FILTRAGEM COLABORATIVA APRIMORADA: EXPLORANDO SIMILARIDADES.

RAMON PEREIRA. LOPES

FILTRAGEM COLABORATIVA APRIMORADA: EXPLORANDO SIMILARIDADES.

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Orientador: Renato Martins Assunção
Coorientador: Rodrygo Luis Teodoro Santos

Belo Horizonte
Maio de 2017

RAMON PEREIRA. LOPES

SIMILARITY-ENHANCED COLLABORATIVE FILTERING.

Thesis presented to the Graduate Program in Computer Science of the Universidade Federal de Minas Gerais in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Advisor: Renato Martins Assunção
Co-Advisor: Rodrygo Luis Teodoro Santos

Belo Horizonte
May 2017

Lopes, Ramon Pereira.

L864s Similarity-enhanced Collaborative Filtering. / Ramon Pereira. Lopes. — Belo Horizonte, 2017
xxiv, 77 f. : il. ; 29cm

Tese (doutorado) — Universidade Federal de Minas Gerais – Departamento de Ciência da Computação

Orientador: Renato Martins Assunção
Coorientador: Rodrygo Luis Teodoro Santos

1. Computação – Teses. 2. Sistemas de recomendação. 3. Filtragem colaborativa. 4. Teoria dos grafos. 5. Teoria bayesiana de decisão estatística.
I. Orientador. II. Coorientador. III. Título.

CDU 519.6*73(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Similarity-enhanced collaborative filtering

RAMON PEREIRA LOPES

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. RENATO MARTINS ASSUNÇÃO - Orientador
Departamento de Ciência da Computação - UFMG

PROF. RODRYGO LUIS TEODORO SANTOS - Coorientador
Departamento de Ciência da Computação - UFMG

PROF. MARCOS ANDRÉ GONÇALVES
Departamento de Ciência da Computação - UFMG

PROF. FLÁVIO VINICIUS DINIZ DE FIGUEIREDO
Departamento de Ciência da Computação - UFMG

PROF. EDLENO SILVA DE MOURA
Departamento de Ciência da Computação - UFAM

PROF. LEANDRO BALBY MARINHO
Departamento de Sistemas e Computação - UFCG

PROF. MARCELO GARCIA MANZATO
Departamento de Ciências da Computação - USP

Belo Horizonte, 09 de maio de 2017.

Acknowledgments

Aos meus orientadores, Profs. Renato Assunção e Rodrygo Santos, na falta de palavras para expressar minha profunda e eterna gratidão, recorro à singeleza de um muito obrigado.

Aos Profs. Euclimar e Ninfa, agradeço pelo financiamento de meus estudos pelos sete primeiros anos de minha vida escolar. Esse ato de generosidade foi o ponto germinal para esta tese de doutorado.

Aos Profs. Alexandre Salles, Flávio Assis e Thiago Noronha, agradeço pelos ensinamentos e orientação nas primeiras etapas dessa longa caminhada acadêmica.

À minha família, Edna, Larissa e Raimundo, agradeço pelo incentivo e apoio incondicional ao longo dessa jornada.

À Patrícia, agradeço por todo açúcar e afeto, além das boas risadas.

Aos amigos, agradeço pelo apoio nas horas de dificuldade.

Por fim, agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) por ter proporcionado apoio financeiro, e todos aqueles que direta ou indiretamente contribuíram para o desenvolvimento deste trabalho.

"And now that you don't have to be perfect, you can be good."
(John Steinbeck)

Resumo

Fatoração de Matrizes (FM) é a técnica de recomendação mais efetiva para filtragem colaborativa. Contudo, o estado-da-arte em FM geralmente requer dados adicionais, os quais podem estar nem sempre disponíveis. Se por um lado modelos de recomendação baseados em grafos surgiram recentemente como uma alternativa no contexto de fitragem colaborativa, por outro lado os modelos mais efetivos ignoram tanto o conjunto de notas quanto qualquer noção de confiabilidade nos caminhos que conectam usuários e itens. Neste trabalho, exploramos dados colaborativos com o objetivo de prover recomendações mais precisas nos contextos de FM e de recomendação baseada em grafos. Para esse fim, propomos abordagens baseadas em grafos e três funções de pontuação baseadas no paradigma Bayesiano, e um novo modelo Bayesiano para FM. Nossa abordagem baseada em grafos explora caminhos de comprimento três a partir do usuário alvo no grafo de relacionamentos entre usuários e itens, enquanto as funções de pontuação exploram aspectos distribucionais das notas dadas pelos usuários a fim de extrair informações latentes. Nossa abordagem baseada em FM, por sua vez, explora similaridades tanto entre usuários quanto entre itens, de modo que as similaridades são computadas a partir da matriz de interação entre usuários e itens. Avaliamos o desempenho de nossos métodos em várias coleções de teste disponíveis publicamente e comparamos com outras abordagens da literatura sob um conjunto de métricas. Os resultados mostram que os nossos métodos superam aqueles comparados em vários cenários.

Palavras-chave: Sistemas de Recomendação, Filtragem Colaborativa, Grafos, Fatoração de Matrizes, Estatística Bayesiana.

Abstract

Matrix factorization (MF) is the most successful recommendation approach in the context of Collaborative Filtering (CF). However, existing MF approaches usually demand external data for improved regularization, which limit its applicability as external data are not always available. On the one hand graph-based models have recently emerged as an alternative recommendation approach, on the other hand state-of-the-art graph-based approaches ignore the set of ratings and the reliability of the path connecting users and items. In this dissertation, we exploit collaborative data to improve MF and graph-based recommendation. To be precise, we propose a computationally efficient graph-based approach to CF and three scoring functions based on the Bayesian paradigm, and a novel MF approach. Our graph-based approach exploits three-step paths starting from the target user in the user-item bipartite graph and relies on the scoring functions, which exploit distributional aspects of the ratings given by users to extract latent information. Our MF approach, in turn, exploits both user-user and item-item similarities extracted from the raw user-item rating matrix. We experiment with several publicly available datasets against state-of-the-art baselines. The results attest the value of our methods as they provide significant gains in several settings.

Palavras-chave: Recommender Systems, Collaborative filtering, Graph-based recommendation, Matrix Factorization, Bayesian statistics.

List of Figures

List of Tables

Acronym List

AP	Average Precision
BIS	Binary Inequality Scoring
BJS	Binary Joint Scoring
BORS	Binary Odds Ratio Scoring
BPRMF	Bayesian Personalized Ranking Matrix Factorization
BX	BookCrossing
CF	Collaborative Filtering
CIS	Categorical Inequality Scoring
MAP	Mean Average Precision
MF	Matrix Factorization
ML	Movie Lens
MP	Most Popular
MRR	Mean Reciprocal Rank
nDCG	Normalized Discounted Cumulative Gain
P	Precision
PMF	Probabilistic Matrix Factorization
RR	Reciprocal Rank
RRW	Rating-based Random Walk
RS	Recommender Systems

SGD Stochastic Gradient Descent

SMF Similarity-based Matrix Factorization

SVD Singular Value Decomposition

WRMF Weighted Regularized Matrix Factorization

Contents

Chapter 1

Introduction

In this chapter, we provide a brief introduction to the problem addressed in this dissertation. First, we motivate recommender systems and contextualize some state-of-the-art approaches. Next, we discuss some motivations for this dissertation. Finally, we state our hypothesis and summarize our contributions.

1.1 Preliminaries

The emergence of e-commerce platforms has drastically changed the way we shop. First, companies provide customers with a myriad of products and delivery services by means of 24-7 e-commerce sites. Second, shoppers are able to browse a great range of products, search for more specific goods, compare prices and features more easily, and leave reviews on products just by means of a device connected to the Internet wherever they are and whenever they want. On the one hand this scenario encourages people to purchase more products, on the other hand the convenience and availability of products increase the amount of information shoppers must process before they make up their minds. For instance, the Amazon's Web site[1] provides more than 1.8 million books in the "Business & Money" section. As a result, users usually struggle to find the most appropriate items from the immense variety of items available at these e-commerce Web sites [Ricci et al., 2011].

Das et al. [2007] point out users usually do not even know what they want so that they rely on the Web site to present them something that fulfills their needs. For instance, a user ends up looking around for movies that might interest her instead of browsing for a particular movie she wants to watch. Furthermore, Ricci et al. [2011]

[1] www.amazon.com

advocate the great amount of products in e-commerce services overwhelms users, which turns out to lead them to make poor decisions and decrease their well-being. In this context, the use of Recommender Systems (RS) has arisen as an effective solution not only to the information overload problem but also to the improvement of merchant revenue [Jannach et al., 2010; Schafer et al., 1999]. Indeed, these systems provide suggestions for users with the aim at helping in various decision-making processes, such as what items to buy, what music to listen, or what news to read [Ricci et al., 2011].

