

Additionally, LiMe is a general framework for PRF that can be plugged on top of any retrieval engine. Although we can use retrieval-dependent features within LiMe framework, we explore here only well-known information retrieval heuristics.

LiMe modeling of the PRF task paves the way for developing multiple PRF algorithms since the proposed formulations of the matrix decompositions can be calculated in various ways. We use a method based on regularized linear least squares regression. We employ a ℓ_2 regularization scheme to avoid overfitting and ℓ_1 regularization to enforce sparsity into the learned inter-document or inter-term similarities. This method provides an automatic feature selection which gives us a more compact solution with the corresponding efficiency gains. The combination of ℓ_1 and ℓ_2 regularization for linear least squares problems is also known as an elastic net regression in statistics (Zou and Hastie 2005). Additionally, we add non-negativity constraints to force the computed similarities to be positive to increase the interpretability of the models.

We thoroughly evaluate DLiMe and TLiMe on five TREC collections. The obtained results show that TLiMe outperforms state-of-the-art baselines regarding several common effectiveness metrics. Moreover, TLiMe achieved high values of robustness compared to the baselines. These findings highlight the applicability of TLiMe as a pseudo-relevance feedback technique. In contrast, DLiMe provides a computationally cheaper alternative with a slight decrease in effectiveness. It is important to note that the LiMe framework can exploit different features allowing the exploration of further features schemes.

11.2 LINEAR METHODS

Linear methods are a simple but successful collection of techniques that have been commonly used for regression and classification tasks, but also in recommender systems. SLIM is a state-of-the-art recommendation model that learns an item-item similarity matrix using linear methods (Ning and Karypis 2011; Ning et al. 2015). The main problem of this recommender system is its scalability because the size of the matrix grows quadratically with the number of items. However, in this scenario, the number of terms tend to be small because typical PRF models use a pseudo-relevant set formed of only a few documents (Li 2008). Moreover, in Section 11.4.4, we show that our proposal also works best using a few pseudo-relevant documents.

We propose a PRF method called LiMe inspired in SLIM. We devise a framework that learns inter-document (DLiMe) or inter-term similarities (TLiMe). Given n features and m data points, $\vec{y} = (y_1, \ldots, y_m)^T$ is the column vector which contains the response and $\vec{x}_1, \ldots, \vec{x}_n$ are the m-dimensional vectors that contain each of the n features of the m observations. A linear method tries to predict the response $\vec{y}$ using a linear combination of $\vec{x}_1, \ldots, \vec{x}_n$. The vectors of features can be arranged in the form of a matrix X of m rows and n columns. Linear regression aims to find the optimal values of the coefficients $\vec{w} = (w_1, \ldots, w_n)^T$ that minimize the error $\vec{\epsilon}$:

$$\vec{y} = X\vec{w} + \vec{\epsilon} = w_1\vec{x}_1 + \cdots + w_n\vec{x}_n + \vec{\epsilon} \tag{11.1}$$

In particular, ordinary linear least squares models try to find the best approximate solution of this system of linear equations where the sum of squared differences between the data and the prediction made by the model serves as the measure of the goodness of the approximation:

$$\vec{w}^* = \arg\min_{\vec{w}} \|\vec{\epsilon}\|_2^2 = \arg\min_{\vec{w}} \|\vec{y} - X\vec{w}\|_2^2 \tag{11.2}$$

Linear least squares loss is strictly convex; thus, it has a unique minimum. Moreover, the simplicity of the model favors its explainability and interpretability. However, this model suffers from overfitting. For tackling this problem, it is common to add ℓ_2 or Tikhonov regularization (this model is also known as ridge regression in Statistics (Hoerl and Kennard 1970)). Imposing a penalty based on the squared ℓ_2-norm of the coefficients $\vec{w}$ produces a shrinking effect which is controlled by the non-negative parameter β_2:

$$\vec{w}^* = \arg\min_{\vec{w}} \|\vec{y} - X\vec{w}\|_2^2 + \beta_2 \|\vec{w}\|_2^2 \tag{11.3}$$

An alternative strategy to ridge regression is imposing a penalty based on the ℓ_1-norm of the coefficient vector. This approach is commonly known as lasso regression in Statistics (Tibshirani 1996). This approach performs automatic feature selection as the value of the non-negative parameter β_1 grows:

$$\vec{w}^* = \arg\min_{\vec{w}} \|\vec{y} - X\vec{w}\|_2^2 + \beta_1 \|\vec{w}\|_1 \tag{11.4}$$

Since both, ridge and lasso regressions, have beneficial properties, Zou and Hastie (2005) developed a technique combining both ℓ_1 and ℓ_2 regularization: the elastic net, which is a generalization of ridge and lasso regression. This approach can perform shrinkage and feature selection at the same time controlled by the non-negative parameters β_1 and β_2:

$$\vec{w}^* = \arg\min_{\vec{w}} \|\vec{y} - X\vec{w}\|_2^2 + \beta_1 \|\vec{w}\|_1 + \beta_2 \|\vec{w}\|_2^2 \tag{11.5}$$

11.3 LIME FRAMEWORK FOR PRF

LiMe jointly models the query and the pseudo-relevant documents.

LiMe is designed for ranking the candidate terms for producing an expanded query Q'. As it is usual in PRF, LiMe uses only information about the original query Q and the pseudo-relevant set F. The set F is composed of the top-k documents retrieved using the original query Q. We should note that LiMe treats the query as another document. Thus, for convenience, we define the extended feedback set F' as the pseudo-relevant set plus the original query ($F' = \{Q\} \cup F$) and denote its cardinality by $m = |F'| = k + 1$. We consider as candidate terms the subset of words from the collection vocabulary $\mathcal{V}$ that appear in F'. We refer to this set by $\mathcal{V}_{F'}$ and we denote its cardinality by $n = |\mathcal{V}_{F'}|$.

11.3.1 *LiMe formulations*

We can define LiMe using a matrix or a vector formulation. To understand better the idea behind LiMe, we initially present the matrix formulation of our technique. Later, we introduce the vector representation which allows us to optimize the implementation.

The matrix formulation is more intuitive, the vector formulation is more convenient to implement.

Considering the query as another pseudo-relevant document, we define the matrix $X = (x_{ij}) \in \mathbb{R}^{m \times n}$. The first row represents the original query Q while the rest rows correspond to the k documents from F. Each column of X corresponds to a term from $\mathcal{V}_{F'}$. Each element x_{ij} represents a feature between the document (or query) corresponding to the i-th position and the term t_j represented with the j-th column of X. Therefore, each row of X is a sparse feature vector representing the query or a pseudo-relevant document.

The objective of LiMe is to factorize this matrix X into the product of itself and another matrix. In the case of TLiMe, we build an inter-term matrix $W = (w_{ij}) \in \mathbb{R}_{+}^{n \times n}$ whereas, in the case of DLiMe, we build an inter-document matrix $Z = (z_{ij}) \in \mathbb{R}_{+}^{m \times m}$.

11.3.1.1 *Term-based approach*

In our term-based approach (TLiMe), the matrix W represents the inter-term similarity between pairs of words in $\mathcal{V}_{F'}$. In particular, each entry w_{ij} symbolizes the similarity between terms t_i and t_j. To increase the interpretability of the model, we constrain the similarities to be non-negative. Moreover, to avoid the trivial solution (W equal to the identity matrix), we enforce that the main diagonal of W is all zeros. Formally, we define TLiMe as an algorithm that computes the following decomposition:

TLiMe models the similarity between every pair of terms.

$$\begin{aligned} & X \approx X\,W \\ s.t.\ & \operatorname{diag}(W) = 0,\ W \geq 0 \end{aligned} \tag{11.6}$$

We formulate this matrix decomposition task as a constrained linear least squares optimization problem. We want to minimize the residual sum of squares of the factorization. Additionally, to avoid overfitting and to enforce a sparse solution we apply the elastic net penalty which combines ℓ_1 and ℓ_2 regularization. In this way, the objective function of LiMe is the following one:

$$\begin{aligned} W^* \ = \ & \underset{W}{\arg\min} \quad \frac{1}{2}\left\|X - X\,W\right\|_F^2 + \beta_1 \left\|W\right\|_{1,1} + \frac{\beta_2}{2}\left\|W\right\|_F^2 \\ & \text{s.t.} \qquad \operatorname{diag}(W) = 0,\ W \geq 0 \end{aligned} \tag{11.7}$$

Note that the matrix $\ell_{1,1}$-norm (denoted by $\|\cdot\|_{1,1}$) is equivalent to the sum of the ℓ_1-norm of the columns. On the other hand, the squared Frobenius norm (denoted by $\|\cdot\|_F^2$) is calculated as the sum of the squares of each matrix element which is equivalent to the sum of the squared ℓ_2-norm of the columns. Using these equivalences between the matrix and vector norms, we can split this matrix formulation by columns rewriting the optimization problem in the following vector form:

$$\begin{aligned} \vec{w}_{\cdot j}^* \; = \; & \underset{\vec{w}_{\cdot j}}{\arg\min} \quad \frac{1}{2} \left\| \vec{x}_{\cdot j} - X \vec{w}_{\cdot j} \right\|_2^2 + \beta_1 \left\| \vec{w}_{\cdot j} \right\|_1 + \frac{\beta_2}{2} \left\| \vec{w}_{\cdot j} \right\|_2^2 \\ & \text{s.t.} \quad\quad w_{jj} = 0, \; \vec{w}_{\cdot j} \geq 0 \end{aligned} \tag{11.8}$$

where the non-negativity constraint is applied to the elements of $\vec{w}_{\cdot j}$ vector which is the j-th column of the W matrix. Similarly, $\vec{x}_{\cdot j}$ represents the j-th column of the X matrix. For each term j in $\mathcal{V}_{F'}$, we train an elastic net (Zou and Hastie 2005) with an equality constraint to zero in one coefficient and non-negativity constraints on the rest of the coefficients.

We merge the solutions of the regression problems depicted in Equation (11.8) to build the inter-term similarity matrix W^*. We use the computed matrix decomposition to reconstruct the first row of X (which we will denote by $\hat{x}_{1\cdot}$) as follows:

$$\hat{x}_{1\cdot} = \vec{x}_{1\cdot} W^* \tag{11.9}$$

Note that, by construction, X is a sparse matrix (hence also the row vector $\vec{x}_{1\cdot}$) and W^* will be a sparse matrix due to the ℓ_1 regularization. Thus, the product between the row vector $\vec{x}_{1\cdot}$ and the matrix W^* is highly efficient. We use the pseudo-relevant documents for learning the inter-term similarities, but we reconstruct the first row of X because we want to expand only the query.

11.3.1.2 *Document-based approach*

DLiMe models the similarity between pseudo-relevant documents (including the query).

DLiMe, the document-based linear method, computes the matrix $Z = (z_{ij}) \in \mathbb{R}_+^{m \times m}$. This matrix represents the inter-document similarity between pairs of elements from the extended pseudo-relevant set F' (i. e., the query and the pseudo-relevant documents). The matrix formulation of DLiMe is analogous to TLiMe:

$$\begin{aligned} X &\approx Z X \\ s.t.\, \operatorname{diag}(Z) &= 0, \, Z \geq 0 \end{aligned} \tag{11.10}$$

We also constrain Z to be non-negative to foster interpretability and enforce the diagonal to be zero to avoid the trivial solution. Since we are only interested in reconstructing the first row of X, we only need to compute the first row of Z. Therefore, DLiMe factorization can be reduced to a single constrained linear least squares optimization problem as follows:

$$\begin{aligned} \vec{z}^{*}_{1\cdot} = & \operatorname*{arg\,min}_{\vec{z}_{1\cdot}} && \frac{1}{2}\left\|\vec{z}_{1\cdot} - \vec{z}_{1\cdot}X\right\|^2_2 + \beta_1 \left\|\vec{z}_{1\cdot}\right\|_1 + \frac{\beta_2}{2}\left\|\vec{z}_{1\cdot}\right\|^2_2 \\ & \text{s.t.} && z_{11} = 0,\ \vec{z}_{1i} \geq 0 \end{aligned} \tag{11.11}$$

Note that compared to TLiMe, where n least squares problems have to be solved, DLiMe is much more efficient because it only involves solving one least squares problem. To reconstruct the first row of X we simply need to perform the following vector-matrix multiplication:

$$\hat{x}_{1\cdot} = \vec{z}^{*}_{1\cdot}X \tag{11.12}$$

11.3.2 *LiMe feedback model*

LiMe feedback model is created from $\hat{x}_{1\cdot}$, which can be reconstructed using either DLiMe or TLiMe. We can normalize this vector to obtain a probability estimate. In this way, the probability of the j-th term given the feedback model is given by:

$$p(t_j|\theta_F) = \begin{cases} \dfrac{\hat{x}_{1j}}{\sum_{t_v \in \mathcal{V}_{F'}} \hat{x}_{1v}} & \text{if } t_j \in \mathcal{V}_{F'}, \\ 0 & \text{otherwise} \end{cases} \tag{11.13}$$

We only rank those terms that appear in the pseudo-relevant set or the query. Although some PRF techniques can rank all the terms in the collection, in practice, it is common to only rank those appearing in the pseudo-relevant set or the query (Lavrenko and Croft 2001; Zamani et al. 2016). In fact, scoring terms that do not appear in F' would contradict the foundations of PRF since this approach is based on local information (i. e., the pseudo-relevant set and the query).

Although both LiMe and RFMF decomposes a similar matrix, they use different objective functions and optimization algorithms. Additionally, LiMe employs elastic net regularization. In contrast, RFMF is based on non-negative matrix factorization which can deal with non-negative and sparse data while LiMe deals with this data by enforcing non-negativity constraints in the optimization problem. Additionally, LiMe discovers inter-document (DLiMe) or inter-term similarities (TLiMe) that yield within the pseudo-relevant set and the query while RFMF learns document and term latent factor representations.

Next, we discuss how we fill the matrix $X = (x_{ij})$ with features relating query/documents i with terms j.

11.3.3 *Feature schemes*

One advantage of LiMe is its flexibility: we can use any feature scheme to build matrix X. To foster sparsity in the matrix X, we decided to fill with zeros all those entries that correspond to terms that do not appear in the current document. This approach will provide a quite sparse matrix which can be more efficiently decomposed than a dense one.

Let $s(w, D)$ be the function that assigns a score to the term w given the document D and let $f(w, D)$ be the frequency of occurrence of the term w in the document D, the matrix X is filled in the following manner:

$$x_{ij} = \begin{cases} s(w_j, Q) & \text{if } i = 1 \text{ and } f(w_j, Q) > 0, \\ s(w_j, D_{i-1}) & \text{if } i > 1 \text{ and } f(w_j, D_{i-1}) > 0, \\ 0 & \text{otherwise} \end{cases} \tag{11.14}$$

It is remarkable that TF-IDF heuristic is still valid today in many IR tasks.

We explored several strategies based on well-known weighting functions used in information retrieval. We studied several term frequency measures: raw frequency counts, binarized counts and logarithmic versions. Additionally, we tried different TF-IDF formulations. We achieved the best results using the following TF-IDF weighting function proposed by Salton (Salton 1971):

$$s_{tf\text{-}idf}(w, D) = (1 + \log_2 f(w, D)) \times \log_2 \frac{|\mathcal{C}|}{df(w)} \tag{11.15}$$

where $|\mathcal{C}|$ is the number of documents in the collection and $df(w)$ represents the document frequency of the term w (i. e., the number of documents in the collection where the term w occurs).

In any case, other alternatives may be possible. In fact, in previous work, we also reported the performance for the logarithmic TF heuristic (Valcarce et al. 2018e). Additionally, it may be worth exploring features related to the first retrieval such as the contribution of an individual term to the document score within a particular retrieval model; however, in that case, LiMe would not be independent of the retrieval technique. Also, we could derive probabilistic weighting functions (as RFMF does) at the expense of introducing a few new parameters to tune into the model. We leave for future work the investigation of additional features schemes. Nevertheless, the ability of LiMe for performing well with simple and well-known features such as TF-IDF is remarkable. Also, this weighting function is supported by decades of research in information retrieval.

11.3.4 *Implementation details*

Equation (11.8) shows that the computation of matrix W^* can be divided into multiple linear regression problems, one for each vector $\vec{w}^*_{\cdot j}$ which represents a term in $\mathcal{V}_{F'}$. Thus, each column of the matrix W^* can be computed separately and, if needed, in parallel without any dependencies among them. In contrast, DLiMe only requires to solve one least squares problem as it can be seen in Equation (11.11). To solve these regression problems, we used the highly efficient BCLS[1] library, which implements a two-metric projected-descent method for solving bound-constrained least squares problems.

An additional optimization for TLiMe is to drop part of the matrix W^*. This matrix is used for computing expansion terms when multiplied by the vector $\vec{x}_{1\cdot}$ as shown in Equation (11.9). Therefore, we only need those rows that correspond to a term in the original query. If we only store those similarities, we save much space since the number of terms in a query prompted by a user is tiny compared to the number of rows.

11.4 EXPERIMENTS

In this section, we assess the performance of LiMe against state-of-the-art techniques. The experiments are performed using Terrier (Macdonald et al. 2012) on five TREC collections commonly used in PRF literature (Lv and Zhai 2009, 2014; Zamani et al. 2016): AP88-89, TREC-678, Robust-04, WT10G and GOV2 (details are shown in Table 3.1). We apply training and test evaluation on all collections. We optimize the model hyperparameters to maximize MAP using the training topics, and we use the test topics to evaluate the performance of the methods.

We produce a rank of 1000 documents per query. We evaluate MAP and nDCG using `trec_eval`[2] at a cut-off of 1000. Additionally, we measure the reliability of improvement (RI). We employ the permutation test at a significance level of 0.05 to measure if the improvements regarding MAP and nDCG are statistically significant. We cannot apply a paired statistic to RI because it is a global metric.

1 The Bound-Constrained Least Squares library is available at: https://www.cs.ubc.ca/~mpf/bcls.

2 The `trec_eval` is available at: https://trec.nist.gov/trec_eval.

We use title queries from TREC topics. We preprocess the collections with the standard Terrier stopwords removal and Porter stemmer since previous work recommended the use of stemming and stopwords removal (Lv and Zhai 2009).

11.4.1 *Baselines*

We use the state-of-the-art language modeling framework for performing the first and second stage retrievals (Ponte and Croft 1998). In particular, we used the Kullback-Leibler divergence retrieval model presented in Equation (2.6) which allows us to introduce a feedback model (Lafferty and Zhai 2001). For smoothing the document language models, we used Dirichlet priors smoothing with the parameter $\mu = 1000$ (Zhai and Lafferty 2004). To compare the effectiveness of our proposals, we use the following state-of-the-art baselines. All the feedback models produced by these PRF baselines are interpolated with the original query as shown in Equation (2.7). Finally, they interpolate the feedback model with the original query model.

11.4.1.1 *Language models (LM)*

First, we should always compare a PRF technique against the performance of a retrieval model without feedback information. We used the query likelihood retrieval model (LM) with Dirichlet priors smoothing ($\mu = 1000$) (Ponte and Croft 1998; Zhai 2008; Zhai and Lafferty 2004).

11.4.1.2 *Relevance feedback matrix factorization (RFMF)*

RFMF was the first technique that applied matrix factorization to the PRF task (Zamani et al. 2016). This approach builds a document-term matrix X from the query and the pseudo-relevant set. They built this matrix using TF-IDF or weights derived from the language modeling framework. RFMF reconstructs, through non-negative matrix factorization (NMF), the document-term matrix and use the new weights as a scoring function to rank candidates terms for expansion. This approach is inspired by the recommender systems literature where matrix factorization techniques are commonplace (Koren and Bell 2015). To have a fair comparison, we build the document-term matrix using the same TF-IDF weights we use in LiMe.

Formally, NMF is a matrix factorization algorithm which decomposes the matrix $X \in \mathbb{R}_+^{m \times n}$ in two matrices $U \in \mathbb{R}_+^{m \times d}$ and $V \in \mathbb{R}_+^{d \times n}$ such that

$X \approx UV$. U represents the latent factors of the query and the pseudo-relevant documents whereas V represents the latent factors of the terms.

11.4.1.3 *Maximum-entropy divergence minimization model (MEDMM)*

The maximum-entropy divergence minimization model (Lv and Zhai 2014) is a PRF technique based on the divergence minimization model (DMM) which stems from the language modeling framework (Zhai and Lafferty 2001). It is similar to the Rocchio algorithm from the vector space model if we use the pseudo-relevant set to compute the relevant documents vectors and the collection model for the non-relevant documents vectors (Rocchio 1971). MEDMM aims to find a feedback model θ_F which minimizes the distance to the language models of the documents of the pseudo-relevant set and, at the same time, maximizes the distance to the collection model θ_C (the assumed non-relevant model). This model has a parameter λ to control the IDF effect and a parameter β to regulate the entropy of the feedback language model:

$$\theta_F = \arg\min_{\theta} \sum_{D \in F} \alpha_D \, H(\theta, \theta_D) - \lambda \, H(\theta, \theta_C) - \beta \, H(\theta) \tag{11.16}$$

where $H(\cdot,\cdot)$ denotes the cross entropy and $H(\cdot)$ denotes the entropy.

MEDMM also gives a weight α_D for each document based on the posterior of the document language model:

$$\alpha_D = p(\theta_D|Q) = \frac{p(Q|\theta_D)}{\sum_{D' \in F} p(Q|\theta'_D)} = \frac{\prod_{t \in Q} p(t|\theta_D)}{\sum_{D' \in F} \prod_{t' \in Q} p(t'|\theta'_D)} \tag{11.17}$$

The analytic solution to MEDMM, obtained with Lagrange multipliers, is given by (Lv and Zhai 2014):

$$p(t|\theta_F) \propto \exp\left(\frac{1}{\beta} \sum_{D \in F} \alpha_D \log p(t|\theta_D) - \frac{\lambda}{\beta} \log p(t|\theta_C)\right) \tag{11.18}$$

where $p(t|\theta_D)$ is the smoothed MLE of the term t under the language model θ_D using additive smoothing with the parameter γ. On the other hand, $p(t|\theta_C)$ represents the MLE of the term t in the collection.

11.4.1.4 *Relevance models (RM)*

Relevance-based language models or, for short, Relevance Models (RM) are a state-of-the-art PRF technique that explicitly introduces the concept of relevance in language models (Lavrenko and Croft 2001). There are two models for estimating the relevance: RM1 (which uses i.i.d. sampling) and RM2 (based on conditional sampling); however, RM1 model has shown to be more effective than RM2 (Lv and Zhai 2009). RM1 estimates can be computed as follows when assuming uniform document prior probabilities:

$$p(t|\theta_F) \propto \sum_{D \in F} p(t|\theta_D) \prod_{q \in Q} p(q|\theta_D) \tag{11.19}$$

where $p(t|\theta_D)$ is the smoothed maximum likelihood estimate (MLE) of the term t under the language model of the document D with Dirichlet priors as the preferred smoothing technique (Lavrenko and Croft 2001; Zhai and Lafferty 2004). RM1 is typically called RM3 when it is interpolated with the original query (see Equation (2.7)) (Abdul-Jaleel et al. 2004). We set the Dirichlet priors smoothing parameter μ' to 1000 as it is typically done (Lv and Zhai 2009, 2014; Zamani et al. 2016).

For all the PRF models, we swept the number of top k documents retrieved in the first stage among $\{5, 10, 25, 50, 75, 100\}$ and the number of expansion terms e among $\{5, 10, 25, 50, 75, 100\}$. We swept the query interpolation parameter α from 0 to 1 in steps of 0.1. Regarding LiMe, we trained the β_1 and β_2 parameters. We tuned the values of β_1 among $\{0.01, 0.1, 1.0\}$ and parameter β_2 among $\{10, 25, 50, 100, 150, 200, 250, 300, 350, 400, 450\}$. We select those parameters that maximize the values of MAP in the training set.

11.4.2 *Effectiveness analysis*

The results of the experiments regarding MAP, nDCG, and RI are summarized in Table 11.1. Overall, all the PRF techniques outperform the language modeling baseline without query expansion. However, TLiMe is the only method that offered significant improvements over LM in MAP and nDCG on all collections. DLiMe showed competitive effectiveness concerning MEDMM and RM3.

To further analyze if PRF techniques are beneficial, we measured the robustness index. This value is positive for all the methods on every collection. This value means that, on average, more queries were improved rather than worsened due to the PRF techniques. Either DLiMe or TLiMe achieved the highest figures in RI on every dataset except for MEDMM on the WT10G collection. Additionally, RM3 achieve the same robustness index as TLiMe does on the Robust-04 collection.

On all datasets, TLiMe achieved the highest results regarding MAP and nDCG. No baseline outperformed TLiMe on any dataset. TLiMe significantly surpassed RFMF on four out of five datasets regarding MAP

Method	Metric	AP88-89	TREC-678	Robust-04	WT10G	GOV2
	MAP	0.2349	0.1931	0.2914	0.2194	0.3310
LM	nDCG	0.5637	0.4518	0.5830	0.5212	0.6325
	RI	–	–	–	–	–
	MAP	0.2774^{a}	0.2072	0.3130^{a}	0.2389^{a}	0.3580^{a}
RFMF	nDCG	0.5749^{a}	0.4746	0.5884	0.5262	0.6453
	RI	0.42	0.23	0.07	0.30	0.42
	MAP	0.3010^{ab}	0.2327^{abde}	0.3447^{ab}	0.2472^{a}	0.3790^{ab}
MEDMM	nDCG	0.5955^{ab}	0.5115^{abde}	0.6227^{ab}	0.5324	0.6653^{ab}
	RI	0.42	0.26	0.32	**0.36**	0.66
	MAP	0.3002^{ab}	0.2235^{ab}	0.3488^{ab}	0.2470^{a}	0.3755^{ab}
RM3	nDCG	0.6005^{ab}	0.4987^{ab}	0.6251^{ab}	0.5352	0.6618^{ab}
	RI	0.50	0.40	**0.37**	0.20	0.60
	MAP	0.3112^{ab}	0.2206^{ab}	0.3435^{ab}	0.2368^{a}	0.3731^{ab}
DLiMe	nDCG	0.6058^{ab}	0.4936^{ab}	0.6247^{ab}	0.5290	0.6588^{ab}
	RI	**0.52**	0.44	0.32	0.26	**0.72**
	MAP	**0.3149**^{abc}	**0.2357**^{abd}	**0.3517**^{a}	**0.2476**	**0.3830**^{a}
TLiMe	nDCG	**0.6085**^{a}	**0.5198**^{abd}	**0.6294**^{a}	**0.5398**	**0.6698**^{ab}
	RI	**0.52**	0.46	**0.37**	0.30	0.62

Table 11.1: Values of MAP, nDCG and RI for each technique on AP88-89, TREC-678, Robust-04, WT10G and GOV2 datasets. Statistically significant improvements according to permutation test ($p < 0.05$) with respect to LM, RFMF, MEDMM, RM3, DLiMe and TLiMe are superscripted with a, b, c, d, e and f, respectively. Highest value of each metric for each dataset is indicated in bold.

and nDCG. Regarding RM3, TLiMe significantly outperformed RM3 on three collections (concerning MAP or nDCG). The strongest baseline, MEDMM, was only significantly surpassed by TLiMe on the AP88-89 collection. However, on all datasets, TLiMe showed higher figures of nDCG and MAP than MEDMM. Although no baseline significantly outperformed TLiMe, MEDMM significantly surpassed RM3 and DLiMe regarding nDCG and MAP on the TREC-678 collection. Also, DLiMe, RM3, and MEDMM significantly improved RFMF in terms of MAP and nDCG on several datasets.

It is interesting to remark that the PRF techniques achieved the smallest improvements in the WT10G collection. This small improvement is probably due to the nature of the web which is a noisy media. Also, the values of RI on this dataset are the lowest.

Regarding the differences between DLiMe and TLiMe, the latter approach showed better figures of MAP and nDCG on all datasets. Nevertheless, the differences are significant only on the TREC-678 collections. In contrast, DLiMe provided higher RI than TLiMe on GOV2 and the same figure on AP88-89 collections.

11.4.3 *Query analysis*

We perform a qualitative analysis of the query expansion methods.

To provide insights into the good results achieved by DLiMe and TLiMe, we manually studied the expanded queries produced by the tested PRF methods. As a representative example, Table 11.2 shows the top 10 expansion terms for the TREC topic 664 ("American Indian Museum") on the Robust-04 collection.

RM3 provided bad expansion terms by adding very common uninformative terms such as "will", "1" or "new". Those terms seem to be a problem of low IDF effect. In contrast, MEDMM yielded much better expansion terms. However, some of them are of dubious utility such as "live" or "part". RFMF provided specific terms, but some of them are completely unrelated to the topic (e. g., "dolphin" or "rafaela"). Hence, the inferior performance of RFMF is likely to be due to the introduction of noisy terms. Regarding our methods, we can see than DLiMe provided good expansion terms. Still, this approach included the term "hey" which we think is uninformative. In this case, TLiMe yielded the best expansion terms. All of them are specific and related to the topic.

In the light of the results, we can claim that RM3 and MEDMM tend to foster those terms that appear in the majority of the pseudo-relevant set in contrast to matrix factorization approaches. LiMe was capable of selecting very specific and relevant terms such as "smithsonian" or "chumash". RFMF was also able to include relevant terms such as "professor" but it also added non-related terms. Therefore, the main advantage of the matrix formulation is its ability to select discriminative words without being biased to popular and non-informative terms in the pseudo-relevant set. However, our approach based on inter-term or inter-doc similarities can select relevant terms while RFMF factorization approach based on document and term latent factors is incapable of filtering non-related terms.

(a) RFMF.

TERM	WEIGHT
indian	0.1725
museum	0.1685
american	0.1505
professor	0.0193
tribal	0.0160
ancient	0.0155
dolphin	0.0153
rafaela	0.0140
activist	0.0137
racist	0.0137

(b) MEDMM.

TERM	WEIGHT
indian	0.1511
museum	0.0802
american	0.0780
cultur	0.0210
year	0.0177
live	0.0153
nation	0.0148
artifact	0.0146
part	0.0139
tribal	0.0127

(c) RM3.

TERM	WEIGHT
indian	0.1285
american	0.0895
museum	0.0874
year	0.0219
will	0.0209
west	0.0182
1	0.0167
tribal	0.0158
time	0.0149
new	0.0147

(d) DLiMe.

TERM	WEIGHT
indian	0.1392
museum	0.1365
american	0.1257
smithsonian	0.0394
artifact	0.0307
hey	0.0272
tribal	0.0271
cultur	0.0250
chumash	0.0219
tribe	0.0213

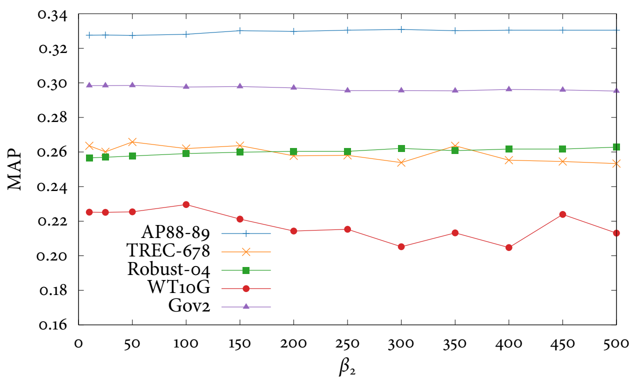
(e) TLiMe.

TERM	WEIGHT
indian	0.1392
museum	0.1364
american	0.1256
tribe	0.0393
artifact	0.0306
cultur	0.0272
tribal	0.0271
nation	0.0249
chumash	0.0219
smithsonian	0.0212

Table 11.2: Top 10 expansion terms for the TREC topic 664 (“American Indian Museum”) when using the different PRF methods on the Robust-04 collection.

11.4.4 *Sensitivity analysis*

Regarding the parameters of LiMe, we observed that the differences in effectiveness between DLiMe and TLiMe when we changed the value of β_1 were minor. Thus, we can set β_1 to 0.01 reducing the number of parameters to tune and obtaining good results. Nevertheless, the inclusion of ℓ_1 regularization into LiMe models is still beneficial since it provides sparsity to the learned matrix W with the corresponding space savings. Regarding β_2, we plot the values of MAP achieved by DLiMe and TLiMe with different amount of ℓ_2 regularization in Figure 11.1. Except for the



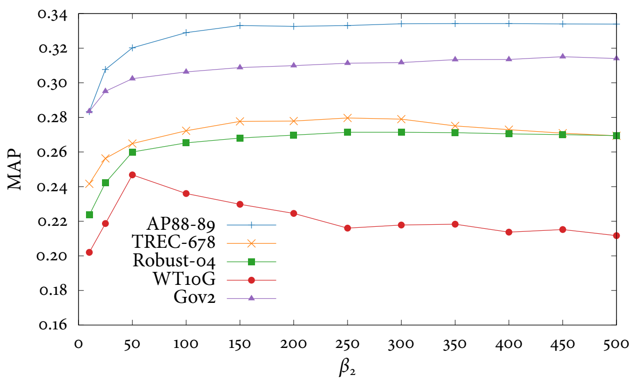


Figure 11.1: Sensitivity of DLiMe (top) and TLiMe (bottom) techniques to β_2 on each collection. The rest of the parameters were fixed to their optimal values.

WT10G collection, the parameter β_2 is relatively stable among the values 150 and 400 for both DLiMe and TLiMe.

We also studied how DLiMe and TLiMe behave varying the size of the pseudo-relevant set k, the number of expansion terms e and the interpolation parameter α. Figures 11.2 and 11.3 present the results of the sensitivity analysis regarding MAP. The sensitivity analysis of the baselines (RFMF, MEDMM and RM3) is reported in our previous work (Valcarce et al. 2018a,b) and is omitted here for the sake of brevity.

The general trend is that a high number of pseudo-relevant documents hurts the performance of the PRF techniques. LiMe is quite stable and behaves optimally with 5-10 documents. LiMe methods are robust to noisy collections and work well with a high number of terms on WT10G. Regarding the interpolation parameter α, except for the GOV2 collection, we observed that the optimal values for DLiMe and TLiMe lie within a

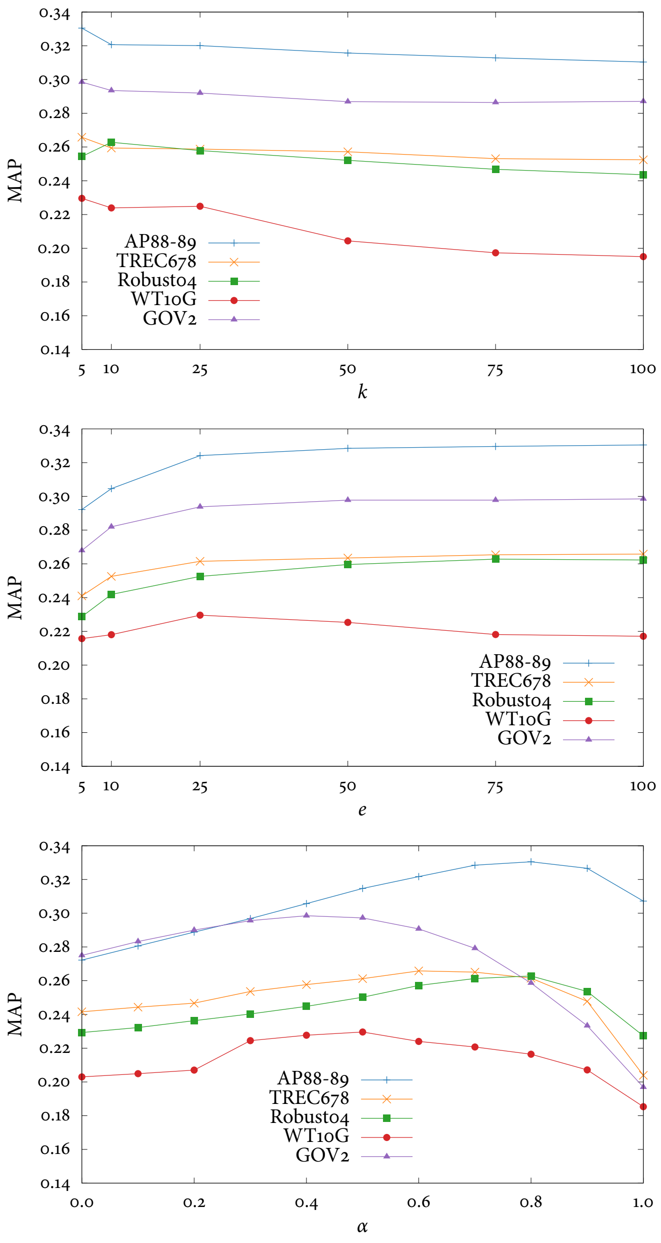


Figure 11.2: Sensitivity of DLiMe to the number of feedback documents (top), the number of expansion terms (middle) and the interpolation parameter of the original query with the expansion terms (bottom) on each collection. The rest of the parameters were fixed to their optimal values.