Schafer et al. [1999] report on several industrial applications of RS technology in e-commerce in the end of 1990's. In reality, RS help e-commerce increase profits in three ways [Schafer et al., 2001]: i) converting browsers into buyers, ii) suggesting additional products for the customer to purchase based on those products already in the shopping cart, and iii) improving customers' loyalty in that customers tend to return to the sites that match their needs. Thus, RS play a fundamental role in marketing activities of e-commerce and have become largely utilized in multiple business niches. In fact, the use of RS has exploded over the last decade, where major companies made use of RS within their services [Jannach et al., 2016]. Currently, the use of RS is so pervasive that encompasses plenty of applications, including [Ricci et al., 2011]: i) entertainment: recommendations of movies, music and friends in social networks, ii) content: personalized news, articles and documents, iii) e-commerce: recommendations for consumers of products in online shopping websites, and iv) services: recommendations of travel services and houses to rent.

Schafer et al. [2001] define RS as specialized data mining systems designed to take advantage of the real-time personalization opportunities of interactive e-commerce. Ricci et al. [2011], in turn, define RS as software tools and techniques providing suggestions for items to be of use to a user. Independently of the definition, the main objective of RS is to guide users in a personalized way to interesting products to maximize users' satisfaction. For instance, Netflix reports 75% of what their consumers watch come from some sort of recommendation algorithms.[2] To this end, these systems learn from customers, compute recommendations using proper techniques and finally present the recommended products that the shopper will probably find most valuable among those available [Wei et al., 2007]. In fact, the products are usually recommended based on four types of user data [Ricci et al., 2011]: i) data about users and available items, ii) rating or reviews (explicit feedback), iii) behavior (implicit feedback), and iv) transaction.

[2]http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html

The forms of recommendation usually include suggesting products to the consumer, providing personalized product information, summarizing community opinion, and providing community critiques [Schafer et al., 2001]. Figure 1.1 shows a recommendation list for a Netflix user based on the fact the user provided a feedback to the system, namely, the user has watched "Paris, Texas". Thus, RS are able not only to recommend movies according to the user's tastes but also to provide explainable recommendations.

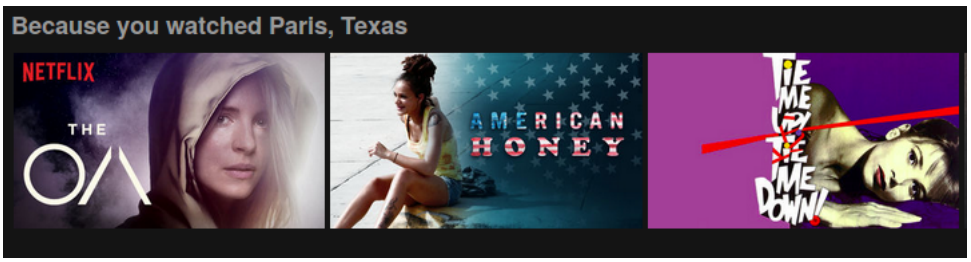


Figure 1.1: Recommendation list for a Netflix user.

RS allow personalization for each customer, where users are presented to items according to their tastes. To this end, the recommender acquires and analyzes users' data, builds a model of consumer behavior, and makes use of algorithms to produce recommendations. In this process, computing recommendations via proper techniques plays a significant role in the quality of the recommendation result [Wei et al., 2007]. Thus, many different approaches have been applied to the problem of making recommendations that fit the users' tastes [Schafer et al., 2001]. For instance, the earliest RS [Resnick et al., 1994; Shardanand and Maes, 1995] make use of nearest-neighbors, where recommendations for a target user are based on similar customers with respect to their preference histories. While the earliest RS dates back to 1990's, RS have become subject of intense research due to the Netflix Prize[3], which began in 2006.

The Netflix Prize was a competition held by Netflix where 51,051 contestants on 41,305 teams from 186 different countries competed for the grand prize of U$1,000,000. Netlifx relied on its own recommendation system Cinematch to predict whether a user will enjoy a movie based on her history and make personalized movie recommendations. In the contest, competitors had to propose algorithms to beat Cinematch by making better predictions, where the winner's method had to show prediction accuracy at least 10% better than Cinematch. To this end, participants had access to training and qualifying test sets. The training data consisted of more than 100 million ratings from over 480 thousand randomly-chosen, anonymous customers on nearly 18 thousand movie titles. The qualifying data, in turn, contained over 2.8 million customer/movie id

[3]http://www.netflixprize.com/

pairs with rating dates but with the ratings withheld. After almost three years, Netflix announced the winner with a 10.09% improvement over Cinematch. As a result, much progress has been made in RS, particularly in Collaborative Filtering (CF) [Jannach et al., 2016].

RS fundamentally take one of two approaches, namely, Content-based Filtering or CF, or show a combination of both. Content-based Filtering, which has its roots in information retrieval and information filtering [Adomavicius and Tuzhilin, 2005; Wei et al., 2007], leverages features of items to find similar content. In fact, this sort of systems recommends items similar to those a given user has liked in the past by matching up the attributes of a user profile, in which preferences and interests are stored, with the attributes of candidate items [Lops et al., 2011]. In turn, CF, which is one of the earliest recommendation technologies [Resnick et al., 1994], makes use of the feedbacks other users have provided to recommend items the target user potentially likes best.

In memory-based CF approaches, n users are usually represented as vectors embedded in an m-dimensional vector space where each dimension corresponds to an item. Thus, data are represented as an $n \times m$ user-item matrix where rows correspond to users, columns to items and each entry usually represents a rating. As an alternative, items can be represented as vectors embedded in a n-dimensional vector space where each dimension corresponds to a user. Figure 1.2 illustrates the matrix representation for 3 users and 4 items. In reality, most users rate only a small subset of the items and the number of items is much larger than the number of users [Koren et al., 2009].

$$\begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \end{array} \begin{array}{c} \begin{array}{cccc} i_1 & i_2 & i_3 & i_4 \end{array} \\ \begin{pmatrix} 4 & 1 & - & - \\ 4 & 3 & 5 & - \\ - & 3 & - & 1 \end{pmatrix} \end{array}$$

Figure 1.2: Memory-based representation for 3 users and 4 items, where a missing entry is represented by a dash.

On the one hand, the user-item matrix provides a simple representation, on the other hand, it is computationally expensive to compute similarities between all pairs of users as the dataset size increases. In addition, Matrix Factorization (MF) approaches show limited accuracy in sparse scenarios [Liang et al., 2016]. As an attempt to overcome these issues, the user-item matrix can be regarded as an adjacency matrix of a user-item undirected bipartite graph, giving rise to so-called graph-based approaches. For instance, Figure 1.3 shows the graph representation for the matrix model illustrated in Figure 1.2. In these approaches, measures of similarity between users might

also be based on graph statistics such as commute or hitting time between nodes. For instance, users that possess the same taste will be connected by a large number of short paths [Fouss et al., 2005]. In contrast, classical measures of similarity between users in the context of CF exploit user behavior. To the best of our knowledge, Horting [Aggarwal et al., 1999] is the first graph-based approach for CF. In particular, nodes are consumers, weighted edges indicate the similarity between two consumers, and recommendations are produced by walking the graph to nearby nodes and then combining the opinions of the nearby consumers. Thus, Horting is able to explore transitive relationships that nearest neighbor algorithms do not consider [Sarwar et al., 2001].

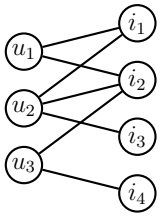


Figure 1.3: Graph-based representation for the instance illustrated in Figure 1.2.

In a seminal paper, Page et al. [1999] propose to treat an entity graph as a Markov chain whose long-term stationary distribution can be used as a global scoring. In the literature, there exist several works [Baluja et al., 2008; Christoffel et al., 2015; Cooper et al., 2014; Fouss et al., 2007, 2005; Gori and Pucci, 2007; Jamali and Ester, 2009; Lee et al., 2012; Singh et al., 2007; Xiang et al., 2010; Yildirim and Krishnamoorthy, 2008] that make use of random walks in the context of CF. To this end, transition probabilities are usually defined so that they exploit the user-item graph structure. For instance, the transition probability between two items is proportional to the number of users that rated both items [Gori and Pucci, 2007]. The authors in [Christoffel et al., 2015; Cooper et al., 2014] take a step further and show that it is possible to improve recommendation accuracy by obtaining the distribution within three or five steps instead of incurring the computational burden of obtaining the long-term stationary distribution as in [Fouss et al., 2007, 2005; Gori and Pucci, 2007; Singh et al., 2007].

Despite all of these advances, the current state-of-the-art graph-based methods require further enhancements to improve recommendation accuracy. In this context, we here propose models and algorithms for graph-based CF. To this end, we further exploit the user-item rating matrix to produce accurate recommendations. Following the state-of-the-art graph-based approaches [Christoffel et al., 2015; Cooper et al., 2014], our algorithms exploit three-step paths in the user-item graph. We evaluated

our approach in several datasets against state-of-the-art approaches, where empirical results attest the effectiveness of our algorithms.