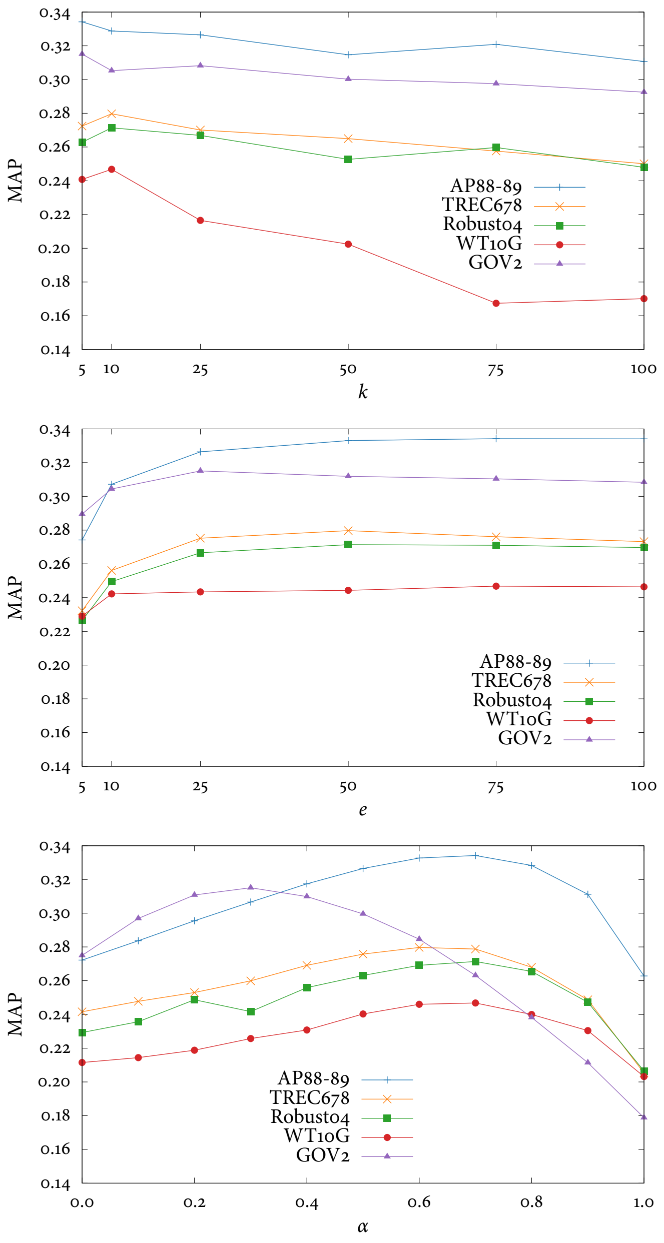


Figure 11.3: Sensitivity of TLiMe to the number of feedback documents (top), the number of expansion terms (middle) and the interpolation parameter of the original query with the expansion terms (bottom) on each collection. The rest of the parameters were fixed to their optimal values.

narrow interval. Nevertheless, we can see that α has a notable impact on any PRF technique and should be tuned adequately.

11.5 RELATED WORK

Pseudo-relevance feedback is a fertile area of research in information retrieval (Carpineto et al. 2001; Croft and Harper 1979; Lavrenko and Croft 2001; Lv and Zhai 2009, 2014; Rocchio 1971; Zamani et al. 2016). Among the PRF techniques, those based on the language modeling framework have showed great effectiveness (Lv and Zhai 2009). Therefore, we used them as baselines and described them in Section 11.2. Additionally, we included RFMF as a baseline because it was the first work that modeled the PRF task as a matrix factorization problem (Zamani et al. 2016).

PRF methods have been adapted to collaborative filtering recommendation with great success (Parapar et al. 2013). In particular, relevance models (see Chapters 5 to 7) and techniques used within the Rocchio framework (see Chapter 8). Conversely, RFMF is a case of a recommendation technique applied to PRF (Zamani et al. 2016).

Following this analogy between PRF and collaborative filtering, we can find a state-of-the-art recommendation technique, SLIM (Ning and Karypis 2011), which is also based on linear methods. SLIM decomposes the full user-item feedback producing an item-item similarity matrix using ℓ_1 and ℓ_2 regularization. With this decomposition, they reconstruct the full user-item feedback matrix to generate recommendations. In contrast, we only need to predict the first row of X since we only have to expand the query. As SLIM does, LiMe fills with zeros all the missing values of the input matrix. In the beginning, in recommender systems, those unknown values were not set to zero. Instead, the objective function was optimized only for the known elements. However, later research found that this procedure produces worse rankings than dealing with the whole matrix considering all missing values as zeros Cremonesi et al. 2010.

Although RFMF and LiMe are PRF techniques based on matrix factorization, they compute different decompositions. The differences in performance are explained by the use of different objective functions and optimization algorithms. LiMe minimizes the elastic net loss and RFMF minimizes the KL-divergence of the NMF decomposition. This diversity in performance is also found in collaborative filtering where approaches such as SLIM outperforms several alternative matrix factorization techniques (Ning and Karypis 2011).

Linear methods have also been used in other information retrieval tasks. For example, (Metzler and Croft 2007) proposed a learning to rank approach based on linear models that directly maximize MAP. Moreover, linear methods have been applied to other tasks such a query difficulty prediction (Carmel and Yom-Tov 2010). In the context of PRF, (Raman et al. 2010) used logistic regression (a linear classification method) to discriminate between relevant and non-relevant terms. However, to the best of our knowledge, multiple elastic net models have never been applied before to the PRF task.

11.6 CONCLUSIONS

In this chapter, we presented LiMe, a framework where the PRF task is modeled as a matrix decomposition problem which involves the computation of inter-term or inter-document similarities. TLiMe and DLiMe factorizations are solved as linear least squares problems with ℓ_1 and ℓ_2 regularization and non-negativity constraints. For that purpose, we use not only the information from the pseudo-relevant set but also the original query before expansion. The experimental evaluation showed that TLiMe outperforms state-of-the-art baselines on five TREC datasets whereas DLiMe presents competitive effectiveness with a reduced computational cost.

In PART III, we adapted pseudo-relevance feedback models to different recommendation tasks. In this chapter, we close the circle by showing that recommendation models can also be effectively adapted to address the pseudo-relevance feedback task. Next, we discuss the findings of this thesis and present the conclusions and future work.

Part VI

CLOSING

We can only see a short distance ahead, but we can see plenty there that needs to be done.

— Alan Turing,
Computing Machinery and Intelligence

12 COMPARISON

Along the previous chapters, we proposed different recommendation models for the top-N recommendation task. We studied these models both empirically and theoretically regarding accuracy, diversity and novelty. Here, in this closing chapter, we compare the effectiveness of these approaches against several state-of-the-art collaborative filtering recommenders. We assess a representative set of neighborhood-based and model-based algorithms using the evaluation protocol followed throughout this thesis (see Section 3.2). We also discuss the advantages and disadvantages of our proposals against the characteristics of the baselines.

12.1 RECOMMENDATION BASELINES

In this section, we briefly describe the recommendation baselines. We can distinguish between neighborhood-based and model-based collaborative filtering techniques.

12.1.1 *Neighborhood-based techniques*

Among the neighborhood-based or memory-based recommender systems, NNCosNgbr is considered a good baseline (Cremonesi et al. 2010). Since we have already studied it in Section 9.2 and our proposal WSR outperformed it, we omit this recommendation approach. Instead, we use as a baseline the following neighborhood-learning technique.

12.1.1.1 *Sparse linear method (SLIM)*

Traditional neighborhood-based approaches compute neighborhoods using similarity measures. However, some recent memory-based recommenders learn neighborhoods automatically from the data, instead of using a predefined similarity measure (Ning et al. 2015).

Sparse linear method (SLIM) is a state-of-the-art item-based neighborhood learning recommender (Ning and Karypis 2011). This technique learns an item-item similarity matrix to generate recommendations. More specifically, SLIM computes a sparse aggregation coefficient matrix W solving the following optimization problem with regularization (Ning and Karypis 2011):

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & \frac{1}{2}\|R - RW\|_F^2 + \frac{\beta}{2}\|W\|_F^2 + \alpha\|W\|_{1,1} \\ \text{subject to} \quad & W \geq 0 \\ & \operatorname{diag}(W) = 0 \end{aligned} \tag{12.1}$$

where $R \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ is the ratings matrix and $W \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{I}|}$ represents the item-item similarity matrix. The score of the item i for the user u is estimated as the dot product between the user vector of ratings, $\vec{r_u}$, and the i-th column vector of W, $\vec{w_i}$:

$$\hat{r}(u,i) = \vec{r_u}\,\vec{w_i} \tag{12.2}$$

SLIM is a highly effective recommendation technique; however, its main drawback is scalability. The size of the matrix W grows quadratically with the number of items. Therefore, this technique cannot be applied in scenarios with many items.

12.1.2 *Model-based techniques*

Regarding model-based techniques, we use several matrix factorization methods as well as two very recent techniques based on neural models.

12.1.2.1 *Pure singular value decomposition (PureSVD)*

Pure singular value decomposition (PureSVD) is a factorization technique oriented to the top-N recommendation problem which has achieved high values of accuracy (Cremonesi et al. 2010). PureSVD computes the standard singular value decomposition of the ratings matrix $R \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ of rank d:

$$R \approx U\,\Sigma\,Q^T = P\,Q^T \tag{12.3}$$

where $U \in \mathbb{R}^{|\mathcal{U}|\times d}$, $\Sigma \in \mathbb{R}^{d\times d}$ and $Q^T \in \mathbb{R}^{d\times|\mathcal{I}|}$. This algorithm aims to minimize the factorization error in terms of the Frobenius norm:

$$\min \|R - P\,Q^T\|_F \tag{12.4}$$

We estimate the score for a user u and an item i as the dot product between the corresponding u-th latent vector of matrix P, $\vec{p}_u$, and the i-th latent vector from matrix Q^T, $\vec{q}_i^{\,T}$:

$$\hat{r}(u,i) = \vec{p}_u \, \vec{q}_i^{\,T} \tag{12.5}$$

12.1.2.2 *Bayesian probabilistic ranking matrix factorization (BPRMF)*

Bayesian probabilistic ranking matrix factorization (BPRMF) learns a matrix factorization based on Bayesian probabilistic ranking, a pairwise ranking loss (Rendle et al. 2009). The decomposition is the same as PureSVD ($R \approx P\,Q^T$) and the score for a user and an item is the dot product between the corresponding user and item latent vectors as shown in Equation (12.5). The optimization problem is as follows:

$$\max \sum_{u \in \mathcal{U},\, i \in \mathcal{R}_u,\, j \in \mathcal{N}_u} \log \sigma\left(\hat{r}(u,i) - \hat{r}(u,j)\right) - \lambda\left(\|P\|_F^2 + \|Q\|_F^2\right) \tag{12.6}$$

where $\mathcal{R}_u$ denotes the set of relevant items for user u and $\mathcal{N}_u$ refers to the set of non-relevant items for that user. The logistic function is denoted by $\sigma(\cdot)$ and λ controls the ℓ_2 regularization.

12.1.2.3 *Weighted regularized matrix factorization (WRMF)*

Weighted regularized matrix factorization (WRMF) is a state-of-the-art matrix factorization technique that gives more importance to the observed ratings (Hu et al. 2008). The decomposition and the rating estimate is the same as in PureSVD and BPRMF, but the optimization problem is different:

$$\min \sum_{u \in \mathcal{U},\, i \in \mathcal{I}} c_{u,i} \left(r(u,i) - \hat{r}(u,i)\right)^2 + \lambda\left(\|P\|_F^2 + \|Q\|_F^2\right) \tag{12.7}$$

where $c_{u,i} = 1 + \alpha\, r(u,i)$. The hyperparameter α controls the importance given to the ratings and λ adjusts the ℓ_2 regularization.

12.1.2.4 *Neural matrix factorization (NeuMF)*

The irruption of deep learning in RS has fostered the development of new recommendation approaches. As a representative baseline, we propose to use neural matrix factorization (NeuMF) which is a state-of-the-art neural collaborative filtering approach (He et al. 2017). The architecture of NeuMF combines matrix factorization (a linear model) and a multilayer perceptron (a non-linear model) using a logistic function in the output layer. Further information about this model can be found in the original article (He et al. 2017).

12.1.2.5 *Probabilistic recommender with item priors and neural model (PRIN)*

We recently published a new probabilistic recommender with item priors and neural model (PRIN) which is an effective model-based recommender (Landin et al. 2019). This recommender exploits the continuous bag-of-words neural model: a fully connected feed-forward network takes as input the item profile and produces as output the conditional probabilities of the users given the item. With that information, PRIN produces item recommendations through Bayesian inversion. The inversion requires the estimation of item priors that are computed using different graph-based centrality measures. More details about this algorithm can be found in the original article (Landin et al. 2019).

12.2 EVALUATION AGAINST THE STATE OF THE ART

We tuned the hyperparameters of all the baselines to maximize the figures of nDCG@100. We report the best value for each recommender system in Table 12.1. We compare these baselines against our best proposals: LM-JMS-WSR-IB and LM-JMS-RM2. Both approaches use language models smoothed with Jelinek-Mercer smoothing for computing neighborhoods (see Chapter 10). The recommendations are computed using the item-based version of weighted sum recommender (see Section 9.2.2) and RM2 (see Chapter 5).

Overall, we can see that our methods provide competitive figures of accuracy, diversity and novelty. Regarding ranking accuracy, SLIM and PRIN offer the best figures. However, our proposals present higher values of nDCG than NeuMF on all datasets. Additionally, LM-JMS-WSR-IB can outperform all the matrix factorization baselines on R3-Yahoo and LibraryThing. On the other hand, LM-JMS-RM2 outperforms PureSVD on MovieLens 1M and all the MF baselines on R3-Yahoo.

Regarding diversity and novelty, their values are tightly related to the figures of accuracy. In general, we find that any pair of recommendation techniques with similar values of ranking accuracy also present similar values of diversity and novelty. The most notable exception is the case of LM-JMS-WSR-IB and LM-JMS-RM2 on R3-Yahoo where RM2 provides the same accuracy with much better diversity. On the MovieLens datasets, we find that the higher the value of nDCG is, the lower the values of Gini and MSI are. However, on R3-Yahoo and LibraryThing datasets, we find the opposite trend to some degree. The reason is that R3-Yahoo and LibraryThing are sparser datasets and contain users with

Method	Metric	ML 100k	ML 1M	R3-Yahoo	LibraryThing
SLIM	nDCG	0.5223^{bcdegh}	$\mathbf{0.4609}^{bcdegh}$	0.0741^{bcde}	$\mathbf{0.3654}^{bcdefgh}$
	Gini	0.2479	0.1731	0.4356	0.2062
	MSI	171.97	184.53	328.10	436.50
PureSVD	nDCG	0.5107^{egh}	0.4281^{eg}	0.0663	0.2475^{eh}
	Gini	0.2801	0.1663	0.2208	0.0991
	MSI	180.27	186.73	302.80	427.85
BPRMF	nDCG	0.5181^{bdegh}	0.4575^{bdegh}	0.0721^{be}	0.2997^{bdeh}
	Gini	0.2655	0.1769	0.3683	0.0907
	MSI	178.38	183.30	312.93	377.69
WRMF	nDCG	0.5079^{egh}	0.4468^{begh}	0.0713^{b}	0.2938^{beh}
	Gini	0.3147	0.2086	0.4240	0.1524
	MSI	189.97	200.02	340.07	443.43
NeuMF	nDCG	0.4977	0.4074	0.0700^{b}	0.2262
	Gini	0.3141	0.2393	0.3926	0.1572
	MSI	190.85	203.08	321.73	422.37
PRIN	nDCG	$\mathbf{0.5233}^{bcdegh}$	0.4585^{bdegh}	$\mathbf{0.0750}^{bcdegh}$	0.2987^{bdeh}
	Gini	0.2530	0.1701	0.4021	0.0976
	MSI	170.38	179.80	319.94	378.07
LM-JMS WSR-IB	nDCG	0.4989	0.4232^{e}	0.0731^{be}	0.3118^{bcdefh}
	Gini	0.2952	0.1854	0.3520	0.3368
	MSI	190.23	191.34	318.00	499.73
LM-JMS RM2	nDCG	0.5021^{e}	0.4392^{beg}	0.0731^{bde}	0.2406^{e}
	Gini	0.2794	0.1825	0.4281	0.1285
	MSI	184.29	189.27	332.49	418.39

Table 12.1: Values of nDCG@100, Gini@100 and MSI@100 for each baseline recommender and our proposals on the MovieLens 100k and 1M, R3-Yahoo and LibraryThing datasets. Statistically significant improvements in nDCG@100 according to permutation test ($p < 0.05$) with respect to SLIM, PureSVD, BPRMF, WRMF, NeuMF, PRIN, LM-JMS-WSR-IB and LM-JMS-RM2 are superscripted with a, b, c, d, e, f, g and h, respectively. Highest value of nDCG@100 for each metric on each dataset is indicated in bold.

more diverse tastes than the MovieLens collections. Therefore, a highly accurate recommender system must be able to model this diversity.

Among matrix factorization techniques, BPRMF provides the best results. Compared to our approaches, BPRMF only presents slightly higher accuracy figures on the MovieLens datasets. Additionally, this technique provides lower diversity and novelty figures on all datasets.

Regarding neural models, NeuMF is not able to compete against other MF techniques. This may be caused by the complexity of this model: it has a high number of parameters and hyperparameters. In general, deep learning models tend to work best on scenarios with huge amounts of data but also requiring a lot of computing power to train. This approach is only able to beat our proposals in diversity and novelty on the MovieLens datasets.

PRIN presents the best figures of accuracy on MovieLens 100k and R3-Yahoo and the second-best ones on the other datasets with good values of diversity and novelty. This technique shows the potential of combining a neural model within a probabilistic framework.