Model-based CF approaches rely on the user-item matrix representation to produce recommendations. In fact, these models exploit collaborative data to learn a predictive model that represents latent characteristics of the users and items in the system with factors in a latent space of reduced dimensionality. In this context, MF [Koren et al., 2009] is the most prominent approach, which regularizes the latent factors through l^2-norm. On the other hand, several works [Adams et al., 2010; Agarwal and Chen, 2009; Porteous et al., 2010; Shan and Banerjee, 2010; Shi et al., 2013; Singh and Gordon, 2008; Yao et al., 2014] extend MF by further regularizing user and/or item factors when side information is available for improving recommendation accuracy. For instance, user side information may comprise age, gender and location, while item side information may include item metadata such as genre, title and reviews. Thus, these works can be regarded as hybrid methods in that they jointly use a collaborative filtering technique along with user and/or item profiles to make recommendations. Nonetheless, side information may not be available, limiting the applicability of these works.

Liang et al. [2016] extend MF by jointly factorizing both the user-item interaction matrix and item-item similarity matrix. Different from previous MF extensions, their approach, CoFactor, directly computes item-item similarities from the raw user-item interaction matrix, which makes CoFactor applicable to a wider range of scenarios, including those where no further data are available. On the other hand, CoFactor is designed for implicit feedback domains and only exploits item-item interactions, which makes room for further advances to improve recommendation accuracy. To this end, we here extend MF by further exploiting the user-item rating matrix in that we extract both user-user and item-item matrices to jointly factorize these matrices and the user-item rating matrix. Empirical results attest the effectiveness of embedding both user-user and item-item similarities into MF, where our approach provides more accurate recommendations compared to the variant that only embeds either user-user or item-item similarity matrix.

1.2 Motivation

In this section, we discuss some motivation topics for this dissertation.

1.2.1 Exploitation of the user-item matrix

Figures 1.4 and 1.5 show two similar products available in Amazon. One can note the product presented in Figure 1.4 has an average rating smaller than that displayed in Figure 1.5. However, the former has 143 times more customer reviews than the latter. Thus, we can ask: which product is more likely to maximize user's satisfaction?

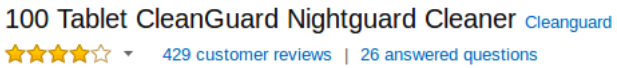


Figure 1.4: 429 customer reviews and 4 stars average rating.

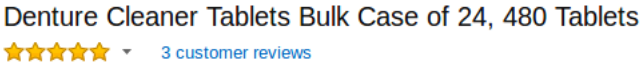


Figure 1.5: 3 customer reviews and 5 stars average rating.

The state-of-the-art graph-based methods [Christoffel et al., 2015; Cooper et al., 2014] do not exploit any information from the set of items' ratings. In fact, these methods solely make use of the set of user-item pairs to define the structure of the user-item bipartite graph. Thus, they take into account neither latent information present on the set of ratings nor similarities between users and/or between items to make recommendations. Recalling the scenario presented in Figures 1.4 and 1.5, these state-of-the-art methods treat both items the same under the light of the items' rating. Therefore, we claim graph-based approaches should exploit the user-item matrix to improve recommendation accuracy.

In the context of MF, CoFactor [Liang et al., 2016] jointly factorizes the user-item interaction matrix and item-item similarity matrix for implicit feedback settings. However, the authors do not take user-user similarities into consideration. Thus, there is room for further exploitation of the user-item interaction matrix by extracting user-user similarities and then embedding both user-user and item-item similarity matrices into MF to improve recommendation accuracy.

1.2.2 Markovian Property

Cooper et al. [2014] show the exploitation of three-length paths in the user-item bipartite graph provides better recommendations compared to those provided by five-length paths. Therefore, the state-of-the-art graph-based methods [Christoffel et al., 2015; Cooper et al., 2014] exploit three-length paths starting from the target user to reach

items not consumed by this user. In fact, any odd-length path in the user-item bipartite graph starting from a user ends up in an item.

From Figure 1.3, one can see the path $\langle u_1, i_1, u_2, i_3 \rangle$ does end in an item. In this context, item i_3 is a candidate item for recommendation as it is not consumed by the target user u_1, while item i_1 is in both users' history. Thus, we can ask how item i_3 compares to the consumed item i_1. Since users usually resort to acquitances for recommendations and the candidate item i_3 is in u_2's history, we can also ask how similar users u_1 and u_2 are with respect to their tastes.

The methods presented in [Christoffel et al., 2015; Cooper et al., 2014] do not take into account any of these aspects. In fact, these methods show the Markov property [Norris, 1998], where the probability of moving to the next state depends only on the current state. For instance, the probability of a random walker starting from u_1 reaching i_3 does not depend on the previous states u_1 and i_1 but only on u_2. Thus, we claim graph-based approaches should benefit from the consideration of previous states for recommending.

1.2.3 Space and Time Complexity

Due to the size of the transition matrix, the methods presented in [Christoffel et al., 2015; Cooper et al., 2014; Fouss et al., 2007, 2005; Gori and Pucci, 2007; Singh et al., 2007; Yildirim and Krishnamoorthy, 2008] suffer from memory limitation as the dataset size increases. In addition, these methods are computationally intensive as they require matrix multiplications or matrix inversions. To overcome these issues, Christoffel et al. [2015] and Cooper et al. [2014] propose to approximate the final distribution by a sampling process. The proposed methods are applicable to medium or large size datasets, but recommendation accuracy depends on the total number of random walks adopted in the sampling process. Therefore, we claim there is a lack of computationally efficient methods in the context of graph-based approaches that do not degrade recommendation accuracy.

1.3 Hypothesis and Goals

In this dissertation, we build on the main hypothesis that the user-item rating matrix contains a valuable source of latent information that MF and graph-based approaches should take into account for improving recommendation. In this context, our specific hypotheses are:

- graph-based recommendation can benefit from the exploitation of both distributional aspects of the item ratings and similarities between users and/or between items,
- MF should take into account both user-user and item-item similarities to produce more accurate recommendations.

The main goal of this dissertation is to propose and evaluate CF methods that exploit the information contained in the items' ratings and the similarities between different entities present in the matrix to improve recommendation accuracy. Furthermore, the proposed methods should be applicable to larger datasets. To accomplish this goal and test our hypotheses, the specific goals are to:

1. Propose Bayesian scoring functions and path scoring mechanisms to improve graph-based recommendation.
2. Exploit information from states in a path in the user-item graph to improve recommendation accuracy. For instance, given the path $\langle u_1, i_1, u_2, i_3 \rangle$, how does user u_2 (candidate item i_2) compare with the target user u_1 (item in the user history i_1)?
3. Devise a more computationally efficient strategy to exploit short-length paths in the bipartite user-item graph than the state-of-the-art graph-based methods.
4. Propose a MF model and a computationally efficient inference algorithm that embed both user-user and item-item similarities.

1.4 Contributions

We propose a graph-based recommendation method that includes three path scoring mechanisms, and four scoring functions based on two Bayesian models that are at the core of our graph-based recommendation approach. Following the approaches presented in [Christoffel et al., 2015; Cooper et al., 2014], we exploit three-step paths in the user-item bipartite graph starting from the target user with the advantage we resort neither to matrix multiplications nor to sampling processes. Instead, we propose the enumeration of all such paths hence overcoming the computational burden incurred by the allocation and multiplication of transition matrices. We analytically show the proposed enumeration process is more computationally efficient than the approaches presented in [Christoffel et al., 2015; Cooper et al., 2014]. Finally, we devise a random

walk CF approach that exploits the set of items' ratings for item sampling. We carried out experiments on several publicly available datasets and provide a comprehensive empirical evaluation. The results show our better method clearly outperforms the approaches presented in [Christoffel et al., 2015; Cooper et al., 2014] in all metrics and datasets used for comparison. It is common knowledge in the CF literature that MF approaches are the state-of-the-art methods. Thus, we compare our method against a state-of-the-art MF approach and show our method provide better results in all but one dataset. To the best of our knowledge, our methods are the first three-step graph-based approach to exploit distributional aspects of the ratings and take similarities into account to boost recommendation accuracy. This set of contributions is presented in Chapter 3. In fact, preliminary results were published in [Lopes et al., 2016] and we have been working on a paper to be submitted to a top-tier journal where we present our improved models, algorithms and results.