Finally, SLIM provides the best figures of accuracy on MovieLens 1M and LibraryThing and the second-best values of nDCG on the other two collections with decent figures of diversity and novelty. Nevertheless, this baseline is limited by its scalability. The temporal and spatial requirements of SLIM grow quadratically with the number of items which make it infeasible in large-scale scenarios.

Most of the baselines have several hyperparameters, especially neural models such as PRIN and NeuMF. In contrast, our approaches only have two hyperparameters for the neighborhoods (the number of neighbors k and the JMS parameter of the language model λ). WSR does not add any additional hyperparameter while RM2 demands two extra hyperparameters: the additive smoothing parameter and the probabilistic item prior smoothing parameter.

Another advantage of our proposals is that they do not need training. All the baselines require a training phase to learn the recommendation model. In contrast, our proposals only need to compute neighborhoods. Although naïve kNN neighborhood computation has quadratic complexity, more efficient approaches such as L2Knng (Anastasiu and Karypis 2015) exist. If we are willing to accept a small decrease in accuracy in exchange for a huge speed-up, we can also use approximate kNN algorithms such as NN-descent (Dong et al. 2011). Moreover, we can leverage existing implementations of the language modeling framework using inverted indexes to compute neighborhoods efficiently.

The simplicity of our neighborhood-based approaches not only reduces the number of hyperparameters to tune but also provides other advantages such as explainability. Neighborhood-based recommendations are easier to explain than the suggestions produced by model-based approaches (Ning et al. 2015; Tintarev and Masthoff 2015). For example, item-based systems can show the neighbor items rated by the user as an explanation for a particular recommendation.

Finally, the last advantage of our proposals is their sound statistical foundation that allows to model prior information or use different estimators. This flexibility was crucial to be able to adapt relevance models to different recommendation tasks in Chapters 6 and 7.

13

CONCLUSIONS AND FUTURE WORK

In this chapter, we summarize the conclusions of this doctoral thesis and provide future work suggestions.

13.1 CONCLUSIONS

In this thesis, our research aim was that cross-pollination between IR and RS could lead to new useful models. We drew inspiration from information retrieval literature to study ranking accuracy metrics and adapted different pseudo-relevance feedback and ad hoc retrieval models to several recommendation tasks. We also explored the opposite direction by building a novel pseudo-relevance feedback framework inspired by recommendation models. In light of the results presented in this thesis, we can conclude that effective models and ideas can be borrowed from one field and applied to the other.

In the following, we present in more detail the findings of this work.

13.1.1 *Research Method*

In this thesis, we did not only follow common evaluation methods used in information retrieval and recommender systems research but we also further investigate RS evaluation. In Chapter 4, we leveraged the fact that ranking accuracy metrics used in the evaluation of top-N recommenders are also used in information retrieval. We proposed methodologies for comparing the robustness and the discriminative power of different ranking accuracy metrics inspired in similar studies done in IR. On the one hand, we studied cut-offs and found that deeper cut-offs offer greater robustness and discriminative power. On the other hand, we found that precision offers high robustness and normalized discounted cumulative gain provides the best discriminative power.

13.1.2 *PRF models for recommender systems*

In PART III, we examined the adaptation of different pseudo-relevance feedback models to diverse recommendation problems.

Chapter 5 thoroughly evaluates the applicability of relevance models (more specifically, the RM2 model) to user-based top-N recommendation. On the one hand, we studied the impact of smoothing methods estimators in the accuracy, diversity and novelty of the recommendations. We established a connection between the IDF effect from IR to the concept of novelty in RS and performed an axiomatic analysis to study whether different smoothing methods affect the IDF effect on RM2. We found that while collection-based methods penalize this effect, additive smoothing is neutral with respect to this property. In fact, experimentation revealed that additive smoothing offers better figures of accuracy, diversity and novelty than collection-based smoothing methods. On the other hand, we examined different user and item prior probability estimators. Our experiments showed that a uniform neighbor prior with the probabilistic estimator based on Jelinek-Mercer smoothing for items can significantly improve the ranking accuracy, diversity and novelty of the recommendation.

In Chapter 6, we formulated a novel recommendation task consisting in the liquidation of long tail items and proposed three different strategies for estimating the long tail items. We derived an item-based adaptation, the IRM2 model, of the relevance modeling framework to address the long tail item liquidation task. Our experiments showed that our proposal outperforms state-of-the-art collaborative filtering algorithms in this novel task.

We also addressed the user-item group formation problem with the IRM2 model in Chapter 7. We framed the group formation problem as an item relevance modeling task and proposed different user prior estimators to model inter-group relationships. The experiments showed that our proposal is very effective and surpasses all the baselines on the denser datasets.

Finally, in Chapter 8, we explored the adaptation of term scoring functions used in the context of the Rocchio framework to top-N recommendation. Our work led to four neighborhood-based recommenders. We also proposed a probability estimator that takes into account the size of the neighborhood yielding better recommendation rankings. We tested our proposals and found that the proposed algorithms are dramatically faster than RM2 while incurring in a small penalty in accuracy.

13.1.3 *Improving neighborhoods*

PART IV is devoted to the computation of neighborhoods, a critical piece of memory-based recommender systems.

In Chapter 9, we proposed a greedy and a cosine-based oracles that provide an approximation of the optimal user neighborhoods in a collaborative filtering scenario. These oracles showed that there is room for improvement in the memory-based collaborative filtering by computing better neighborhoods. We analyzed some characteristics of those ground truth neighborhoods to search for desirable properties. We adjusted cosine similarity with the findings of these analyses and proposed two modified versions of the similarity that improves the original formulation in terms of ranking accuracy, diversity and novelty.

Additionally, in Chapter 10, we established an analogy between ad hoc retrieval and neighborhood computation. In this way, we modeled the formation of neighborhoods using the language modeling framework. Our proposal leveraged the advantages of this successful retrieval technique for calculating user and item neighborhoods. We found that the query likelihood model with Jelinek-Mercer smoothing outperforms cosine similarity in accuracy, diversity and novelty. We also provided an axiomatic analysis that showed that our proposal satisfies two desirable properties that cosine similarity does not.

13.1.4 *Recommender systems models for PRF*

In PART V, we explored the opposite path to the one studied in PART III: adapting recommendation models to pseudo-relevance feedback. We presented the adaptation of linear methods, used in recommendation techniques such as SLIM in Chapter 11. This work resulted in a novel and effective PRF framework called LiMe. We derived two formulations of the PRF task involving inter-document and inter-term similarities and an algorithm based on constrained elastic net regression for solving the proposed models and computing the expansion terms. The empirical evaluation showed that the term-based formulation outperforms the state of the art whereas the document-based model presents competitive effectiveness with a reduced computational cost.

13.1.5 *Final Implications*

The effectiveness achieved by the recommendation models derived from IR ideas supports the research aim of this thesis. Our proposals not only offer competitive effectiveness, but are also simple and explainable recommendation models. Additionally, the flexibility of their probabilistic formulation enables their adaptation to different recommendation tasks. Finally, we also found that pseudo-relevance feedback can benefit from models developed by the RS community.

13.2 FUTURE DIRECTIONS

Next, we propose future lines of work to continue the research presented in this thesis.

13.2.1 *Research Method*

We studied the robustness and discriminative power of ranking accuracy metrics. However, we envision to extend this analysis to different types of metrics. Apart from ranking accuracy, diversity and novelty are also important properties of recommender systems (Castells et al. 2015). Additionally, it would be worth studying the impact of different dataset partitioning schemes such as temporal splits and n-fold cross-validation.

13.2.2 *PRF models for recommender systems*

We found that additive smoothing does not promote nor demote the IDF effect in relevance models. However, it would be interesting to develop new smoothing methods that do actively promote this effect. Additionally, we think that performing an axiomatic analysis of the proposed priors for relevance models may be useful to identify why some estimators are more effective than others. Finally, it would be interesting to analyze if the combination of different priors can lead to better results as in IR (Peng and Ounis 2007).

Regarding the problem of liquidating long tail items, we used relevance models with uniform priors. Therefore, it would be interesting to analyze the effect of priors in IRM2. Moreover, the probability distributions of the user priors can be modified to introduce business rules in the model. For example, we can demote the probability of certain VIP users because we do not want to overflow them with liquidation advertisements.

We also think that our work on group formation with item-based relevance models opens the way for further research. For example, it would be interesting to compute the optimal size of the recommended group automatically—perhaps we can exploit the probability scores for this. Additionally, we have seen that the effectiveness of IRM2 in this task depends largely on the prior estimators. Therefore, we recommend further work on this topic to develop better prior estimators. Moreover, an axiomatic analysis may shed light on the effectiveness of the priors.

Finally, we have seen that term scoring functions used within the Rocchio framework are cost-effective alternatives to RM2 in the top-N recommendation task. We think that exploring other state-of-the-art PRF techniques such as divergence minimization models or mixture models (Tao and Zhai 2006; Zhai and Lafferty 2001) may be a fruitful area for further research.

13.2.3 *Improving neighborhoods*

We proposed two oracles for computing neighborhoods that led us to improvements in cosine similarity. We think that this work paves the way for plenty of future research in the area of neighborhood-based recommender systems. In this thesis, we focused our analysis on user-based approaches, but extending the presented study to the item-based counterpart would be interesting. Additionally, we envision to explore other neighborhood-based collaborative filtering techniques apart from WSR as well as other methods for computing neighborhoods different from *k*NN. Furthermore, we investigated how to improve cosine similarity, a measure based on rating co-occurrence in user profiles. However, to approach the high figures of the greedy oracle, this will not suffice. Hence, it would be interesting to study other aspects such as trustiness or expertise to complement co-occurrence.

We also envision to expand our work on language models for computing neighborhoods. In this thesis, we used the query likelihood model. We found that Jelinek-Mercer provides the best results although Dirichlet priors is also a great choice. This result is analogous to the information retrieval task where JMS works better than DPS for long queries (Zhai and Lafferty 2004). To overcome the problem that JMS does not vary the amount of smoothing applied depending on the document length (in contrast to DPS), Losada and Azzopardi (2008a) proposed the use of a length-based document prior. This prior is equivalent to the linear prior that we used in RM2 (see Section 5.3.1.2). Testing the applicability of this prior combined with JMS smoothing would be an interesting avenue for further work. Furthermore, we think that exploring language models based on different probability distributions such as the multivariate Bernoulli (Losada and Azzopardi 2008b) may be worthwhile.

Finally, we also intend to explore neural models that are becoming increasingly popular in IR (Mitra and Craswell 2018). However, we take the reported results of these models with a grain of salt due to recent criticism (Lin 2018).

13.2.4 *Recommender systems models for PRF*

We found that not only can ideas from IR be adapted to RS, but also the other way around. The good results achieved by the LiMe framework using only TF-IDF suggest that exploring alternative features may be an interesting avenue for future work. We also envision to include richer representations of text features in the model. For example, the use of features extracted from Wikipedia has proved to be beneficial in PRF (Xu et al. 2009). Additionally, we plan to study other similarity measures used in translation models which are based on inter-term similarities (Berger and Lafferty 1999; Lafferty and Zhai 2001). Previous work on translation models learned inter-term similarities from training data (Berger and Lafferty 1999) or employed mutual information (Karimzadehgan and Zhai 2010).

APPENDICES

A

PUBLICATIONS

In this appendix, we list all the articles published during the doctoral period. For the conferences, we provide their rank according to CORE 2018[1] and the acceptance rate of the full or short papers track. For each journal, we detail its Journal Citation Reports Impact Factor[2] and its quartile.

A.1 CONFERENCE ARTICLES

Alfonso Landin, Eva Suárez-García and Daniel Valcarce. “When Diversity Met Accuracy: A Story of Recommender Systems.” In: *Proceedings of XoveTIC 2018*. Vol. 2. 18. Basel, Switzerland: MDPI, 2018, p. 1178. DOI: 10.3390/proceedings2181178.

Alfonso Landin, Daniel Valcarce, Javier Parapar and Álvaro Barreiro. “PRIN: A Probabilistic Recommender with Item Priors and Neural Models.” In: *Proceedings of the 41th European Conference on Information Retrieval*. ECIR ’19. Berlin, Heidelberg: Springer, 2019. CORE 2018: A. Acceptance rate (full papers): 23%.

Eva Suárez-García, Alfonso Landin, Daniel Valcarce and Álvaro Barreiro. “Term Association Measures for Memory-based Recommender Systems.” In: *Proceedings of the 5th Spanish Conference on Information Retrieval*. Vol. 18. CERI ’18. New York, NY, USA: ACM, 2018, pp. 1–8. DOI: 10.1145/3230599.3230606. Best Student Paper Award.

Daniel Valcarce. “Exploring Statistical Language Models for Recommender Systems.” In: *Proceedings of the 9th ACM Conference on Recommender Systems*. RecSys ’15. ACM, 2015, pp. 375–378. DOI: 10.1145/2792838.2796547. CORE 2018: B. Doctoral Symposium paper.

1 The CORE 2018 Conference ranking is available at: http://portal.core.edu.au/conf-ranks.

2 The JCR Impact Factor can be consulted at: https://jcr.incites.thomsonreuters.com.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. “When Recommenders Met Big Data: An Architectural Proposal and Evaluation.” In: *Proceedings of the 3rd Spanish Conference on Information Retrieval*. CERI ’14. 2014, pp. 73–84.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. “A Study of Priors for Relevance-Based Language Modelling of Recommender Systems.” In: *Proceedings of the 9th ACM Conference on Recommender Systems*. RecSys ’15. New York, NY, USA: ACM, 2015, pp. 237–240. DOI: 10.1145/2792838.2799677. CORE 2018: B. Acceptance rate (short papers): 26%.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. “A Study of Smoothing Methods for Relevance-Based Language Modelling of Recommender Systems.” In: *Proceedings of the 37th European Conference on Information Retrieval*. Vol. 9022. ECIR ’15. Berlin, Heidelberg: Springer, 2015, pp. 346–351. DOI: 10.1007/978-3-319-16354-3_38. CORE 2018: A. Acceptance rate (short papers): 38%.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. “Additive Smoothing for Relevance-Based Language Modelling of Recommender Systems.” In: *Proceedings of the 4th Spanish Conference on Information Retrieval*. CERI ’16. New York, NY, USA: ACM, 2016, pp. 1–8. DOI: 10.1145/2934732.2934737. Best Student Paper Award.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. “Computing Neighbourhoods with Language Models in a Collaborative Filtering Scenario.” In: *Proceedings of the 7th Italian Information Retrieval Workshop*. Vol. 1653. IIR ’16. CEUR Workshop Proceedings, 2016. URL: http://ceur-ws.org/Vol-1653/paper_13.pdf.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. “Efficient Pseudo-Relevance Feedback Methods for Collaborative Filtering Recommendation.” In: *Proceedings of the 38th European Conference on Information Retrieval*. ECIR ’16. Berlin, Heidelberg: Springer, 2016, pp. 602–613. DOI: 10.1007/978-3-319-30671-1_44. CORE 2018: A. Acceptance rate (full papers): 22%.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. “Language Models for Collaborative Filtering Neighbourhoods.” In: *Proceedings of the 38th European Conference on Information Retrieval*. ECIR ’16. Berlin, Heidelberg: Springer, 2016, pp. 614–625. DOI: 10.1007/978-3-319-30671-1_45. CORE 2018: A. Acceptance rate (full papers): 22%.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. “Combining Top-N Recommenders with Metasearch Algorithms.” In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’17. New York, NY, USA: ACM,

2017, pp. 805–808. DOI: 10.1145/3077136.3080647. CORE 2018: A*. Acceptance rate (short papers): 30%.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. "LiMe: Linear Methods for Pseudo-Relevance Feedback." In: *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. SAC '18. New York, NY, USA: ACM, 2018, pp. 678–687. DOI: 10.1145/3167132.3167207. CORE 2018: B. Acceptance rate (full papers): 25%.

Daniel Valcarce, Alejandro Bellogín, Javier Parapar and Pablo Castells. "On the robustness and discriminative power of information retrieval metrics for top-N recommendation." In: *Proceedings of the 12th ACM Conference on Recommender Systems*. RecSys '18. New York, NY, USA: ACM, 2018, pp. 260–268. DOI: 10.1145/3240323.3240347. CORE 2018: B. Acceptance rate (full papers): 18%.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. "Query Expansion as a Matrix Factorization Problem." In: *Proceedings of the 5th Spanish Conference on Information Retrieval*. CERI '18. New York, NY, USA: ACM, 2018, pp. 1–4. DOI: 10.1145/3230599.3230603.

A.2 JOURNAL ARTICLES

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. "A Distributed Recommendation Platform for Big Data." In: *Journal of Universal Computer Science* 21.13 (2015), pp. 1810–1829. DOI: 10.3217/jucs-021-13-1810. IF 2015: 0.546 (Q4).

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. "Item-based Relevance Modelling of Recommendations for Getting Rid of Long Tail Products." In: *Knowledge-Based Systems* 103 (2016), pp. 41–51. DOI: 10.1016/j.knosys.2016.03.021. IF 2016: 4.529 (Q1).

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. "Axiomatic Analysis of Language Modelling of Recommender Systems." In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 25.Suppl. 2 (2017), pp. 113–127. DOI: 10.1142/S0218488517400141. IF 2017: 1.159 (Q3).

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. "A MapReduce implementation of posterior probability clustering and relevance models for recommendation." In: *Engineering Applications of Artificial Intelligence* 75 (2018), pp. 114–124. DOI: 10.1016/j.engappai.2018.08.006. IF 2017: 2.819 (Q1).

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. "Document-based and Term-based Linear Methods for Pseudo-Relevance Feedback." In:

ACM SIGAPP Applied Computing Review 18.4 (2018), pp. 5–17. URL: https://www.sigapp.org/acr/Issues/V18.4/ACR%2018-4.pdf.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. "Finding and analysing good neighbourhoods to improve collaborative filtering." In: *Knowledge-Based Systems* 159 (2018), pp. 193–202. DOI: 10.1016/j.knosys.2018.06.030. IF 2017: 4.396 (Q1).

Daniel Valcarce, Igo Brilhante, Jose Antonio Macedo, Franco Maria Nardini, Raffaele Perego and Chiara Renso. "Item-driven group formation." In: *Online Social Networks and Media* 8 (2018), pp. 17–31. DOI: 10.1016/j.osnem.2018.10.002.

B

EXTENDED SUMMARY IN SPANISH

In accordance with the current Regulations of the PhD Studies of the Universidade da Coruña, we present in this appendix an extended summary of this doctoral thesis in Spanish.

B.1 INTRODUCCIÓN

La historia de la humanidad ha sido moldeada por la forma en la que hemos manejado la información. A medida que surgieron las sociedades, el desarrollo de la escritura, hace más de cinco milenios, fue el primer hito. Empezamos a recopilar información que sólo se había difundido de boca en boca. Impulsados por necesidades pragmáticas, continuamos creando nuevas formas de almacenar y procesar información. Las bibliotecas, los principales lugares donde se almacenaba y conservaba la información escrita, florecieron. Sin embargo, la cantidad de datos era mínima debido al alto coste de la escritura a mano. Fue la invención de la imprenta la que multiplicó exponencialmente la tasa de crecimiento de la información escrita. Más recientemente, la invención de la computadora llevó a desarrollos modernos fundamentales como el diseño de estructuras de datos especializadas para la consulta de bibliotecas digitales. Finalmente, la aparición de la World Wide Web en 1989 desencadenó una explosión sin precedentes en la disponibilidad de información. La creación de Tim Berners-Lee se ha convertido en un depósito universal de conocimiento humano que ha transformado la forma en la que accedemos a la información.

La recuperación de información y el filtrado de información son dos campos de estudio que giran en torno al procesamiento de la información. El desarrollo de las tecnologías de la computación y de la comunicación ha aumentado la importancia de estos campos. Los sistemas de recuperación de información se ocupan de la representación, almacenamiento y acceso a la información. Su objetivo es exponer a los usuarios a información relevante de acuerdo a sus necesidades (Baeza-Yates and Ribeiro-Neto 2011; Manning et al. 2008). Por otra parte, los sistemas de filtrado de información tienen por objeto seleccionar aquellos elementos de un flujo de información que pueden ser de interés para un usuario determinado (Hanani et al. 2001). Entre los diferentes tipos de filtros de información, los sistemas de recomendación son probablemente los más importantes en la actualidad. El objetivo de un sistema de recomendación es generar sugerencias personalizadas para los artículos basados en los intereses de un usuario (Ricci et al. 2015).

Dado que el objetivo final de los sistemas de recuperación de información y los de filtrado de información es, en última instancia, proporcionar a los usuarios elementos de información relevantes, algunos autores consideran que ambos campos son dos caras de la misma moneda (Belkin and Croft 1992). No obstante, a pesar de las similitudes entre la recuperación de información y el filtrado de información, ha habido poca investigación sobre la aplicación de técnicas clásicas de recuperación de información a sistemas de recomendación hasta hace poco tiempo (Bellogín et al. 2013a; Kallumadi et al. 2018; Parapar et al. 2013). En esta tesis doctoral, pretendemos salvar la brecha entre la recuperación de información y los sistemas de recomendación, adaptando varios modelos de recuperación de información a diferentes tareas de recomendación y estableciendo nuevas analogías entre ambos campos.

B.2 MOTIVACIÓN

Los primeros trabajos sobre recuperación de información se remontan a los años 50 (Kent et al. 1955; Mooers 1951). Desde entonces, la recuperación de información ha evolucionado enormemente. La primera conferencia de la ACM sobre recuperación de información (SIGIR) se celebró en 1971 y actualmente cuenta con cientos de asistentes. Además, la Web ha traído un nuevo paradigma de acceso a la información en el que la búsqueda se ha vuelto crucial. Por el contrario, los sistemas de recomendación son un campo mucho más joven. Esta área surgió a mediados de los años 90 con la explosión de la World Wide Web (Resnick and Varian 1997; Resnick et al. 1994; Shardanand and Maes 1995). Aunque la primera conferencia de la ACM sobre sistemas de recomendación (RecSys) tuvo lugar en 2007, ha crecido muy rápidamente y actualmente atrae a cientos de asistentes, muchos de ellos de la industria.

A medida que la Web proporciona cada vez más información, los sistemas de información tienen que enfrentarse a nuevos retos. El volumen de información disponible para el público resulta abrumador. La dificultad para encontrar y seleccionar información relevante aumenta a medida que más contenido está disponible. Sin las herramientas adecuadas para hacer frente a la sobrecarga de información, los usuarios pueden perderse información interesante o consumir contenido poco interesante. Por esta razón, las ciencias de la información, como la recuperación y el filtrado de la información, son cruciales en el panorama actual.

Los sistemas de recuperación de información suelen estar orientados a producir listas ordenadas de documentos ordenados por relevancia. Aunque los sistemas de recomendación se orientaron inicialmente a predecir con precisión las valoraciones de los usuarios a los ítems, se ha producido un cambio de paradigma hacia la generación de listas de ítems ordenadas por relevancia estimada. Por lo tanto, las técnicas de recuperación de información y de sistemas de recomendación modernas parecen tener objetivos muy similares.

Los sistemas de recomendación se han convertido en una tecnología omnipresente para abordar el problema de la sobrecarga de información. El enorme ritmo de crecimiento de los datos ha cambiado radicalmente la forma en que accedemos a la información. Además, a medida que los sistemas de información ofrecen capacidades más avanzadas, los usuarios son cada vez más exigentes. En este contexto, las funciones de búsqueda tradicionales no son suficientes. Los usuarios esperan recibir de los sistemas sugerencias proactivas en lugar de tener que especificar explícitamente consultas que expresen sus necesidades de información.

La principal diferencia entre los sistemas de recuperación de información y los sistemas de recomendación radica en la representación de la necesidad de información: mientras que un sistema de recuperación de información suele utilizar una consulta especificada explícitamente por el usuario, un sistema de recomendación explota el historial del usuario como una consulta implícita. Sin embargo, al final, ambos campos comparten el mismo objetivo: proporcionar a los usuarios acceso a información relevante.

B.3 OBJETIVOS Y ALCANCE

Creemos que el intercambio de ideas entre la recuperación de información y los sistemas de recomendación puede conducir a nuevos y útiles enfoques. En esta tesis doctoral, retrocedemos a las raíces del campo de los sistemas de recomendación y exploramos su relación con la recuperación de información. La recuperación de información ha existido durante más tiempo que los sistemas de recomendación; por lo tanto, creemos que podemos aprovechar el trabajo y el conocimiento existente desarrollado por la comunidad de recuperación de información para proponer nuevos modelos de recomendación. Sin embargo, también pensamos que podemos inspirarnos en las técnicas de recomendación para dar un soplo de aire fresco a tareas consolidadas en recuperación de información.

En esta tesis doctoral, nos centramos en la aplicabilidad de algunos modelos de recuperación de información en problemas de recomendación. Limitamos el alcance de este trabajo a dos tareas principales de recuperación de información: la recuperación ad hoc y la retroalimentación de pseudo-relevancia. En particular, exploramos la adaptación de modelos de recuperación ad hoc para computar vecindarios y algoritmos de retroalimentación de pseudo-relevancia para diferentes tareas de recomendación. Por otro lado, para cerrar el círculo, al final de esta tesis también exploramos cómo los métodos lineales utilizados en la recomendación pueden construir modelos efectivos de retroalimentación de pseudo-relevancia.

La evaluación desempeña un papel crucial en las ciencias experimentales, como la recuperación de información y los sistemas de recomendación. En esta tesis, evaluamos la efectividad y eficiencia del modelo propuesto empleando colecciones de datos. Este enfoque suele constituir el primer paso en la evaluación debido a sus costes reducidos y a su alta reproducibilidad. Por el contrario, las evaluaciones en vivo requieren experimentos con usuarios reales que son caros y difíciles de realizar en la academia. Por estas razones, nos centramos en realizar nuestros experimentos con conjuntos de datos públicos.

B.4 ESTRUCTURA

Esta tesis doctoral se divide en seis partes con trece capítulos. El Capítulo 1 contiene la introducción a este trabajo. El Capítulo 2 presenta los principales conceptos de las áreas de la recuperación de información y de los sistemas de recomendación. Aunque un especialista en los temas puede omitirlo, un lector interesado puede encontrarlo una introducción útil a ambos campos. El Capítulo 3 detalla los protocolos de evaluación y las configuraciones experimentales utilizadas a lo largo de esta investigación. Los Capítulos 4 a 11 presentan las contribuciones novedosas de esta tesis. Los capítulos de contribuciones han sido redactados para ser lo más independientes posible. Por ello, pueden ser leídos y comprendidos con solo la información proporcionada en el Capítulo 2. El Capítulo 12 presenta la comparación y discusión de los hallazgos de esta tesis. Finalmente, el Capítulo 13 contiene las conclusiones y el trabajo futuro. A continuación, presentamos la organización de las partes y capítulos más detalladamente:

PARTE I La primera parte incluye el Capítulo 1, que es la introducción a esta tesis, y el Capítulo 2, que discute los antecedentes. La introducción presenta el contexto y la motivación de la tesis, el objetivo y el alcance de nuestro trabajo y la estructura y contribuciones del estudio. El capítulo de antecedentes, por otro lado, presenta una visión general de los sistemas de recuperación de información y de recomendación e introduce los principales conceptos de ambos campos. En cuanto a la recuperación de información, nos centramos en la recuperación ad hoc y en la retroalimentación de pseudo-relevancia, que adaptamos en los capítulos siguientes a las tareas de recomendación. También presentamos trabajos anteriores que estudian o explotan la relación entre recuperación de información y sistemas de recomendación.

PARTE II Esta parte de la tesis describe los métodos de investigación en dos capítulos. Por un lado, el Capítulo 3 describe la evaluación en recuperación de información y sistemas de recomendación utilizada a lo largo de este trabajo. Por otro lado, el Capítulo 4 contiene un estudio novedoso de la robustez y el poder discriminatorio de las métricas utilizadas para evaluar listas de recomendaciones ordenadas. Los resultados de este estudio justifican las métricas de evaluación empleadas en esta tesis.

PARTE III Presentamos aquí la adaptación de varios modelos de retroalimentación de pseudo-relevancia a diferentes tareas de recomendación. En particular, el Capítulo 5 mejora una adaptación existente de modelos de relevancia a recomendación mediante la exploración de técnicas de suavización y de estimadores de probabilidad a priori. El Capítulo 6 propone una adaptación complementaria basada en ítems de los modelos de relevancia que usamos para resolver un nuevo problema de recomendación: cómo liquidar ítems de la larga cola. El Capítulo 7 emplea los modelos relevancia basados en ítems con estimadores personalizados de probabilidad a priori de usuario para abordar la tarea de formación de grupos a partir de un usuario-ítem. Por último, exploramos la adaptación de las técnicas de retroalimentación de pseudo-relevancia propuestas en el marco de Rocchio a la tarea convencional de recomendación en el Capítulo 8.

PARTE IV Esta parte contiene dos capítulos de enfocados en los vecindarios de los sistemas de filtrado colaborativo. El Capítulo 9 mide el margen de mejora de diferentes técnicas para computar vecindarios. Basado en estos hallazgos, proponemos modificaciones de principio de la similitud del coseno basados en esquemas de normalización usados en recuperación de información. El Capítulo 10, por otro lado, propone la adaptación de los modelos de lenguaje estadístico empleados en la recuperación ad hoc a la computación de vecindarios.

PARTE V En esta parte, exploramos la dirección contraria adaptando una técnica de recomendación para realizar retroalimentación de pseudo-relevancia. En particular, el Capítulo 11 describe cómo los métodos lineales dispersos, utilizados con gran éxito en recomendación, también pueden utilizarse para expandir las consultas de los usuarios y mejorar la eficacia de la búsqueda.

PARTE VI En la última parte, el Capítulo 12 discute los resultados obtenidos y los compara con el estado del arte. Finalmente, el Capítulo 13 resume las contribuciones de esta tesis, presenta las conclusiones y sugiere futuras líneas de investigación.

B.5 CONCLUSIONES

En esta tesis, el objeto de nuestra investigación es explorar el intercambio de ideas entre la recuperación de información y los sistemas de recomendación con la finalidad de que pueda conducir a nuevos modelos. Nos inspiramos en la literatura de recuperación de información para estudiar las métricas de evaluación en recomendación y adaptamos diferentes modelos de retroalimentación de pseudo-relevancia y de recuperación ad hoc a varias tareas de recomendación. También exploramos la dirección opuesta construyendo un nuevo método de retroalimentación de pseudo-relevancia inspirado en modelos de recomendación. A la luz de los resultados presentados en esta tesis, podemos concluir que los modelos e ideas eficaces pueden ser tomados de un campo y aplicados al otro.

A continuación, presentamos con más detalle las conclusiones de este trabajo.

B.5.1 *Método de evaluación*

En esta tesis, no solo seguimos los métodos de evaluación comúnmente utilizados en la investigación en recuperación de información y sistemas de recomendación, sino que también investigamos a fondo diversas alternativas. En el Capítulo 4, aprovechamos el hecho de que las métricas de precisión utilizadas en la evaluación de las recomendaciones también se emplean en recuperación de información. Proponemos metodologías para comparar la robustez y el poder discriminatorio de diferentes métricas de precisión inspiradas en estudios similares realizados en recuperación de información. Por un lado, estudiamos diferentes cortes de métricas y encontramos que los más profundos ofrecen mayor robustez y poder discriminatorio. Por otro lado, comprobamos que la precisión ofrece una alta robustez, mientras que la ganancia acumulativa descontada normalizada proporciona el mejor poder discriminatorio.

B.5.2 *Modelos de retroalimentación de pseudo-relevancia para sistemas de recomendación*

En la PARTE III, examinamos la adaptación de diferentes modelos de retroalimentación de pseudo-relevancia a diversos problemas de recomendación.

El Capítulo 5 evalúa a fondo la aplicabilidad de los modelos de relevancia (más específicamente, el modelo RM2) a la recomendación basada en usuarios. Por un lado, estudiamos el impacto de los estimadores de los métodos de suavización en la precisión, diversidad y novedad de las recomendaciones. Establecimos una conexión entre el efecto IDF de recuperación de información y el concepto de novedad en los sistemas de recomendación y realizamos un análisis axiomático para estudiar si los diferentes métodos de suavización afectan al efecto IDF de RM2. Comprobamos formalmente que mientras que los métodos de suavizado basados en la colección penalizan este efecto, la suavización aditiva presenta un comportamiento neutro con respecto a esta propiedad. De hecho, la experimentación reveló que la suavización aditiva ofrece mejores cifras de precisión, diversidad y novedad que los métodos de suavización basados en la colección. Por otro lado, se examinaron diferentes estimadores de probabilidad a priori de usuarios y de ítems. Nuestros experimentos demostraron que un estimador uniforme para los vecinos junto con un estimador probabilístico basado en la suavización de Jelinek-Mercer para los ítems puede mejorar significativamente la precisión, la diversidad y la novedad de las recomendaciones.

En el Capítulo 6, formulamos una novedosa tarea de recomendación consistente en la liquidación de ítems de la larga cola y propusimos tres estrategias diferentes para estimar los ítems de la larga cola. Derivamos una adaptación basada en ítems de los modelos de relevancia, a la que llamamos IRM2, para abordar la tarea de liquidación de ítems de la larga cola. Nuestros experimentos demostraron que nuestra propuesta supera en esta tarea a los algoritmos de filtrado colaborativo convencionales del estado del arte.

También abordamos el problema de la formación de grupos a partir de un usuario y un ítem con el modelo de IRM2 en el Capítulo 7. Enmarcamos el problema de formación de grupos como una tarea de modelado de relevancia de ítems y propusimos diferentes estimadores probabilidad a priori de usuarios para modelar las relaciones intergrupales. Los experimentos demostraron que nuestra propuesta es muy efectiva y supera a las técnicas existentes en las colecciones de datos más densas.

Finalmente, en el Capítulo 8, exploramos la adaptación de las funciones de puntuación de términos utilizadas en el contexto del método de Rocchio a la recomendación de listas ordenadas de ítems. Nuestro trabajo condujo a la creación de cuatro recomendadores basados en vecindarios. También propusimos una estimación de probabilidad que considera el tamaño del vecindario, lo que produce mejores listas de recomendaciones. Probamos nuestras propuestas y encontramos que los algoritmos propuestos son drásticamente más rápidos que RM2 incurriendo en una pequeña penalización en precisión.

B.5.3 *Mejorando vecindarios*

La PARTE IV está dedicada al cálculo de vecindarios, una pieza crítica de los sistemas de recomendación basados en memoria.

En el Capítulo 9, propusimos un oráculo basado en un algoritmo voraz y un oráculo basado en la similitud coseno que proporcionan una aproximación de los vecindarios óptimos en un escenario de filtrado colaborativo. Estos oráculos mostraron que hay margen de mejora en la computación de vecindarios en el filtrado colaborativo basado en memoria. Analizamos algunas características de esos vecindarios de referencia para buscar propiedades deseables. Ajustamos la similitud coseno con los resultados de estos análisis y propusimos dos versiones modificadas de dicha similitud que mejoran la formulación original en términos de precisión, diversidad y novedad.

Además, en el Capítulo 10, establecimos una analogía entre la recuperación ad hoc y el cálculo de vecindarios. De esta manera, modelamos la formación de vecindarios utilizando modelos de lenguaje estadísticos. Nuestra propuesta aprovechó las ventajas de esta exitosa técnica de recuperación de información para calcular los vecindarios de usuarios e ítems. Comprobamos que el modelo de verosimilitud de la consulta junto con el índice de suavización de Jelinek-Mercer supera la similitud coseno en precisión, diversidad y novedad. También proporcionamos un análisis axiomático que mostró que nuestra propuesta satisface dos propiedades deseables que la similitud del coseno no satisface.

B.5.4 *Modelos de sistemas de recomendación para retroalimentación de pseudo-relevancia*

En la PARTE V, exploramos el camino opuesto al estudiado en PARTE III: adaptar los modelos de recomendación a la retroalimentación de pseudo-relevancia. Presentamos la adaptación de métodos lineales, utilizados en técnicas de recomendación como SLIM, en el Capítulo 11. Este trabajo dio como resultado un nuevo método de retroalimentación de pseudo-relevancia al que llamamos LiMe. Derivamos dos formulaciones que involucran similitudes entre documentos y entre términos y un algoritmo basado en regresión de red elástica con restricciones para resolver los modelos propuestos y calcular los términos de expansión. La evaluación empírica mostró que la formulación basada en términos supera a técnicas del estado del arte, mientras que el modelo basado en documentos presenta una eficacia competitiva con un coste computacional reducido.