We propose SMF, a MF model that jointly decomposes the user-item rating matrix and both user-user and item-item similarity matrices. In fact, our model makes use of shared latent factors to decompose such matrices, so that similarities play a fundamental role in regularizing the latent factors. Different from most works found in the literature, our approach makes no use of additional data, such as user demographics or item metadata. Our empirical results demonstrate the effectiveness of SMF, with significant improvements over state-of-the-art MF approaches across several publicly available datasets. Moreover, we empirically show SMF improvements are due to the joint factorization of both user-user and item-item similarity matrices. Finally, we also provide a breakdown analysis to better characterize the circumstances where SMF provides substantial improvements. To the best of our knowledge, SMF is the first MF approach to joint factorize both user-item rating matrix and both user-user and item-item similarities matrices. These contributions are discussed in Chapter 4 and were submitted to the 11th ACM Conference on Recommender Systems (RecSys 2017).

In summary, the contributions of this dissertation are five-fold: (i) we propose a more computationally efficient graph-based approach than those presented in [Christoffel et al., 2015; Cooper et al., 2014], (ii) we propose novel Bayesian scoring functions and path scoring mechanisms that lie at the core of our graph-based recommendation approach, (iii) we propose a novel random walk algorithm that takes advantage of the statistical distribution of the ratings for item sampling, (iv) we propose a novel Bayesian MF model that jointly factorizes the user-item data matrix and both user-user and item-item similarity matrices, (v) we present a comprehensive empirical evaluation of our methods against state-of-the-art CF approaches across several datasets and provide analytical results to understand their effectiveness and applicability.

1.5 Thesis Organization

This thesis is organized as follows. Chapter 2 covers background. Chapter 3 presents our Bayesian models, algorithms and results regarding our graph-based approaches. Chapter 4 introduces our SMF and discusses the results of the empirical evaluation. Finally, Chapter 5 presents concluding remarks and discusses future work.

Chapter 2

Background and Literature Review

In this chapter, we provide background in RS and detail related work. First, we introduce end pose the recommendation problem. After that, we briefly review content-based recommendation. Finally, we discuss CF methods, specially graph-based and MF approaches.

2.1 The Recommendation Problem

RS may serve two different purposes. On the one hand, they can be used to stimulate users into doing something such as buying a specific book or watching a specific movie, which turns out to improve merchant revenue. On the other hand, RS can also be seen as tools for dealing with information overload, as these systems aim to select the most interesting items from a larger set [Jannach et al., 2010]. In general, commercial RS present users to the best recommendations in that the set of recommended items might meet user preferences.

RS may generate personalized or non-personalized recommendations. In non-personalized recommendations, a fixed list of items is presented to any user regardless of his preferences. For instance, items with the highest average rating or the most sold items are recommended. In personalized recommendations, a set of candidate items is suitably chosen depending on the target user's taste; users with different preferences potentially receive different recommendation lists. As a matter of fact, empirical tests show customers tend to choose more often items suggested based on personalized methods compared to those recommended based on non-personalized approaches [Jannach et al., 2016]. In both settings, recommendations are made on the basis of users' feedback. Thus, RS fundamentally have to implement ways to acquire feedback; these systems may differ in the way they accomplish such task.

Users' feedback may be acquired by *implicitly* monitoring user behavior or *explicitly* asking users about their preferences [Jannach et al., 2010]. In implicit feedback contexts, the feedback is inferred in an implicit way through customer actions such as buying or browsing an item; this sort of feedback is often easier to obtain because users are more likely to interact with items than to explicitly rate them. In explicit feedback contexts, in turn, users explicitly select a rating from a specified evaluation system, which in turn tries to quantify the user satisfaction regarding the item consumed.

The evaluation system may vary with the system; Aggarwal [2016] enumerates several sorts of evaluation systems where the most relevant are: (a) interval-based, where a discrete set of ordered numbers, for example, the Netflix's 5-star rating system, (b) binary, where user may express only a like or dislike, for example, users express only thumbs-up or thumbs-down in the YouTube's rating system, and (c) unary, where there is only a mechanism for a user to specify a liking for a item, for example, Facebook only provided in the past a like button to express liking for a post.

The Netflix challenge has profoundly changed the research in RS [Jannach et al., 2016]. First, great advance has been made with respect to the application of machine learning approaches for RS, where various forms of MF and ensemble methods proven to be successful. Second, the challenge led to a formulation of the recommendation problem as one of matrix completion. In this formulation, given an incomplete (usually sparse) user-item interaction matrix for training, the matrix completion problem amounts to predict (or fill in) the unobserved entries [Aggarwal, 2016]. Thus, we can recommend items with the highest predicted ratings to the target user.

Employed learning algorithms in the task of matrix completion are generally assessed on how well they can predict the values in some known entries that are deliberately held out. For instance, the root mean squared error is one of the most employed performance measures. However, methods with good accuracy at the matrix completion problem are not sufficient to make the best recommendations in many practical settings [Cremonesi et al., 2010; Jannach et al., 2016], which motivates an alternative formulation of the recommendation problem. In fact, Jannach et al. [2016] states "Predicting held-out matrix entries is really predicting the past rather than the future".

Different from the matrix completion formulation, the ranking formulation is concerned with ranking and selecting few items according to their utility for the customer. The motivation for the ranking formulation of the recommendation problem lies in the fact it is not necessary to predict the ratings of users for specific items. Instead, in this formulation, we seek to recommend the top-N items for a particular user, or determine the top-N users for a particular item [Aggarwal, 2016]. First, RS predict a score for items based on the customer's preferences and constraints [Ricci et al., 2011]. Next,

items are usually ranked in a non-increasing order with respect to the predicted scores. Finally, the top-N items in this ranked list are presented to the user, which is based on the fact customers tend to look at and select items at the beginning of a list [Jannach et al., 2010]. This formulation is usually referred to as the top-N recommendation problem.

The personalized recommendation problem can be formally posed as follows [Adomavicius and Tuzhilin, 2005]. Let U be the set of users and let I be the set of items that can be recommended. We define $\kappa : U \times I \to \mathbb{R}$ a utility function that measures the utility of an item $i \in I$ to the user $u \in U$. The recommendation problem amounts to solve the following optimization problem for a target user $u \in U$:

$$i_u^\star = \arg\max_{i \in I} \kappa(u, i)$$

where we must find the item $i_u^\star$ that maximizes the the user's utility. Alternatively, we can recommend the N best items to a user or a set of users to an item.

The key issue in RS lies in the utility function κ is only known for a subset of $U \times I$. As a result, we must resort to techniques for extrapolating the utility function to the whole set. For instance, in the context of matrix completion, the utility of an item is usually represented by a rating, which indicates how much a user liked the item, and the utility function is only defined for those items in the user's history.

In this context, personalized RS should be able to predict the utility for unknown user-item interactions and make personalized recommendations based on these predictions. To this end, several methods from machine learning, approximation theory, and various heuristics have been applied [Adomavicius and Tuzhilin, 2005]. In the rest of this chapter, we discuss two canonical approaches for personalized RS, where we review the most relevant methods and algorithms in the literature for each approach, and refer readers to up-to-date references.

2.2 Content-based Recommendation

In reality, it would be easy to recommend the new Harry Potter book to a user, if we knew that this book is a fantasy novel and he likes fantasy novels. A recommender can accomplish this task on the basis of a description of the new Harry Potter book characteristics and a description of the user's interests. In this context, the recommendation task consists of determining the items that match the user's preferences best with respect to attributes of the items in the user' history [Jannach et al., 2010]. This sort of systems is called content-based systems.

Content-based systems are largely used in scenarios where a significant amount of attribute information is available at hand; these attributes, in turn, are usually keywords, which are extracted from the product description [Aggarwal, 2016]. In fact, these systems try to match users to items based on the attributes of the items the user liked. Content-based systems depend on two sources of data [Jannach et al., 2010]: (a) description of items in terms of content attributes, and (b) user profile that describes her interests in terms of preferred item characteristics. Usually, this sort of systems employs feature extraction techniques to convert information from various sources into a suitable vector space representation. In addition, these systems construct a specific model for each user on the basis of her history of either buying or rating items.

In content-based recommendation, the utility $\kappa(u, i)$ of a candidate item i for a target user u is estimated based on the utilities assigned to items in the u's history that are similar to i. In other words, the system learns to recommend items that are similar to those items the user liked in the past. Let $Content(i)$ be an item profile and let $ContentProfile(u)$ be a user profile. Since content-based systems are designed mostly to recommend text-based items [Adomavicius and Tuzhilin, 2005], profiles are usually defined as vector of weights. In fact, each weight denotes the importance of the corresponding keyword and is usually determined by the term frequency/inverse document frequency (TF-IDF) [Baeza-Yates and Ribeiro-Neto, 1999]. User profiles are obtained by analyzing the content of the items in the user history. Thus, the utility function is usually defined as follows [Adomavicius and Tuzhilin, 2005]:

$$\kappa(u, i) = ContentProfile(u) \otimes Content(i)$$

where $\otimes$ is a heuristic scoring function, such as the cosine similarity measure [Baeza-Yates and Ribeiro-Neto, 1999].