B.5.5 *Implicaciones finales*

La eficacia alcanzada por los modelos de recomendación derivados de ideas de recuperación de información respalda el objeto de investigación de esta tesis. Nuestras propuestas no solo presentan una eficacia competitiva, sino que también son modelos de recomendación simples y explicables. Además, la flexibilidad de su formulación probabilística permite su adaptación a diferentes tareas de recomendación. Finalmente, también comprobamos que la retroalimentación de pseudo-relevancia puede beneficiarse de los modelos desarrollados por la comunidad de sistemas de recomendación.

BIBLIOGRAPHY

Behnoush Abdollahi and Olfa Nasraoui. "Explainable Matrix Factorization for Collaborative Filtering." In: *Proceedings of the 25th International Conference Companion on World Wide Web*. WWW '16 Companion. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2016, pp. 5–6. DOI: 10.1145/2872518.2889405.

Nasreen Abdul-Jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D. Smucker and Courtney Wade. "UMass at TREC 2004: Novelty and HARD." In: *Proceedings of the 13th Text REtrieval Conference*. Portland, Oregon, USA, 2004, pp. 1–13.

Panagiotis Adamopoulos and Alexander Tuzhilin. "On Over-specialization and Concentration Bias of Recommendations: Probabilistic Neighborhood Selection in Collaborative Filtering Systems." In: *Proceedings of the 8th ACM Conference on Recommender Systems*. RecSys '14. New York, NY, USA: ACM, 2014, pp. 153–160. DOI: 10.1145/2645710.2645752.

Gediminas Adomavicius and Youngok Kwon. "Improving Aggregate Recommendation Diversity Using Ranking-Based Techniques." In: *IEEE Transactions on Knowledge and Data Engineering* 24.5 (2012), pp. 896–911. DOI: 10.1109/TKDE.2011.15.

Gediminas Adomavicius and Alexander Tuzhilin. "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions." In: *IEEE Transactions on Knowledge and Data Engineering* 17.6 (2005), pp. 734–749. DOI: 10.1109/TKDE.2005.99.

Gediminas Adomavicius and Alexander Tuzhilin. "Context-Aware Recommender Systems." In: *Recommender Systems Handbook*. Ed. by Francesco Ricci, Lior Rokach and Bracha Shapira. Boston, MA, USA: Springer, 2015, pp. 191–226. DOI: 10.1007/978-1-4899-7637-6_6.

Charu C Aggarwal and ChengXiang Zhai. "A Survey of Text Clustering Algorithms." In: *Mining Text Data*. Boston, MA, USA: Springer, 2012, pp. 77–128. DOI: 10.1007/978-1-4614-3223-4_4.

Aris Anagnostopoulos, Reem Atassi, Luca Becchetti, Adriano Fazzone and Fabrizio Silvestri. "Tour recommendation for groups." In: *Data Mining and Knowledge Discovery* 31.5 (2017), pp. 1157–1188. DOI: 10.1007/s10618-016-0477-7.

David C. Anastasiu and George Karypis. “L2Knng: Fast Exact K-Nearest Neighbor Graph Construction with L2-Norm Pruning.” In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. CIKM ’15. New York, NY, USA: ACM, 2015, pp. 791–800. DOI: 10.1145/2806416.2806534.

Chris Anderson. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hachette Books, 2008.

Amos Azaria, Avinatan Hassidim, Sarit Kraus, Adi Eshkol, Ofer Weintraub and Irit Netanely. “Movie Recommender System for Profit Maximization.” In: *Proceedings of the 7th ACM Conference on Recommender Systems*. RecSys ’13. New York, NY, USA: ACM, 2013, pp. 121–128. DOI: 10.1145/2507157.2507162.

Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval: The Concepts and Technology Behind Search*. Addison Wesley, 2011, p. 913.

Linas Baltrunas and Francesco Ricci. “Locally Adaptive Neighborhood Selection for Collaborative Filtering Recommendations.” In: *Proceedings of the 5th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*. AH ’08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 22–31. DOI: 10.1007/978-3-540-70987-9_5.

Nicola Barbieri and Giuseppe Manco. “An Analysis of Probabilistic Methods for Top-N Recommendation in Collaborative Filtering.” In: *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. ECML PKDD ’11. 2011, pp. 172–187. DOI: 10.1007/978-3-642-23780-5_21.

Nicola Barbieri, Giuseppe Manco and Ettore Ritacco. “Probabilistic Approaches to Recommendations.” In: *Synthesis Lectures on Data Mining and Knowledge Discovery* 5.2 (2014), pp. 1–197. DOI: 10.2200/S00574ED1V01Y201403DMK009.

Senjuti Basu Roy, Laks V. S. Lakshmanan and Rui Liu. “From Group Recommendations to Group Formation.” In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’15. New York, NY, USA: ACM, 2015, pp. 1603–1616. DOI: 10.1145/2723372.2749448.

Nicholas J. Belkin and W. Bruce Croft. “Information Filtering and Information Retrieval: Two Sides of the Same Coin?” In: *Communications of the ACM* 35.12 (1992), pp. 29–38. DOI: 10.1145/138859.138861.

Alejandro Bellogín and Pablo Castells. “Predicting Neighbor Goodness in Collaborative Filtering.” In: *Proceedings of the 8th International Conference on Flexible Query Answering Systems*. FQAS ’09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 605–616. DOI: 10.1007/978-3-642-04957-6_52.

Alejandro Bellogín and Javier Parapar. "Using Graph Partitioning Techniques for Neighbour Selection in User-based Collaborative Filtering." In: *Proceedings of the 6th ACM Conference on Recommender Systems*. RecSys '12. New York, NY, USA: ACM, 2012, pp. 213–216. DOI: 10.1145/2365952.2365997.

Alejandro Bellogín, Pablo Castells and Iván Cantador. "Precision-oriented evaluation of recommender systems." In: *Proceedings of the 5th ACM Conference on Recommender Systems*. RecSys '11. New York, NY, USA: ACM, 2011, p. 333. DOI: 10.1145/2043932.2043996.

Alejandro Bellogín, Jun Wang and Pablo Castells. "Bridging memory-based collaborative filtering and text retrieval." In: *Information Retrieval* 16.6 (2013), pp. 697–724. DOI: 10.1007/s10791-012-9214-z.

Alejandro Bellogín, Javier Parapar and Pablo Castells. "Probabilistic Collaborative Filtering with Negative Cross Entropy." In: *Proceedings of the 7th ACM Conference on Recommender Systems*. RecSys '13. New York, NY, USA: ACM, 2013, pp. 387–390. DOI: 10.1145/2507157.2507191.

Alejandro Bellogín, Pablo Castells and Iván Cantador. "Neighbor Selection and Weighting in User-Based Collaborative Filtering: A Performance Prediction Approach." In: *ACM Transactions on the Web* 8.2 (2014), 12:1–12:30. DOI: 10.1145/2579993.

Alejandro Bellogín, Alan Said and Arjen P. de Vries. "The Magic Barrier of Recommender Systems – No Magic, Just Ratings." In: *Proceedings of the 22nd International Conference on User Modeling, Adaptation, and Personalization*. UMAP '. Springer, 2014, pp. 25–36. DOI: 10.1007/978-3-319-08786-3_3.

Alejandro Bellogín, Pablo Castells and Iván Cantador. "Statistical biases in Information Retrieval metrics for recommender systems." In: *Information Retrieval Journal* 20.6 (2017), pp. 606–634. DOI: 10.1007/s10791-017-9312-z.

James Bennett and Stan Lanning. "The Netflix Prize." In: *Proceedings of KDD Cup and Workshop 2007*. 2007, pp. 3–6.

Adam Berger and John Lafferty. "Information Retrieval as Statistical Translation." In: *Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '99. New York, NY, USA: ACM, 1999, pp. 222–229. DOI: 10.1145/312624.312681.

Roi Blanco and Álvaro Barreiro. "Probabilistic Document Length Priors for Language Models." In: *Proceedings of the 30th European Conference* *on IR Research*. ECIR '08. Berlin, Heidelberg: Springer, 2008, pp. 394–405. DOI: 10.1007/978-3-540-78646-7_36.

David M. Blei, Andrew Y. Ng and Michael I. Jordan. "Latent Dirichlet Allocation." In: *Journal of Machine Learning Research* 3 (2003), pp. 993–1022.

John S Breese, David Heckerman and Carl Kadie. "Empirical Analysis of Predictive Algorithms for Collaborative Filtering." In: *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*. UAI'98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 43–52.

Igo Brilhante, Jose Antonio Macedo, Franco Maria Nardini, Raffaele Perego and Chiara Renso. "Group Finder: An Item-Driven Group Formation Framework." In: *Proceedings of the 17th IEEE International Conference on Mobile Data Management*. MDM '16. IEEE, 2016, pp. 8–17. DOI: 10.1109/MDM.2016.16.

Sergey Brin and Lawrence Page. "The Anatomy of a Large-scale Hypertextual Web Search Engine." In: *Proceedings of the 7th International Conference on World Wide Web*. Vol. 30. WWW '98. Amsterdam, The Netherlands, The Netherlands: Elsevier Science Publishers B. V., 1998, pp. 107–117. DOI: 10.1016/S0169-7552(98)00110-X.

Chris Buckley and Ellen M. Voorhees. "Evaluating evaluation measure stability." In: *Proceedings of the 23rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '00. New York, NY, USA: ACM, 2000, pp. 33–40. DOI: 10.1145/345508.345543.

Chris Buckley and Ellen M. Voorhees. "Retrieval Evaluation with Incomplete Information." In: *Proceedings of the 27th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '04. New York, NY, USA: ACM, 2004, pp. 25–32. DOI: 10.1145/1008992.1009000.

Chris Buckley, Darrin Dimmick, Ian Soboroff and Ellen Voorhees. "Bias and the limits of pooling for large collections." In: *Information Retrieval* 10.6 (2007), pp. 491–508. DOI: 10.1007/s10791-007-9032-x.

Stefan Büttcher, Charles L. A. Clarke, Peter C. K. Yeung and Ian Soboroff. "Reliable Information Retrieval Evaluation with Incomplete and Biased Judgements." In: *Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '07. New York, NY, USA: ACM, 2007, pp. 63–70. DOI: 10.1145/1277741.1277755.

Pedro G. Campos, Fernando Díez and Iván Cantador. "Time-aware Recommender Systems: A Comprehensive Survey and Analysis of Existing

Evaluation Protocols." In: *User Modeling and User-Adapted Interaction* 24.1-2 (2014), pp. 67–119. DOI: 10.1007/s11257-012-9136-x.

Iván Cantador and Pablo Castells. "Group Recommender Systems: New Perspectives in the Social Web." In: *Recommender Systems for the Social Web*. Vol. 32. Springer Berlin Heidelberg, 2012, pp. 139–157. DOI: 10.1007/978-3-642-25694-3.

David Carmel and Elad Yom-Tov. "Estimating the Query Difficulty for Information Retrieval." In: *Synthesis Lectures on Information Concepts, Retrieval, and Services* 2.1 (2010), pp. 1–89. DOI: 10/dpkb6h.

Claudio Carpineto and Giovanni Romano. "A Survey of Automatic Query Expansion in Information Retrieval." In: *ACM Computing Surveys* 44.1 (2012), 1:1–1:50. DOI: 10.1145/2071389.2071390.

Claudio Carpineto, Renato de Mori, Giovanni Romano and Brigitte Bigi. "An Information-Theoretic Approach to Automatic Query Expansion." In: *ACM Transactions on Information Systems* 19.1 (2001), pp. 1–27. DOI: 10.1145/366836.366860.

Pablo Castells, Neil J. Hurley and Saúl Vargas. "Novelty and Diversity in Recommender Systems." In: *Recommender Systems Handbook*. Ed. by Francesco Ricci, Lior Rokach and Bracha Shapira. 2nd. Boston, MA, USA: Springer, 2015, pp. 881–918. DOI: 10.1007/978-1-4899-7637-6_26.

S.-J. Chen and Li Lin. "Modeling Team Member Characteristics for the Formation of a Multifunctional Team in Concurrent Engineering." In: *IEEE Transactions on Engineering Management* 51.2 (2004), pp. 111–124. DOI: 10.1109/TEM.2004.826011.

Eunjoon Cho, Seth A Myers and Jure Leskovec. "Friendship and Mobility: User Movement in Location-based Social Networks." In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '11. New York, NY, USA: ACM, 2011, pp. 1082–1090. DOI: 10.1145/2020408.2020579.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. *Introduction to Algorithms, Third Edition*. 3rd. MIT Press, 2009.

Michele Coscia, Fosca Giannotti and Dino Pedreschi. "A classification for community discovery methods in complex networks." In: *Statistical Analysis and Data Mining* 4.5 (2011), pp. 512–546. DOI: 10.1002/sam.10133.

Paolo Cremonesi, Yehuda Koren and Roberto Turrin. "Performance of Recommender Algorithms on Top-N Recommendation Tasks." In: *Proceedings of the 4th ACM Conference on Recommender Systems*. RecSys '10. New York, NY, USA: ACM, 2010, pp. 39–46. DOI: 10.1145/1864708.1864721.

W. Bruce Croft and David J. Harper. "Using Probabilistic Models of Document Retrieval Without Relevance Information." In: *Journal of Documentation* 35.4 (1979), pp. 285–295. DOI: 10.1108/eb026683.

W. Bruce Croft, Donald Metzler and Trevor Strohman. *Search Engines: Information Retrieval in Practice*. 2015.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer and Richard Harshman. "Indexing by Latent Semantic Analysis." In: *Journal of the American Society for Information Science* 41.6 (1990), pp. 391–407. DOI: 10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9.

Mukund Deshpande and George Karypis. "Item-based top-N Recommendation Algorithms." In: *ACM Transactions on Information Systems* 22.1 (2004), pp. 143–177. DOI: 10.1145/963770.963776.

Chris Ding, Tao Li, Dijun Luo and Wei Peng. "Posterior Probabilistic Clustering Using NMF." In: *Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '08. New York, NY, USA: ACM, 2008, pp. 831–832. DOI: 10.1145/1390334.1390527.

Wei Dong, Charikar Moses and Kai Li. "Efficient K-nearest Neighbor Graph Construction for Generic Similarity Measures." In: *Proceedings of the 20th International Conference on World Wide Web*. WWW '11. New York, NY, USA: ACM, 2011, pp. 577–586. DOI: 10.1145/1963405.1963487.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Monographs on Statistics and Applied Probability. Boca Raton, Florida, USA: Chapman & Hall/CRC, 1993.

Michael D. Ekstrand, Michael Ludwig, Joseph A. Konstan and John T. Riedl. "Rethinking the Recommender Research Ecosystem: Reproducibility, Openness, and LensKit." In: *Proceedings of the Fifth ACM Conference on Recommender Systems*. RecSys '11. New York, NY, USA: ACM, 2011, pp. 133–140. DOI: 10.1145/2043932.2043958.

Hui Fang, Tao Tao and ChengXiang Zhai. "A Formal Study of Information Retrieval Heuristics." In: *Proceedings of the 27th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '04. New York, NY, USA: ACM, 2004, p. 49. DOI: 10.1145/1008992.1009004.

Uriel Feige, David Peleg and Guy Kortsarz. "The Dense k -Subgraph Problem." In: *Algorithmica* 29.3 (2001), pp. 410–421. DOI: 10.1007/s004530010050.

Daniel Fleder and Kartik Hosanagar. "Blockbuster Culture's Next Rise or Fall: The Impact of Recommender Systems on Sales Diversity." In:

Management Science 55.5 (2009), pp. 697–712. DOI: 10.1287/mnsc.1080.0974.

Norbert Fuhr. “Some Common Mistakes In IR Evaluation, And How They Can Be Avoided.” In: *SIGIR Forum* 51.3 (2018), pp. 32–41. DOI: 10.1145/3190580.3190586.

Inma Garcia, Laura Sebastia and Eva Onaindia. “On the design of individual and group recommender systems for tourism.” In: *Expert Systems with Applications* 38.6 (2011), pp. 7683–7692. DOI: 10.1016/j.eswa.2010.12.143.

Marco de Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci and Giovanni Semeraro. “Semantics-Aware Content-Based Recommender Systems.” In: *Recommender Systems Handbook*. Ed. by Francesco Ricci, Lior Rokach and Bracha Shapira. 2nd. Boston, MA, USA: Springer US, 2015, pp. 119–159. DOI: 10.1007/978-1-4899-7637-6_4.

Thomas George and Srujana Merugu. “A Scalable Collaborative Filtering Framework Based on Co-Clustering.” In: *Proceedings of the Fifth IEEE International Conference on Data Mining*. ICDM ’05. Washington, DC, USA: IEEE, 2005, pp. 625–628. DOI: 10.1109/ICDM.2005.14.

Ken Goldberg, Theresa Roeder, Dhruv Gupta and Chris Perkins. “Eigentaste: A Constant Time Collaborative Filtering Algorithm.” In: *Information Retrieval* 4.2 (2001), pp. 133–151. DOI: 10.1023/A:1011419012209.

Carlos A. Gomez-Uribe and Neil Hunt. “The Netflix Recommender System: Algorithms, Business Value, and Innovation.” In: *ACM Transactions on Management Information Systems* 6.4 (2015), 13:1–13:19. DOI: 10.1145/2843948.

Asela Gunawardana and Guy Shani. “Evaluating Recommender Systems.” In: *Recommender Systems Handbook*. Ed. by Francesco Ricci, Lior Rokach and Bracha Shapira. 2nd. Boston, MA, USA: Springer, 2015, pp. 265–308. DOI: 10.1007/978-1-4899-7637-6_8.

Uri Hanani, Bracha Shapira and Peretz Shoval. “Information Filtering: Overview of Issues, Research and Systems.” In: *User Modeling and User-Adapted Interaction* 11.3 (2001), pp. 203–259. DOI: 10.1023/A:1011196000674.

Zellig S. Harris. “Distributional Structure.” In: *WORD* 10.2-3 (1954), pp. 146–162. DOI: 10.1080/00437956.1954.11659520.

Hussein Hazimeh and ChengXiang Zhai. “Axiomatic Analysis of Smoothing Methods in Language Models for Pseudo-Relevance Feedback.” In: *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*. ICTIR ’15. New York, NY, USA: ACM, 2015, pp. 141–150. DOI: 10.1145/2808194.2809471.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua. “Neural Collaborative Filtering.” In: *Proceedings of the 26th International Conference on World Wide Web - WWW ’17*. WWW ’17. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2017, pp. 173–182. DOI: 10.1145/3038912.3052569.

Jon Herlocker, Joseph A. Konstan and John T. Riedl. “An Empirical Analysis of Design Choices in Neighborhood-Based Collaborative Filtering Algorithms.” In: *Information Retrieval* 5.4 (2002), pp. 287–310. DOI: 10.1023/A:1020443909834.

Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen and John T. Riedl. “Evaluating Collaborative Filtering Recommender Systems.” In: *ACM Transactions on Information Systems* 22.1 (2004), pp. 5–53. DOI: 10.1145/963770.963772.

Arthur E. Hoerl and Robert W. Kennard. “Ridge Regression: Biased Estimation for Nonorthogonal Problems.” In: *Technometrics* 12.1 (1970), pp. 55–67. DOI: 10.1080/00401706.1970.10488634.

Thomas Hofmann. “Probabilistic latent semantic indexing.” In: *Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’99. New York, NY, USA: ACM, 1999, pp. 50–57. DOI: 10.1145/312624.312649.

Thomas Hofmann. “Latent semantic models for collaborative filtering.” In: *ACM Transactions on Information Systems* 22.1 (2004), pp. 89–115. DOI: 10.1145/963770.963774.

Kartik Hosanagar, Daniel Fleder, Dokyun Lee and Andreas Buja. “Will the Global Village Fracture into Tribes: Recommender Systems and their Effects on Consumers.” In: *Management Science* 60.4 (2014), pp. 805–823. DOI: 10.2139/ssrn.1321962.

Liang Hu, Jian Cao, Guandong Xu, Longbing Cao, Zhiping Gu and Wei Cao. “Deep Modeling of Group Preferences for Group-based Recommendation.” In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. AAAI’14. AAAI Press, 2014, pp. 1861–1867. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8458.

Yifan Hu, Yehuda Koren and Chris Volinsky. “Collaborative Filtering for Implicit Feedback Datasets.” In: *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*. ICDM ’08. Washington, DC, USA: IEEE, 2008, pp. 263–272. DOI: 10.1109/ICDM.2008.22.

Kalervo Järvelin and Jaana Kekäläinen. “Cumulated Gain-based Evaluation of IR Techniques.” In: *ACM Transactions on Information Systems* 20.4 (2002), pp. 422–446. DOI: 10.1145/582415.582418.

Frederick Jelinek and Robert L. Mercer. “Interpolated Estimation of Markov Source Parameters from Sparse Data.” In: *Proceedings of the Workshop on Pattern Recognition in Practice*. 1980, pp. 381–397.

Surya Kallumadi, Bhaskar Mitra and Tereza Iofciu. “A Line in the Sand: Recommendation or Ad-hoc Retrieval? Overview of RecSys Challenge 2018 Submission by Team BachPropagate.” In: *Proceedings of the ACM Recommender Systems Challenge 2018*. RecSys Challenge ’18. New York, NY, USA: ACM, 2018, 7:1–7:6. DOI: 10.1145/3267471.3267478.

Maryam Karimzadehgan and ChengXiang Zhai. “Estimation of Statistical Translation Models Based on Mutual Information for Ad Hoc Information Retrieval.” In: *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’10. New York, NY, USA: ACM, 2010, p. 323. DOI: 10.1145/1835449.1835505.

M. G. Kendall. “A New Measure of Rank Correlation.” In: *Biometrika* 30.1-2 (1938), pp. 81–93. DOI: 10.1093/biomet/30.1-2.81.

Allen Kent, Madeline M. Berry, Fred U. Luehrs and J. W. Perry. “Machine literature searching VIII. Operational criteria for designing information retrieval systems.” In: *American Documentation* 6.2 (1955), pp. 93–101. DOI: 10.1002/asi.5090060209.

Yehuda Koren. “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. New York, NY, USA: ACM, 2008, pp. 426–434. DOI: 10.1145/1401890.1401944.

Yehuda Koren and Robert Bell. “Advances in Collaborative Filtering.” In: *Recommender Systems Handbook*. Ed. by Francesco Ricci, Lior Rokach and Bracha Shapira. 2nd. Boston, MA, USA: Springer US, 2015, pp. 77–118. DOI: 10.1007/978-1-4899-7637-6_3.

Yehuda Koren, Robert Bell and Chris Volinsky. “Matrix Factorization Techniques for Recommender Systems.” In: *Computer* 42.8 (2009), pp. 30–37. DOI: 10.1109/MC.2009.263.

Denis Kotkov, Shuaiqiang Wang and Jari Veijalainen. “A survey of serendipity in recommender systems.” In: *Knowledge-Based Systems* 111 (2016), pp. 180–192. DOI: 10.1016/j.knosys.2016.08.014.

Wessel Kraaij, Thijs Westerveld and Djoerd Hiemstra. “The Importance of Prior Probabilities for Entry Page Search.” In: *Proceedings of the 25th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’02. New York, NY, USA: ACM, 2002, pp. 27–34. DOI: 10.1145/564376.564383.

Matevž Kunaver and Tomaž Požrl. “Diversity in recommender systems – A survey.” In: *Knowledge-Based Systems* 123 (2017), pp. 154–162. DOI: 10.1016/j.knosys.2017.02.009.

John Lafferty and ChengXiang Zhai. “Probabilistic Relevance Models Based on Document and Query Generation.” In: *Language Modeling for Information Retrieval*. Ed. by W. Bruce Croft and John Lafferty. Springer Netherlands, 2003, pp. 1–10. DOI: 10.1007/978-94-017-0171-6.

John Lafferty and Chengxiang Zhai. “Document Language Models, Query Models, and Risk Minimization for Information Retrieval.” In: *Proceedings of the 24th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’01. New York, NY, USA: ACM, 2001, pp. 111–119. DOI: 10.1145/383952.383970.

Frederick W. Lancaster and Emily G. Fayen. *Information Retrieval: Online*. Melville Publishing Company, 1973.

Alfonso Landin, Eva Suárez-García and Daniel Valcarce. “When Diversity Met Accuracy: A Story of Recommender Systems.” In: *Proceedings of XoveTIC 2018*. Vol. 2. 18. Basel, Switzerland: MDPI, 2018, p. 1178. DOI: 10.3390/proceedings2181178.

Alfonso Landin, Daniel Valcarce, Javier Parapar and Álvaro Barreiro. “PRIN: A Probabilistic Recommender with Item Priors and Neural Models.” In: *Proceedings of the 41th European Conference on Information Retrieval*. ECIR ’19. Berlin, Heidelberg: Springer, 2019.

Theodoros Lappas, Kun Liu and Evimaria Terzi. “Finding a Team of Experts in Social Networks.” In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’09. New York, NY, USA: ACM, 2009, pp. 467–476. DOI: 10.1145/1557019.1557074.

Victor Lavrenko and W. Bruce Croft. “Relevance-Based Language Models.” In: *Proceedings of the 24th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’01. New York, NY, USA: ACM, 2001, pp. 120–127. DOI: 10.1145/383952.383972.

Sebastian Lee, Daniel; Seung. “Algorithms for Non Negative Matrix Factorization.” In: *Advances in Neural Information Processing Systems 13*. NIPS ’01. MIT Press, 2001, pp. 556–562. DOI: 10.1109/ICASSP.2006.1661352.

Danielle Hyunsook Lee and Peter Brusilovsky. “Improving personalized recommendations using community membership information.” In: *Information Processing & Management* 53.5 (2017), pp. 1201–1214. DOI: 10.1016/j.ipm.2017.05.005.

Justin J Levandoski, Mohamed Sarwat, Ahmed Eldawy and Mohamed F Mokbel. "LARS: A Location-Aware Recommender System." In: *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*. ICDE '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 450–461. DOI: 10.1109/ICDE.2012.54.

Xiaoyan Li. "A New Robust Relevance Model in the Language Model Framework." In: *Information Processing & Management* 44.3 (2008), pp. 991–1007. DOI: 10.1016/j.ipm.2007.07.005.

Jimmy Lin. "The Neural Hype and Comparisons Against Weak Baselines." In: *SIGIR Forum* 52.2 (2018), pp. 40–51. URL: http://sigir.org/wp-content/uploads/2019/01/p040.pdf.

Chao Liu, Hung-chih Yang, Jinliang Fan, Li-Wei He and Yi-Min Wang. "Distributed Nonnegative Matrix Factorization for Web-scale Dyadic Data Analysis on MapReduce." In: *Proceedings of the 19th International Conference on World Wide Web*. WWW '10. New York, NY, USA: ACM, 2010, pp. 681–690. DOI: 10.1145/1772690.1772760.

Haifeng Liu, Zheng Hu, Ahmad Mian, Hui Tian and Xuzhen Zhu. "A new user similarity model to improve the accuracy of collaborative filtering." In: *Knowledge-Based Systems* 56 (2014), pp. 156–166. DOI: 10.1016/j.knosys.2013.11.006.

Weimo Liu, Weiwei Sun, Chunan Chen, Yan Huang, Yinan Jing and Kunjie Chen. "Circle of Friend Query in Geo-Social Networks." In: *Proceedings of the 17th International Conference on Database Systems for Advanced Applications - Volume Part II*. DASFAA'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 126–137. DOI: 10.1007/978-3-642-29035-0_9.

David E. Losada and Leif Azzopardi. "An Analysis on Document Length Retrieval Trends in Language Modeling Smoothing." In: *Information Retrieval* 11.2 (2008), pp. 109–138. DOI: 10.1007/s10791-007-9040-x.

David E. Losada and Leif Azzopardi. "Assessing Multivariate Bernoulli Models for Information Retrieval." In: *ACM Transactions on Information Systems* 26.3 (2008), 17:1–17:46. DOI: 10.1145/1361684.1361690.

Xiaolu Lu, Alistair Moffat and J. Shane Culpepper. "The Effect of Pooling and Evaluation Depth on IR Metrics." In: *Information Retrieval Journal* 19.4 (2016), pp. 416–445. DOI: 10.1007/s10791-016-9282-6.

Yuanhua Lv and ChengXiang Zhai. "A Comparative Study of Methods for Estimating Query Language Models with Pseudo Feedback." In: *Proceedings of the 18th ACM Conference on Information and Knowledge*

Management. CIKM ’09. New York, NY, USA: ACM, 2009, pp. 1895–1898. DOI: 10.1145/1645953.1646259.

Yuanhua Lv and ChengXiang Zhai. “Revisiting the Divergence Minimization Feedback Model.” In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. CIKM ’14. New York, NY, USA: ACM, 2014, pp. 1863–1866. DOI: 10.1145/2661829.2661900.

David J. C. MacKay and Linda C. Bauman Peto. “A hierarchical Dirichlet language model.” In: *Natural Language Engineering* 1.03 (1995), pp. 289–308. DOI: 10.1017/S1351324900000218.

Craig Macdonald, Richard McCreadie, Rodrygo L. T. Santos and Iadh Ounis. “From Puppy to Maturity: Experiences in Developing Terrier.” In: *Proceedings of the SIGIR 2012 Workshop in Open Source Information Retrieval*. 2012, pp. 60–63.

Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

Benjamin M. Marlin, Richard S. Zemel, Sam Roweis and Malcolm Slaney. “Collaborative Filtering and the Missing at Random Assumption.” In: *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*. UAI’07. Arlington, VA, US: AUAI Press, 2007, pp. 267–275.

M. E. Maron and J. L. Kuhns. “On Relevance, Probabilistic Indexing and Information Retrieval.” In: *Journal of the ACM* 7.3 (1960), pp. 216–244. DOI: 10.1145/321033.321035.

Judith Masthoff. “Group Recommender Systems: Aggregation, Satisfaction and Group Attributes.” In: *Recommender Systems Handbook*. Ed. by Francesco Ricci, Lior Rokach and Bracha Shapira. Boston, MA, USA: Springer, 2015, pp. 743–776. DOI: 10.1007/978-1-4899-7637-6_22.

Matthew R. McLaughlin and Jonathan L. Herlocker. “A Collaborative Filtering Algorithm and Evaluation Metric That Accurately Model the User Experience.” In: *Proceedings of the 27th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’04. New York, NY, USA: ACM, 2004, pp. 329–336. DOI: 10.1145/1008992.1009050.

Donald Metzler and W. Bruce Croft. “Linear Feature-Based Models for Information Retrieval.” In: *Information Retrieval* 10.3 (2007), pp. 257–274. DOI: 10.1007/s10791-006-9019-z.

Alan Miller. *Subset Selection in Regression*. 2nd. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Chapman & Hall/CRC, 2002.

Bhaskar Mitra and Nick Craswell. "An Introduction to Neural Information Retrieval." In: *Foundations and Trends® in Information Retrieval* 13.1 (2018), pp. 1–126. DOI: 10.1561/1500000061.

Calvin N. Mooers. *Making Information Retrieval Pay*. Vol. 55. Zator Company, 1951.

Raymond J. Mooney and Loriene Roy. "Content-Based Book Recommending Using Learning for Text Categorization." In: *Proceedings of the 5th ACM Conference on Digital Libraries*. DL '00. New York, NY, USA: ACM, 2000, pp. 195–204. DOI: 10.1145/336597.336662.

Hermann Ney, Ute Essen and Reinhard Kneser. "On Structuring Probabilistic Dependences in Stochastic Language Modelling." In: *Computer Speech & Language* 8.1 (1994), pp. 1–38. DOI: 10.1006/csla.1994.1001.

Xia Ning and George Karypis. "SLIM: Sparse Linear Methods for Top-N Recommender Systems." In: *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*. ICDM '11. Washington, DC, USA: IEEE, 2011, pp. 497–506. DOI: 10.1109/ICDM.2011.134.

Xia Ning, Christian Desrosiers and George Karypis. "A Comprehensive Survey of Neighborhood-Based Recommendation Methods." In: *Recommender Systems Handbook*. Ed. by Francesco Ricci, Lior Rokach and Bracha Shapira. 2nd. Boston, MA, USA: Springer US, 2015, pp. 37–76. DOI: 10.1007/978-1-4899-7637-6_2.

Javier Parapar. "Relevance-based language models: new estimations and applications." PhD thesis. Universidade da Coruña, 2013. URL: https://hdl.handle.net/2183/10332.

Javier Parapar and Álvaro Barreiro. "Promoting Divergent Terms in the Estimation of Relevance Models." In: *Proceedings of the 3rd International Conference on Advances in Information Retrieval Theory*. ICTIR '11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 77–88. DOI: 10.1007/978-3-642-23318-0_9.

Javier Parapar, Alejandro Bellogín, Pablo Castells and Álvaro Barreiro. "Relevance-based language modelling for recommender systems." In: *Information Processing and Management* 49.4 (2013), pp. 966–980. DOI: 10.1016/j.ipm.2013.03.001.

Yoon-Joo Park and Alexander Tuzhilin. "The long tail of recommender systems and how to leverage it." In: *Proceedings of the 2008 ACM Conference on Recommender Systems*. RecSys '08. New York, NY, USA: ACM, 2008, p. 11. DOI: 10.1145/1454008.1454012.

Jie Peng and Iadh Ounis. "Combination of Document Priors in Web Information Retrieval." In: *Proceedings of the 29th European Conference* *on IR Research*. ECIR '07. Berlin, Heidelberg: Springer, 2007, pp. 732–736. DOI: 10.1007/978-3-540-71496-5_80.

Georgios Pitsilis, Xiangliang Zhang and Wei Wang. "Clustering Recommenders in Collaborative Filtering Using Explicit Trust Information." In: *Proceedings of the 5th IFIP International Conference on Trust Management*. IFIPTM '11. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 82–97. DOI: 10.1007/978-3-642-22200-9_9.

Jay M. Ponte and W. Bruce Croft. "A Language Modeling Approach to Information Retrieval." In: *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '98. New York, NY, USA: ACM, 1998, pp. 275–281. DOI: 10.1145/290941.291008.

Karthik Raman, Raghavendra Udupa, Pushpak Bhattacharya and Abhijit Bhole. "On Improving Pseudo-Relevance Feedback Using Pseudo-Irrelevant Documents." In: *Proceedings of the 32nd European Conference on Advances in Information Retrieval*. ECIR '10. Berlin, Heidelberg: Springer, 2010, pp. 573–576. DOI: 10.1007/978-3-642-12275-0_50.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner and Lars Schmidt-Thieme. "BPR: Bayesian Personalized Ranking from Implicit Feedback." In: *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. UAI '09. Arlington, VA, US: AUAI Press, 2009, pp. 452–461.

Paul Resnick and Hal R. Varian. "Recommender systems." In: *Communications of the ACM* 40.3 (1997), pp. 56–58. DOI: 10.1145/245108.245121.

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom and John T. Riedl. "GroupLens: An Open Architecture for Collaborative Filtering of Netnews." In: *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*. CSCW '94. New York, NY, USA: ACM, 1994, pp. 175–186. DOI: 10.1145/192844.192905.

Francesco Ricci, Lior Rokach and Bracha Shapira. *Recommender Systems Handbook*. Ed. by Francesco Ricci, Lior Rokach and Bracha Shapira. 2nd. Boston, MA, USA: Springer, 2015. DOI: 10.1007/978-1-4899-7637-6.

Stephen E. Robertson. "The Probability Ranking Principle in IR." In: *Journal of Documentation* 33.4 (1977), pp. 294–304. DOI: 10.1108/eb026647.

Stephen E. Robertson. "On term selection for query expansion." In: *Journal of Documentation* 46.4 (1990), pp. 359–364. DOI: 10.1108/eb026866.

Stephen E. Robertson. “Understanding inverse document frequency: on theoretical arguments for IDF.” In: *Journal of Documentation* 60.5 (2004), pp. 503–520. DOI: 10.1108/00220410410560582.

Stephen E. Robertson, Karen Spärck Jones and K. Sparck Jones. “Relevance Weighting of Search Terms.” In: *Journal of the American Society for Information Science* 27.3 (1976), pp. 129–146. DOI: 10.1002/asi.4630270302.