Content-based systems have known drawbacks that limit their applicability [Jannach et al., 2010; Lops et al., 2011]. First, these systems are limited by the features extracted from the items to be recommended as content must be in a form that can be parsed automatically. In fact, some domains have issues with automatic feature extraction, such as graphical images, audio streams and video streams. Second, these systems show a overspecialization in that recommendations for a target user are limited to items that are similar to those already rated, that is, these systems are not able to recommend items that are different from anything the user has seen before. Since we focus on CF recommendation in this dissertation, we refer the reader to [Aggarwal, 2016] as an up-to-date reference on the subject.

In the next section, we discuss CF recommendation. In fact, this sort of

recommendation overcomes some inherent limitations of content-based recommendation [Desrosiers and Karypis, 2011]: i) items whose content is not available or difficult to obtain can be recommended based on the feedbacks provided by other users, ii) recommendation relies on the quality of items instead of relying on the content, which might not be a proper indicator of quality, and iii) items with very different content can be recommended as similar users might have interests for different items compared to the target user.

2.3 Collaborative Filtering Recommendation

In reality, users often resort to like-minded acquaintances for recommendations [Sinha and Swearingen, 2001]. For instance, to recommend a movie for a target user, we should find other users that have similar tastes with the target user, and recommend those movies most liked by these users. Thus, the recommendation task consists of determining a set of users similar to the target users and then selecting items liked by these users that fit the target user's tastes. In fact, this is the basic idea of CF recommenders.

The intuition behind CF recommendation is that if users shared the same interests in the past, they may also share similar tastes in the future. Thus, recommendations are based on the user behavior, which can be regarded as item purchases, rentals, clicks or ratings. The term CF is named as such because these systems must filter the most relevant items from a large candidate set, and users implicitly collaborate with one another [Jannach et al., 2010]. For example, suppose user u and v have purchase histories that strongly overlap and u has recently bought an item that v has not yet seen, so it is sensible to present this item to v.

CF recommendation amounts to identify users whose preferences are similar to those of the target user and then recommend items they have liked. In fact, these systems try to predict the utility of items for a target user based on the items previously rated by other users. To be precise, the utility $\kappa(u, i)$ of item i for target u is estimated based on the utilities $\kappa(u_2, i)$ assigned to item i by other users $u_2 \in U \setminus \{u_1\}$ similar to u [Adomavicius and Tuzhilin, 2005]. Jannach et al. [2010] present several questions that arise in the context of CF recommendation, where the two most relevant are: (a) how to find users (items) similar to the target user (candidate item)? and (b) how to measure similarity between users (items)?

Breese et al. [1998] group CF recommenders into two general classes: (a) memory-based algorithms, which are based on similarity among either users or items and uses

ratings directly in the prediction, and (b) model-based factor algorithms, which learn a predictive model to produce recommendations. These two approaches are discussed next.

2.3.1 Memory-based Approaches

Memory-based (or neighborhood-based) approaches exploit the fact that similar users display close patterns of rating behavior and similar items are prone to receive conforming ratings [Aggarwal, 2016]. These methods are essentially heuristics where the utility of a candidate item i for target user u is based on an aggregate of the feedbacks provided by the users most similar to user u that have consumed the candidate item [Adomavicius and Tuzhilin, 2005]:

$$\kappa(u,i) = \underset{v \in \eta(u,i)}{\text{aggr}} \; r_{vi}$$

where r_{vi} represents the feedback for user v on item i, and $\eta(u,i)$ defines a neighborhood that contains the users most similar to user u that have rated item i. The underlying assumptions of these approaches are twofold [Jannach et al., 2010]: i) if users had similar consumption behavior in the past they will have similar tastes in the future, and ii) user preferences remain stable and consistent over time. As a matter of fact, an alternative formulation can be obtained by aggregating the feedbacks given to the items most similar to i that have been consumed by the target user u.

In the literature, there exists two classes of these approaches: (a) *User-based*: the predicted rating for a candidate item is computed as the weighted average of the ratings provided by the k users most similar to the target user, where each weight is a measure of similarity between the target and the similar user [Resnick et al., 1994], and (b) *Item-based*: the predicted rating for a candidate item is computed as the weighted average of the ratings provided by the target user for the k items most similar to the candidate item, where each weight is a measure of similarity between the candidate and the similar item [Sarwar et al., 2001]; it was the earliest algorithm of choice by Amazon [Linden et al., 2003]. In conclusion, Sarwar et al. [2001] present empirical evidence that item-based approaches provide better recommendation accuracy than user-based counterparts.

The choice of an aggregation function is a fundamental issue to the recommendation accuracy in memory-based approaches. The simplest aggregation function does not take into account the similarity between peers and is defined as the average among the feedbacks: $\kappa(u,i) = \frac{1}{|U_i|} \sum_{v \in U_i} r_{vi}$, where U_i represents the set of users that have

consumed item i. In contrast, the most common aggregate function is the weighted sum, which is usually defined as:

$$\kappa(u,i) = C \sum_{v \in \eta(u,i)} sim(u,v) \times r_{vi}$$

where C is a normalizing constant. This function weights the feedback provided by a user v similar to user u by a similarity measure between these two users ($sim(u,v)$), where the more similar these users are the more weight the feedback r_{vi} will carry in the utility function.

In memory-based approaches, we must determine either similar users or similar items. Therefore, a measure of similarity is at the core of these approaches and directly impacts recommendation accuracy. The two most relevant similarity measures in the context of memory-based approaches are discussed next.

The Pearson correlation coefficient between two users u and v is defined by

$$Pearson(u,v) = \frac{\sum_{k \in I_u \cap I_v} (r_{uk} - \mu_u).(r_{vk} - \mu_v)}{\sqrt{\sum_{k \in I_u \cap I_v} (r_{uk} - \mu_u)^2}.\sqrt{\sum_{k \in I_u \cap I_v} (r_{vk} - \mu_v)^2}} \tag{2.1}$$

where I_u denotes the set of items that u rated and μ_u denotes the average of ratings given by u. In fact, the Pearson correlation takes into account the fact that different users may use the rating scale differently as it uses deviations from the average rating of the corresponding user. The Pearson correlation between two items is accordingly defined.

The cosine similarity measure between two users u and v is defined by

$$Cosine(u,v) = \frac{\sum_{k \in I_u \cap I_v} r_{uk}.r_{vk}}{\sqrt{\sum_{k \in I_u} r_{uk}^2}.\sqrt{\sum_{k \in I_v} r_{vk}^2}} \tag{2.2}$$

The cosine similarity measure between two items is accordingly defined. As with the Pearson correlation, ratings can be mean-centered before computing the similarity measure, which gives rise to the so-called *adjusted* cosine similarity measure. Finally, the Jaccard similarity measure between two users u and v is defined as follows:

$$Jaccard(u,v) = \frac{|I_u \cap I_v|}{|I_u \cup I_v|} \tag{2.3}$$

The Jaccard similarity measure between two items is accordingly defined. Different from the previous similarity measures, the Jaccard measure does not take the set of

ratings into account to compute similarities.

Aggarwal [2016] advocates that the *adjusted* cosine similarity measure generally provides superior results when compared to the Pearson correlation coefficient in the context of item-based approaches. However, the Pearson correlation coefficient is preferable to the cosine similarity measure because the former accounts for the fact that different users exhibit different levels of bias in rating. In the context of user-based approaches, the Pearson coefficient outperforms other measures [Jannach et al., 2010].

In reality, these approaches suffer from computational limitations. As a matter of fact, it is computationally expensive to compute similarities between pairs of users as the dataset size increases since this task retards the recommendation in real-time. To bypass this issue, the most common strategy is to calculate the similarities between all pairs of users or items in advance and recalculate them periodically [Adomavicius and Tuzhilin, 2005]. Thus, the utilities can be efficiently calculated on demand using precomputed similarities. Finally, Aggarwal [2016] advocates the use of similarity values as combination weights is heuristic and arbitrary.

The neighborhood selection plays an important a crucial role in this process. A naive approach amounts to include all users that have consumed item i in the neighborhood $\eta(u, i)$ for a target user u. Jannach et al. [2010] advocates this strategy increases the required calculation time and has an effect on the accuracy of the recommendation as the feedbacks of other users who are not really similar to u would be taken into account. The most effective approaches for reducing the size of the neighborhood are to define a minimum threshold of the similarity or to limit the size to a fixed a number k and include the k nearest users to the target user in the neighborhood. However, Herlocker et al. [1999] show a trade-off in selecting the ideal threshold in that: i) if the threshold is too high, the neighborhood size will be very small (predictions can be made for very few items), and ii) if the threshold is too low, the neighborhood size are no significantly reduced.

Memory-based approaches are able to capture local associations in the data as they exploit a neighborhood of the target user (candidate item) through similarity measures [Desrosiers and Karypis, 2011]. As a result, these approaches can recommend an item very different from the usual users' taste or an item not known by the target user if some of his closest neighbors have consumed this item. The main advantages of these approaches are [Desrosiers and Karypis, 2011]: i) simplicity: these methods are simple to implement, ii) justifiability: they provide explainable recommendations since the neighbors might be presented as a justification, and iii) stability: they are little affected by the addition of users, items and feedbacks.