Joseph J. Rocchio. “Relevance Feedback in Information Retrieval.” In: *The SMART Retrieval System - Experiments in Automatic Document Processing*. Ed. by Gerard Salton. Englewood Cliffs, NJ, USA: Prentice Hall, 1971, pp. 313–323. URL: http://sigir.org/files/museum/pub-08/XXIII-1.pdf.

Ian Ruthven and Mounia Lalmas. “A Survey on the Use of Relevance Feedback for Information Access Systems.” In: *The Knowledge Engineering Review* 18.2 (2003), pp. 95–145. DOI: 10.1017/S0269888903000638.

Alan Said, Ben Fields, Brijnesh J. Jain and Sahin Albayrak. “User-centric Evaluation of a K-furthest Neighbor Collaborative Filtering Recommender Algorithm.” In: *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*. CSCW ’13. New York, NY, USA: ACM, 2013, pp. 1399–1408. DOI: 10.1145/2441776.2441933.

Tetsuya Sakai. “Evaluating Evaluation Metrics Based on the Bootstrap.” In: *Proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’06. New York, NY, USA: ACM, 2006, pp. 525–532. DOI: 10.1145/1148170.1148261.

Tetsuya Sakai and Noriko Kando. “On Information Retrieval Metrics Designed for Evaluation with Incomplete Relevance Assessments.” In: *Information Retrieval* 11.5 (2008), pp. 447–470. DOI: 10.1007/s10791-008-9059-7.

Tetsuya Sakai, Toshihiko Manabe and Makoto Koyama. “Flexible Pseudo-Relevance Feedback via Selective Sampling.” In: *ACM Transactions on Asian Language Information Processing* 4.2 (2005), pp. 111–135. DOI: 10.1145/1105696.1105699.

Gerard Salton. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1971.

Gerard Salton, A. Wong and C. S. Yang. “A Vector Space Model for Automatic Indexing.” In: *Communications of the ACM* 18.11 (1975), pp. 613–620. DOI: 10.1145/361219.361220.

Gerard Salton, Edward A. Fox and Harry Wu. "Extended Boolean information retrieval." In: *Communications of the ACM* 26.11 (1983), pp. 1022–1036. DOI: 10.1145/182.358466.

Badrul M. Sarwar, George Karypis, Joseph A. Konstan and John T. Riedl. "Analysis of Recommendation Algorithms for e-Commerce." In: *Proceedings of the 2nd ACM Conference on Electronic Commerce*. EC '00. New York, NY, USA: ACM, 2000, pp. 158–167. DOI: 10.1145/352871.352887.

Badrul M. Sarwar, George Karypis, Joseph A. Konstan and John T. Riedl. "Item-based Collaborative Filtering Recommendation Algorithms." In: *Proceedings of the 10th International Conference on World Wide Web*. WWW '01. New York, NY, USA: ACM, 2001, pp. 285–295. DOI: 10.1145/371920.372071.

Badrul M. Sarwar, George Karypis, Joseph A. Konstan and John T. Riedl. "Recommender systems for large-scale e-commerce: Scalable neighborhood ..." In: *Proceedings of the 5th International Conference on Computer and Information Technology*. ICCIT '02. 2002.

Mohamed Sarwat, Justin J Levandoski, Ahmed Eldawy and Mohamed F Mokbel. "LARS*: An Efficient and Scalable Location-Aware Recommender System." In: *IEEE Transactions on Knowledge and Data Engineering* 26.6 (2014), pp. 1384–1399. DOI: 10.1109/TKDE.2013.29.

Upendra Shardanand and Pattie Maes. "Social information filtering: Algorithms for Automating "Word of Mouth"." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '95. New York, NY, USA: ACM, 1995, pp. 210–217. DOI: 10.1145/223904.223931.

Amit Sharma, Jake M Hofman and Duncan J Watts. "Estimating the Causal Impact of Recommendation Systems from Observational Data." In: *Proceedings of the Sixteenth ACM Conference on Economics and Computation*. EC '15. New York, NY, USA: ACM, 2015, pp. 453–470. DOI: 10.1145/2764468.2764488.

Amit Singhal, Chris Buckley and Mandar Mitra. "Pivoted Document Length Normalization." In: *Proceedings of the 19th International ACM SIGIR Conference on Research and development in Information Retrieval*. SIGIR '96. New York, NY, USA: ACM, 1996, pp. 21–29. DOI: 10.1145/243199.243206.

Mauro Sozio and Aristides Gionis. "The Community-search Problem and How to Plan a Successful Cocktail Party." In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '10. New York, NY, USA: ACM, 2010, pp. 939–948. DOI: 10.1145/1835804.1835923.

Karen Spärck Jones. "A Statistical Interpretation of Term Specificity and its Retrieval." In: *Journal of Documentation* 28.1 (1972), pp. 11–21. DOI: 10.1108/eb026526.

Karen Spärck Jones and Cornelis J. Van Rijsbergen. *Report on the Need for and Provision of an Ideal Information Retrieval Test Collection*. Tech. rep. University of Cambridge, 1975.

Karen Spärck Jones, Stephen Walker and Stephen E. Robertson. "A probabilistic model of information retrieval: development and comparative experiments: Part 1." In: *Information Processing & Management* 36.6 (2000), pp. 779–808. DOI: 10.1016/S0306-4573(00)00015-7.

Karen Spärck Jones, Stephen Walker and Stephen E. Robertson. "A probabilistic model of information retrieval: development and comparative experiments: Part 2." In: *Information Processing & Management* 36.6 (2000), pp. 809–840. DOI: 10.1016/S0306-4573(00)00016-9.

Harald Steck. "Training and Testing of Recommender Systems on Data Missing Not at Random." In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '10. New York, NY, USA: ACM, 2010, pp. 713–722. DOI: 10.1145/1835804.1835895.

Harald Steck. "Evaluation of Recommendations: Rating-prediction and Ranking." In: *Proceedings of the 7th ACM Conference on Recommender Systems*. RecSys '13. New York, NY, USA: ACM, 2013, pp. 213–220. DOI: 10.1145/2507157.2507160.

Xiaoyuan Su and Taghi M. Khoshgoftaar. "Collaborative Filtering Recommender Systems." In: *Advances in Artificial Intelligence* 2009 (2009), pp. 1–19. DOI: 10.1561/1100000009.

Eva Suárez-García, Alfonso Landin, Daniel Valcarce and Álvaro Barreiro. "Term Association Measures for Memory-based Recommender Systems." In: *Proceedings of the 5th Spanish Conference on Information Retrieval*. Vol. 18. CERI '18. New York, NY, USA: ACM, 2018, pp. 1–8. DOI: 10.1145/3230599.3230606.

Gábor Takács, István Pilászy, Bottyán Németh and Domonkos Tikk. "Scalable Collaborative Filtering Approaches for Large Recommender Systems." In: *Journal of Machine Learning Research* 10 (2009), pp. 623–656. DOI: 10.1145/1577069.1577091.

Tao Tao and ChengXiang Zhai. "Regularized Estimation of Mixture Models for Robust Pseudo-relevance Feedback." In: *Proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '06. New York, NY, USA: ACM, 2006, pp. 162–169. DOI: 10.1145/1148170.1148201.

Robert Tibshirani. “Regression Shrinkage and Selection via the Lasso.” In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996), pp. 267–288. URL: http://www.jstor.org/stable/2346178.

Nava Tintarev and Judith Masthoff. “Explaining Recommendations: Design and Evaluation.” In: *Recommender Systems Handbook*. Ed. by Francesco Ricci, Lior Rokach and Bracha Shapira. Boston, MA, USA: Springer, 2015, pp. 353–382. DOI: 10.1007/978-1-4899-7637-6_10.

KhanhQuan Truong, Fuyuki Ishikawa and Ishikawa Honided. “Improving Accuracy of Recommender System by Item Clustering.” In: *IEICE Transactions on Information and Systems* E90-D.9 (2007), pp. 1363–1373. DOI: 10.1093/ietisy/e90-d.9.1363.

Daniel Valcarce. “Exploring Statistical Language Models for Recommender Systems.” In: *Proceedings of the 9th ACM Conference on Recommender Systems*. RecSys ’15. ACM, 2015, pp. 375–378. DOI: 10.1145/2792838.2796547.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. “When Recommenders Met Big Data: An Architectural Proposal and Evaluation.” In: *Proceedings of the 3rd Spanish Conference on Information Retrieval*. CERI ’14. 2014, pp. 73–84.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. “A Distributed Recommendation Platform for Big Data.” In: *Journal of Universal Computer Science* 21.13 (2015), pp. 1810–1829. DOI: 10.3217/jucs-021-13-1810.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. “A Study of Priors for Relevance-Based Language Modelling of Recommender Systems.” In: *Proceedings of the 9th ACM Conference on Recommender Systems*. RecSys ’15. New York, NY, USA: ACM, 2015, pp. 237–240. DOI: 10.1145/2792838.2799677.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. “A Study of Smoothing Methods for Relevance-Based Language Modelling of Recommender Systems.” In: *Proceedings of the 37th European Conference on Information Retrieval*. Vol. 9022. ECIR ’15. Berlin, Heidelberg: Springer, 2015, pp. 346–351. DOI: 10.1007/978-3-319-16354-3_38.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. “Additive Smoothing for Relevance-Based Language Modelling of Recommender Systems.” In: *Proceedings of the 4th Spanish Conference on Information Retrieval*. CERI ’16. New York, NY, USA: ACM, 2016, pp. 1–8. DOI: 10.1145/2934732.2934737.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. “Computing Neighbourhoods with Language Models in a Collaborative Filtering Scenario." In: *Proceedings of the 7th Italian Information Retrieval Workshop*. Vol. 1653. IIR '16. CEUR Workshop Proceedings, 2016. URL: http://ceur-ws.org/Vol-1653/paper_13.pdf.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. "Efficient Pseudo-Relevance Feedback Methods for Collaborative Filtering Recommendation." In: *Proceedings of the 38th European Conference on Information Retrieval*. ECIR '16. Berlin, Heidelberg: Springer, 2016, pp. 602–613. DOI: 10.1007/978-3-319-30671-1_44.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. "Item-based Relevance Modelling of Recommendations for Getting Rid of Long Tail Products." In: *Knowledge-Based Systems* 103 (2016), pp. 41–51. DOI: 10.1016/j.knosys.2016.03.021.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. "Language Models for Collaborative Filtering Neighbourhoods." In: *Proceedings of the 38th European Conference on Information Retrieval*. ECIR '16. Berlin, Heidelberg: Springer, 2016, pp. 614–625. DOI: 10.1007/978-3-319-30671-1_45.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. "Axiomatic Analysis of Language Modelling of Recommender Systems." In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 25.Suppl. 2 (2017), pp. 113–127. DOI: 10.1142/S0218488517400141.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. "Combining Top-N Recommenders with Metasearch Algorithms." In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '17. New York, NY, USA: ACM, 2017, pp. 805–808. DOI: 10.1145/3077136.3080647.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. "A MapReduce implementation of posterior probability clustering and relevance models for recommendation." In: *Engineering Applications of Artificial Intelligence* 75 (2018), pp. 114–124. DOI: 10.1016/j.engappai.2018.08.006.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. "Document-based and Term-based Linear Methods for Pseudo-Relevance Feedback." In: *ACM SIGAPP Applied Computing Review* 18.4 (2018), pp. 5–17. URL: https://www.sigapp.org/acr/Issues/V18.4/ACR%2018-4.pdf.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. "Finding and analysing good neighbourhoods to improve collaborative filtering." In: *Knowledge-Based Systems* 159 (2018), pp. 193–202. DOI: 10.1016/j.knosys.2018.06.030.

Daniel Valcarce, Igo Brilhante, Jose Antonio Macedo, Franco Maria Nardini, Raffaele Perego and Chiara Renso. "Item-driven group formation." In: *Online Social Networks and Media* 8 (2018), pp. 17–31. DOI: 10.1016/j.osnem.2018.10.002.

Daniel Valcarce, Javier Parapar and Álvaro Barreiro. "LiMe: Linear Methods for Pseudo-Relevance Feedback." In: *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. SAC '18. New York, NY, USA: ACM, 2018, pp. 678–687. DOI: 10.1145/3167132.3167207.

Daniel Valcarce, Alejandro Bellogín, Javier Parapar and Pablo Castells. "On the robustness and discriminative power of information retrieval metrics for top-N recommendation." In: *Proceedings of the 12th ACM Conference on Recommender Systems*. RecSys '18. New York, NY, USA: ACM, 2018, pp. 260–268. DOI: 10.1145/3240323.3240347.

Saúl Vargas and Pablo Castells. "Improving Sales Diversity by Recommending Users to Items." In: *Proceedings of the 8th ACM Conference on Recommender Systems*. RecSys '14. New York, NY, USA: ACM, 2014, pp. 145–152. DOI: 10.1145/2645710.2645744.

Ellen M. Voorhees. "Evaluation by Highly Relevant Documents." In: *Proceedings of the 24th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '01. New York, NY, USA: ACM, 2001, pp. 74–82. DOI: 10.1145/383952.383963.

Ellen M. Voorhees. "The Philosophy of Information Retrieval Evaluation." In: *Evaluation of Cross-Language Information Retrieval Systems: Second Workshop of the Cross-Language Evaluation Forum*. CLEF 2001. Berlin, Heidelberg: Springer, 2002, pp. 355–370. DOI: 10.1007/3-540-45691-0_34.

Jun Wang. "Language Models of Collaborative Filtering." In: *Proceedings of the 5th Asia Information Retrieval Symposium on Information Retrieval Technology*. Vol. 5839. AIRS '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 218–229. DOI: 10.1007/978-3-642-04769-5_19.

Jun Wang, Arjen P. de Vries and Marcel J. T. Reinders. "A User-Item Relevance Model for Log-Based Collaborative Filtering." In: *Proceedings of the 28th European Conference on IR Research*. Vol. 3936. ECIR '06. London, UK: Springer-Verlag, 2006, pp. 37–48. DOI: 10.1007/11735106_5.

Jun Wang, Arjen P. de Vries and Marcel J. T. Reinders. "Unified relevance models for rating prediction in collaborative filtering." In: *ACM Transactions on Information Systems* 26.3 (2008), pp. 1–42. DOI: 10.1145/1361684.1361689.

S Wang, Z Zhao and X Hong. "The Research on Collaborative Filtering Recommendation Algorithm Based on Improved Clustering Processing." In: *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; De-*

pendable, Autonomic and Secure Computing; Pervasive Intelligence and Computing. CIT/IUCC/DASC/PICOM '15. IEEE, 2015, pp. 1012–1015. DOI: 10.1109/CIT/IUCC/DASC/PICOM.2015.153.

Xing Wei and W. Bruce Croft. "LDA-based Document Models for Ad-hoc Retrieval." In: *Proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '06. New York, NY, USA: ACM, 2006, pp. 178–185. DOI: 10.1145/1148170.1148204.

William J. Welch. "Algorithmic complexity: three NP - hard problems in computational statistics." In: *Journal of Statistical Computation and Simulation* 15.1 (1982), pp. 17–25. DOI: 10.1080/00949658208810560.

Yang Xu, Gareth J.F. Jones and Bin Wang. "Query Dependent Pseudo-Relevance Feedback Based on Wikipedia." In: *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '09. New York, NY, USA: ACM, 2009, p. 59. DOI: 10.1145/1571941.1571954.

Gui-Rong Xue, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu and Zheng Chen. "Scalable Collaborative Filtering Using Cluster-based Smoothing." In: *Proceedings of the 28th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '05. New York, NY, USA: ACM, 2005, pp. 114–121. DOI: 10.1145/1076034.1076056.

De-Nian Yang, Chih-Ya Shen, Wang-Chien Lee and Ming-Syan Chen. "On socio-spatial group query for location-based social networks." In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '12. New York, NY, USA: ACM, 2012, p. 949. DOI: 10.1145/2339530.2339679.

Emine Yilmaz and Javed A. Aslam. "Estimating Average Precision when Judgments are Incomplete." In: *Knowledge and Information Systems* 16.2 (2008), pp. 173–211. DOI: 10.1007/s10115-007-0101-7.

Hongzhi Yin, Bin Cui, Jing Li, Junjie Yao and Chen Chen. "Challenging the Long Tail Recommendation." In: *Proceedings of the VLDB Endowment* 5.9 (2012), pp. 896–907. DOI: 10.14778/2311906.2311916.

Hamed Zamani, Javid Dadashkarimi, Azadeh Shakery and W. Bruce Croft. "Pseudo-Relevance Feedback Based on Matrix Factorization." In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. CIKM '16. New York, NY, USA: ACM, 2016, pp. 1483–1492. DOI: 10.1145/2983323.2983844.

ChengXiang Zhai. "Statistical Language Models for Information Retrieval." In: *Synthesis Lectures on Human Language Technologies* 1.1 (2008), pp. 1–141. DOI: 10.2200/S00158ED1V01Y200811HLT001.

ChengXiang Zhai and John Lafferty. "Model-based Feedback in the Language Modeling Approach to Information Retrieval." In: *Proceedings of the 10th International Conference on Information and Knowledge Management*. CIKM '01. New York, NY, USA: ACM, 2001, p. 403. DOI: 10.1145/502585.502654.

ChengXiang Zhai and John Lafferty. "A Study of Smoothing Methods for Language Models Applied to Information Retrieval." In: *ACM Transactions on Information Systems* 22.2 (2004), pp. 179–214. DOI: 10.1145/984321.984322.

ChengXiang Zhai and Sean Massung. *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*. New York, NY, USA: ACM, 2016. DOI: 10.1145/2915031.

Renjie Zhou, Samamon Khemmarat and Lixin Gao. "The Impact of YouTube Recommendation System on Video Views." In: *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*. IMC '10. New York, NY, USA: ACM, 2010, pp. 404–410. DOI: 10.1145/1879141.1879193.

Tao Zhou, Zoltán Kuscsik, J.-G. Liu, Matús Medo, Joseph Rushton Wakeling and Y.-C. Zhang. "Solving the apparent diversity-accuracy dilemma of recommender systems." In: *Proceedings of the National Academy of Sciences* 107.10 (2010), pp. 4511–4515. DOI: 10.1073/pnas.1000488107.

Hui Zou and Trevor Hastie. "Regularization and Variable Selection via the Elastic Net." In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2 (2005), pp. 301–320. DOI: 10.1111/j.1467-9868.2005.00503.x.

INDEX