When compared to User-based CF, Item-based CF is more amenable to pre-

computation of (item-item) similarities, which makes it suitable for large-scale deployments [Linden et al., 2003]. Furthermore, Item-based methods often provide more relevant recommendations because the user's rating set is used to perform the recommendation [Aggarwal, 2016]. For further information on memory-based approaches, we refer the reader to [Aggarwal, 2016; Desrosiers and Karypis, 2011].

2.3.1.1 Graph-based Approaches

In the context of RS, graphs provide a structural representation of the relationships among users, items or both. Graph-based approaches try to overcome the major problem in the computation of similarity in canonical neighborhood-based methods as the former defines similarity with the use of either structural transitivity or ranking techniques. In other words, these approaches allow nodes that are not directly connected to influence each other by propagating information along the edges of the graph [Desrosiers and Karypis, 2011], where: i) the greater the weight of an edge, the more information is allowed to pass through it, and ii) the influence of a node on another should be smaller if the two nodes are further away in the graph. Thus, these approaches are more effective for sparse ratings matrices since they exploit structural transitivity of edges for the recommendation process [Aggarwal, 2016].

Graph-based approaches rely on graph models to define neighborhoods and make use of structural transitivity or ranking techniques to produce recommendations [Aggarwal, 2016]. In fact, the graphs can be constructed on the users, on the items, or on both, where edges encode the similarity or interaction between nodes. The user-item graph is usually defined as an undirected and bipartite graph $G = (U \cup I, E)$, where there exists an edge $(u, i) \in E$ if and only if user u has rated item i. For instance, Figure 1.3 displays the user-item bipartite graph for the rating matrix presented in Figure 1.2. In turn, the user (item) graph is defined by an even number of hops in the user-item graph between users (items). Aggarwal [2016] advocates these two sort of graphs are preferred over the user-item graph since they might take into account the number and similarity of common nodes while creating the edges.

Different from the canonical memory-based approaches previously discussed, in graph-based approaches, two users might be considered neighbors even if they have rated no item in common. To this end, graph-based approaches make use of short paths in the graph between users to define the neighborhood and exploit the notion of indirect transitivity between nodes to produce recommendations [Aggarwal, 2016]. Therefore, graph-based approaches provide a different way of defining neighborhoods, which can be useful in sparse settings. In fact, the notion of indirect connectivity is

achieved with the use of path-based or random walk-based strategies.

In the path-based strategy, the similarity between two nodes is evaluated as a function of the number and the length of paths connecting the two nodes [Desrosiers and Karypis, 2011]. In a seminal paper, Aggarwal et al. [1999] propose a path-based method for CF, where data are modeled as a directed user graph and edges' weight is determined based on the notions of *horting* and *predictability*. Horting is an asymmetric relation between users that quantifies the number of mutually specified ratings between two users. In fact, user u horts user v if $I_u \cap I_v \geq \alpha$ or $\frac{I_u \cap I_v}{I_u} \geq \beta$; where α and β are parameters, and I_u represents the set of items rated by user u. Predictability, in turn, quantifies the level of similarity among the common ratings provided by a pair of users, so that user v predicts user u if u horts v and there exists a linear transformation $f(\cdot)$ such that $\frac{\sum_{k \in I_u \cap I_v} |r_{uk} - f(r_{vk})|}{|I_u \cap I_v|} \leq \gamma$, where γ is a parameter. The rating of a target user u for an item i is computed by determining all the directed shortest paths from user u to all other users who have rated item i.

In the random walk-based strategy, the similarity between two nodes is evaluated as a probability of reaching these nodes in a random walk. Thus, this strategy considers indirect connectivity because a walker from one node to another may use any number of steps [Aggarwal, 2016]. This process can be described with a first-order Markov process [Norris, 1998] defined by a set of n states and a $n \times n$ row-stochastic transition probability matrix P where the probability of jumping from state i to j at any step t is given by $p_{ij} = \mathbb{P}(s(t+1) = j | s(t) = i)$ and $s(t)$ represents the process' current state in time t. Let $\pi(t)$ be the state probability distribution at step t, the evolution of the Markov chain is given by $\pi(t+1) = P^\top \pi(t)$. This process converges to a stable distribution vector $\pi(\infty)$ corresponding to the positive eigenvector of $P^\top$ with an eigenvalue of 1 [Norris, 1998]. Once the stable distribution $\pi(\infty)$ has been obtained, items can be recommended according to the corresponding rank in $\pi(\infty)$. Next, we discuss the most relevant random walk-based approaches.

Fouss et al. [2007, 2005] propose a CF approach that relies upon random walks over the user-item bipartite graph. They consider four similarity measures, namely, average first-passage time, average commute time, Euclidean commute time distance and pseudo inverse K^+ of the Laplacian matrix K, where $K = B - A$, B is a diagonal matrix where each entry represents vertex degrees associated to the user-item bipartite graph, and A represents the adjacency matrix. Results show K^+ provides the best performance for a movie dataset.

ItemRank [Gori and Pucci, 2007] is a random walk based scoring algorithm for recommendation in an item graph where edges connect items that have been rated by common users. This approach ranks the preferences of a target user u for a candidate

item i as the probability of a random walk to visit i. To this end, the authors define a normalized correlation matrix C where the correlation between two items is proportional to the number of users that rated both items. Experiments show ItemRank performs better than the methods proposed by Fouss et al. [2007, 2005].

In fact, ItemRank is a biased version of the seminal PageRank algorithm [Page et al., 1999] where, for a user u, a personalized ranking vector IR_u is iteratively computed until convergence as follows

$$IR_u^{t+1} = \alpha \cdot C \cdot IR_u^t + (1 - \alpha) \cdot d_u$$

where $\alpha \in (0, 1)$ is a parameter and d_u is a normalized vector that takes into account the set of items rated by u. Thus, the relevance of a candidate item i is given by the corresponding value of IR_u. As with ItemRank, Yildirim and Krishnamoorthy [2008] propose a random walk algorithm that obtains the steady state distribution to produce recommendations. In contrast, each entry in the correlation matrix C between two items is proportional to the similarity between them. To be precise, the authors study correlations given by the cosine and the adjusted cosine similarity. For a friendly PageRank explanation, we refer the reader to [David and Jon, 2010; Rajaraman and Ullman, 2011].

Singh et al. [2007] propose an approach that combines social relationships and ownership data to make recommendations. To this end, they model user-item relations as a bipartite graph and augment it with user-user social links. The approach is based on random walks with absorbing states and then induces a distribution per user over all items. A walker begins from a target user from where it may transition to a friend or to an item. Once the walker reaches an item, he cannot be transitioned out of it since this is an absorbing state. The authors evaluate the proposed method using data from an online game service to suggest items the user might buy and from a text corpus to suggest words to papers. Following the method presented in [Singh et al., 2007], Baluja et al. [2008] propose a random walk approach with absorbing states in the user-video graph. In particular, their method performs label propagation where nodes that have labels forward the labels to their neighbors until convergence.

The methods previously discussed must be executed until convergence or obtain the steady state distribution, which may be impractical for large datasets. The next two works we present are also based on random walks but they do not require the execution until convergence. Instead, short-step random walks underlie these methods, which turns out to be less time consuming. These two methods are detailed next.

Cooper et al. [2014] propose three scoring algorithms called P^3, P^5 and P^3_α, which

are based on random walks on the bipartite graph representing associations between users and items. Let $G = (U \cup I, E)$ be an undirected bipartite graph of users and items, where U is the set of users, I is the set of items and there exists an edge between user u and item i if u rated i. The authors define a $(|U| + |I|) \times (|U| + |I|)$ transition matrix $P = B^{-1}A$, where $A = (a_{ij})$ is the adjacency matrix associated to G and $B = (b_{ii})$ is a diagonal matrix where each entry equals the degree of the corresponding vertex. In fact, an entry p_{ui} in P defines the probability of a user u reaching item i in a single step. For instance, Figure 2.1 illustrates the transition matrix P for the user-item graph displayed in Figure 1.3, where one can see P is in fact a row-stochastic matrix.

$$P = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ i_1 \\ i_2 \\ i_3 \\ i_4 \end{array} \begin{array}{c} \begin{array}{ccccccc} u_1 & u_2 & u_3 & i_1 & i_2 & i_3 & i_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Figure 2.1: Example of a transition matrix P.

P^3 and P^5 are based on the distribution of random walks of three and five steps, respectively, starting from the target user vertex. As a matter of fact, these distributions are obtained by means of matrix multiplications. In fact, the transition probability p^3_{ui} for P^3 from user u to item i after a random walk of length three is given by:

$$p^3_{ui} = \sum_{j \in I} \sum_{v \in U} \frac{a_{uj}}{b_{uu}} \times \frac{a_{jv}}{b_{jj}} \times \frac{a_{vi}}{b_{vv}} \tag{2.4}$$

In turn, P^3_α generalizes P^3 in that its transition matrix is raised to the power of $\alpha \in \mathbb{R}_{>0}$. Experiments show P^3_α provides better results than P^3, which in turn provides better results than P^5.

Due to the memory burden of the proposed methods, as they require $(|U| + |I|) \times (|U| + |I|)$-dimensional matrix allocations, the authors resort to estimating the distribution per user over all items via random walk sampling. They show random walk sampling for P^3 and P^5 are more memory efficient compared to methods based on matrix calculations, so random walk sampling can be applied to larger datasets at the cost of reduced recommendation quality. As a conclusion, the authors show P^3

and P^3_α provide better results than the methods presented in [Fouss et al., 2007, 2005; Gori and Pucci, 2007].

Christoffel et al. [2015] introduce an algorithm called RP^3_β, which is based on P^3, to optimize accuracy and diversity. RP^3_β compensates for the influence of popular items by taking into account item popularity in the ranking given by P^3. Let P^3_{ui} be the original score of item i for target user u as the outcome of P^3. RP^3_β re-weights the score with $\tilde{P}^3_{ui} = P^3_{ui}/b^\beta_{ii}$, where b_{ii} represents the degree of vertex i and $\beta \in \mathbb{R}_{>0}$. Experiments show RP^3_β increases accuracy and diversity when compared to P^3. Following Cooper et al. [2014], due to memory limitations, the authors resort to a random walk sampling to estimate the distribution per user over all items by using 5 million random walks. Recently, the authors extend their prior work [Christoffel et al., 2015] by proposing online update mechanisms so that RP^3_β does not need to recompute the values for the entire dataset as new feedbacks are provided [Paudel et al., 2016].

A naive implementation of the exact methods presented in [Christoffel et al., 2015; Cooper et al., 2014; Paudel et al., 2016] results in time complexity $\Theta((|I| + |U|)^3)$ and space complexity $\Theta((|I| + |U|)^2)$. To overcome this issue, Cooper et al. [2014] propose an approach that splits the matrices into blocks and computes P^3 by a sequence of multiplications and additions of these blocks, which results in time complexity $\Theta(|I|^2 \times |U|^2)$ and space complexity $\Theta(|I| \times |U|)$. However, as we move to medium/large size datasets, the methods discussed here have computational limitations since the size of the transition matrix may become too large to fit into memory.

The graph-based methods presented in [Christoffel et al., 2015; Cooper et al., 2014] do not take into account for recommendation any latent information present on the set of ratings. In fact, these graph-based approaches make use of the set of user-item pairs to define the structure of the user-item bipartite graph. Thus, the rating system is completely discarded in these approaches. For instance, in these methods, there is no distinction between a scenario where a user gives the maximum possible rating for an item and a scenario where a different user gives the minimum possible rating for the same item. Likewise, there is no difference between an item that has 8 positive out of 10 ratings and an item that has 2 positive out of 10 ratings. Therefore, we wonder if these approaches are suitable for explicit feedback contexts.

To fulfill this gap, we propose a graph-based approach that takes advantage of ratings given by the users to make recommendations. In fact, we enumerate and exploit three-step paths in the user-item bipartite graph where we assign weights to paths by exploiting distributional aspects of the item's ratings in the path by means of the Bayesian scoring functions we propose. Furthermore, we take into account affinity of pairs of users and/or pairs of items in the paths to produce recommendations. Empir-

ical evaluation in six freely available datasets attest the effectiveness of our approach, which outperforms state-of-the-art graph-based and MF approaches.

2.3.2 Model-based Approaches

In contrast to memory-based systems, model-based (or latent-factor) approaches use the ratings/feedback to learn a predictive model that represents latent characteristics of the users and items with factors. In fact, these models map both users and items to a joint latent factor space of reduced dimensionality, so that user-item interactions are modeled as inner products in that space. For instance, in a movie domain, item factors might correspond to aspects of a movie such as the genre or the type, but they can also be uninterpretable [Jannach et al., 2010]. Likewise, these methods are able to determine that a given user is a fan of movies that are both comedy and romantic, without defining the notions comedy and romantic, and recommend to the user a romantic comedy that may not have been known to this user [Desrosiers and Karypis, 2011]. Once the model is trained using the available data, the latent factors are used to produce recommendations.

In latent-factor approaches, the basic idea is to exploit the fact that significant portions of the rows and columns of data matrices are highly correlated. As a result, the data has redundancies and the resulting data matrix is often approximated quite well by a low-rank matrix. Because of the redundancies in the data, the fully specified low-rank approximation can be determined even with a small subset of the entries in the original matrix. This fully-specified low rank approximation often provides a robust estimation of the missing entries [Aggarwal, 2016].

Model-based approaches make use of dimensionality reduction methods to directly estimate the data matrix and are considered the state-of-the-art in CF [Koren et al., 2009]. These approaches address the problems of limited coverage and sparsity by projecting users and items into a reduced latent space that captures their most salient features. Thus, more meaningful relations can be discovered as users and items are compared in this dense subspace of high-level features instead of the rating space [Aggarwal, 2016; Desrosiers and Karypis, 2011].

To the best of our knowledge, Sarwar et al. [2000] were the first to apply a model-based approach in the context of RS. To be precise, the authors explored the use of Singular Value Decomposition (SVD) to capture latent relationships between customers, produce a low dimensional representation of the original user-item space and generate top-N product recommendations for customers. In this approach, the user-item rating matrix R of rank n is factorized into three matrices $R = A\Sigma B^\top$, where

A is the $|U| \times n$ matrix of left singular vectors, B is the $|I| \times n$ matrix of right singular vectors, and Σ is the $n \times n$ diagonal matrix of singular values ordered in decreasing order. By retaining the $k < n$ largest singular values and their corresponding singular vectors, we obtain reduced matrices A_k, Σ_k and B_k, so that $R \approx R_k = A_k \Sigma_k B_k$. In fact, R_k is the closest rank-k matrix to R with respect to the Frobenius norm. We can obtain the user and item factor matrices $X = A_k \Sigma^{1/2}$ and $Y = \Sigma^{1/2} B_k^\top$, respectively. Thus, a rating prediction is given by $\hat{r}_{ui} = x_u^\top y_i$, where x_u (y_i) is the u-row (i-row) of X (Y) and represents the coordinates of user u (item i) projected in the k-dimensional space. For a thorough explanation in SVD, we refer the reader to [Poole, 2006].

The use of SVD in the context of RS is greatly limited since this method requires a full rating matrix. However, it is common knowledge in RS literature that the user-item rating matrix is usually sparse since users interact with a small amount of available items in the catalog. To bypass this issue, it is possible to assign a default value to a missing entry. However, this sort of imputation has some drawbacks for it introduces bias in the data and makes the rating matrix dense [Aggarwal, 2016; Desrosiers and Karypis, 2011]. This issue motivates the next latent factor approach, where learning is performed using only the known ratings.

MF [Koren et al., 2009] is the most successful latent factor approach and associates each user u with a user factors vector $x_u \in \mathbb{R}^f$ and each item i with an item factors vector $y_i \in \mathbb{R}^f$, where $f \in \mathbb{N}$ represents the dimension of the reduced latent factor space. In these models, a missing feedback r_{ui} is estimated as $\hat{r}_{ui} = x_u^T y_i$, that is, as the inner product between the user and item factors vectors. In training, we aim at solving the following optimization problem

$$\underset{x^*, y^*}{\arg\min} \sum_{r_{ui} \in M} (r_{ui} - x_u^T y_i)^2 + \lambda_u \sum_{u \in U} \|x_u\|_2^2 + \lambda_i \sum_{i \in I} \|y_i\|_2^2$$

where λ_u, λ_i are regularization parameters for avoiding overfitting and M is the set of known feedbacks. Thus, the optimization problem amounts to minimize the squared error, which captures the error between actual and predicted ratings, plus squared Euclidean norm terms. For a thorough and recent review on latent factor approaches present in the literature, we refer the reader to Aggarwal [2016]. Next, we discuss an improved MF approach that tackles missing feedback.

Hu et al. [2008] propose Weighted Regularized Matrix Factorization (WRMF), a weighted matrix factorization approach for implicit feedback. In this approach, factors

are computed by solving the following non-linear optimization problem:

$$\arg\min_{x^*,y^*} \sum_{u\in U}\sum_{i\in I} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda_x \sum_{u\in U} \|x_u\|_2^2 + \lambda_y \sum_{i\in I} \|y_i\|_2^2 \tag{2.5}$$

where $c_{ui} = 1 + \alpha \cdot r_{ui}$ and p_{ui} assumes one if $r_{ui} > 0$ or zero otherwise. In the context of implicit feedback, the authors assume $r_{ui} \in \mathbb{N}$ since an item might be consumed more than once. For instance, a user might play a song as many times as she desires. The p_{ui} value is derived by binarizing r_{ui}. Intuitively, c_{ui} is a confidence measure in observing p_{ui} in that its value is a linear function of r_{ui} and takes the role of a weight in the objective function. The optimization problem not only models scenarios where users provide feedback for items but also those where there exists no feedback (interaction) between a user-item pair. The problem with this approach is that all these unknown interactions are presented to the learning algorithm as negative feedback, then regularization plays a crucial role to avoid overfitting [Rendle et al., 2009]. Finally, the authors resort to an alternating least square process to resolve the optimization problem. Computational results show the proposed method outperforms the baselines used for comparison. Although the authors propose WRMF to the context of implicit feedback, Aggarwal [2016] suggests WRMF can also be utilized in the context of explicit feedback.

Different from WRMF that predicts the absolute values of ratings that individual users would give to the yet unseen items (as discussed above), there has been a class of model-based approaches that predicts the relative preferences of users [Christakopoulou and Banerjee, 2015; Lee et al., 2014; Rendle et al., 2009; Shi et al., 2012]. For example, in a movie recommendation application, preference-based filtering techniques would focus on predicting the correct relative order of the movies, rather than their individual ratings [Adomavicius and Tuzhilin, 2005]. In this context, Bayesian Personalized Ranking Matrix Factorization (BPRMF) [Rendle et al., 2009] is the most representative method in this class and is discussed next.

Rendle et al. [2009] propose Bayesian Personalized Ranking Matrix Factorization (BPRMF), a generic method for solving personalized ranking derived by a Bayesian analysis of the problem. The authors make use of item pairs as training data to optimize for correctly ranking item pairs. To this end, the authors provide each user u with a personalized total ranking $>_u \subset I \times I$ where if an item i has been consumed by u then $i >_u j$ for any item j not consumed by u. The authors model the likelihood function $p(>_u |\theta) = \sigma(\hat{x}_{uij}(\theta))$ for a given $(i, j) \in I \times I$, where σ is the sigmoid function, associate a prior distribution such that $\theta \sim N(0, \Sigma_\theta)$ and maximize the posterior

distribution $p(\theta| >_u)$; $\hat{x}_{uij}(\theta)$ is an arbitrary real-valued function that captures the relationship between u, i and j. The authors instantiate BPRMF in the context of matrix factorization and k-nearest neighbors. In the context of matrix factorization, the authors define $\hat{x}_{uij} = \hat{x}_{ui} - \hat{x}_{uj}$, where for predicting $\hat{x}_{ui}$, they factorize a $|U| \times |I|$-dimensional matrix X by the product of two low-rank matrices W and H, so that $\hat{x}_{ui} = w_u^T h_i$. Learning relies on stochastic gradient descent with bootstrap sampling. Computational results show the proposed method outperforms the baselines used for comparison which includes WRMF.

Canonical MF approaches usually show limited accuracy for short history users, as it cannot accurately infer their latent factors [Liang et al., 2016]. Therefore, several works [Adams et al., 2010; Agarwal and Chen, 2009; Jamali and Ester, 2010; Ma et al., 2009, 2008; Porteous et al., 2010; Shan and Banerjee, 2010; Shi et al., 2013; Yang et al., 2013] propose to embed information from additional data, such as users' location and items' reviews, into MF for improving recommendation accuracy. Nonetheless, such data may not always be available, severely limiting the applicability of these works.

Social RS, for example, exploit information about social networks to improve recommendation accuracy. In this context, social MF approaches [Jamali and Ester, 2010; Ma et al., 2008; Yang et al., 2013] leverage trust values (friendship or influence) to further regularize user factors. These works are based on the belief that if a user u assigns a high trust value to a user v, it is possible to improve recommendations to u by borrowing information from v's latent vector. These trust values, in turn, are usually explicitly assigned in an asymmetric manner by users to peers. Results show these methods improve recommendation accuracy when compared to those that do not take into account any trust information. Next, we discuss a MF approach that improves recommendation accuracy but rely on no additional data other than the user-item interaction matrix.

Liang et al. [2016] extend WRMF and then propose CoFactor, a co-factorization model for implicit feedback that jointly factorizes the user-item interaction matrix and an item-item co-occurrence matrix with shared latent factors. For a pair of items (i, j), the authors propose a novel similarity measure given by $s_{ij} = \max\{PMI(i,j) - log(k), 0\}$, where $PMI(i,j) = \log \frac{\#(i,j).D}{\#(i).\#(j)}$, $\#(i,j)$ corresponds to the number of users that consumed both items i and j, $\#(i) = \sum_{j \in I} \#(i,j)$, $\#(j) = \sum_{i \in I} \#(i,j)$, D is the total number of pairs and k is a parameter that controls the sparsity of the similarity

matrix. Learning aims at solving the the following non-linear optimization problem:

$$\underset{x^*,y^*,w^*,b^*,c^*}{\arg\min} \overbrace{\sum_{u\in U}\sum_{i\in I} c_{ui}(p_{ui} - x_u^\top y_i)^2}^{\text{MF}} + \lambda_x \sum_{u\in U} \|x_u\|_2^2 + \lambda_y \sum_{i\in I} \|y_i\|_2^2 + \overbrace{\sum_{m_{ij}\neq 0} (s_{ij} - y_i^\top w_j - b_i - c_j)^2}^{\text{similarity factorization}} + \lambda_w \sum_{i\in I} \|w_i\|_2^2 \tag{2.6}$$

where p_{ui} indicates if user u interacts with item i, x_u represents respectively user factors, y_i and w_i represent item factors, c_{ui} is a weight parameter to balance the unobserved feedbacks ($p_{ui} = 0$), b_i and c_j are item biases, and λ_x, λ_y and λ_w are regularization parameters. In fact, CoFactor (Equation (2.5)) differs from WRMF (Equation (2.6)) by the last two terms. On the one hand, y_i are latent factors shared by both MF and similarity factorization. On the other hand, w_j are latent factors that account solely for item-item co-occurrence and then lack interpretability. Results show the proposed method improves the quality of recommendation in three datasets and CoFactor provides more accurate recommendations compared to WRMF.

CoFactor only takes similarity between pairs of items into account for recommendation. Likewise, the two set of item latent factors (y and w) make interpretability of the model very hard. Moreover, CoFactor is suitable designed for implicit feedback contexts and exploits a single similarity measure, so that the authors does not assess the effectiveness of the proposed similarity measure against relevant similarity measures found in the literature. In this context, we propose a MF model for explicit feedback domains that jointly factorizes the user-item rating matrix and both user-user and item-item similarity matrices. Based on the similarity measure value, our approach forces not only similar users (items) to have close latent vectors but also dissimilar users (items) to have near-orthogonal or near-opposite latent vectors. Thus, our method is able to provide more accurate recommendations compared to scenarios where either user-user or item-item similarity matrix is jointly factorized.

Chapter 3

Graph-based Recommendation Approaches

In this chapter, we propose our graph-based approach to solve the ranking (top-N) formulation of the recommendation problem. The methods we propose rely on the Bayesian paradigm to produce recommendation as it provides a principled way of incorporating prior information to data, so it allows the assignment of non-zero probabilities to unseen events. These probabilities represent our beliefs in those events. Experiments on several publicly available datasets demonstrate the effectiveness of our proposed approaches compared to state-of-the-art recommenders.

3.1 Rating Models

Let U be the set of users and I be the set of items. We define the set of evaluations $D = \{(u, i, r_{ui}) | u \in U, i \in I, r_{ui} \in R\}$, where R represents the set of possible rating values (e.g. $R = \{1, 2, 3, 4, 5\}$ or $R = \{0, 1\}$). Thus, $(u, i, r_{ui}) \in D$ represents the evaluation made by user u where item i received rating r_{ui}. Given $u \in U$, $I_u = \{i \in I | (u, i, r_{ui}) \in D\}$ represents the set of items evaluated by user u. Given $i \in I$, let $U_i = \{u \in U | (u, i, r_{ui}) \in D\}$ be the set of users who evaluated item i and $R_i = \{r_{ui} \in R | (u, i, r_{ui}) \in D\}$ be the set of ratings assigned to item i.

Let $G = (U \cup I, E)$ be an undirected bipartite graph, where $E \subseteq U \times I$ represents the set of edges such that $E = \{(u, i) | (u, i, r_{ui}) \in D\}$, that is, there exists an edge between user u and item i if u rated i. We define $\Delta_U = \max_{u \in U} |I_u|$ as the maximum vertex degree in U and Δ_I is similarly defined for I, $\Gamma_U = \frac{\sum_{u \in U} |I_u|}{|U|}$ as the average degree of vertices in U and Γ_I is similarly defined for I.

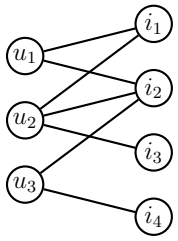


Figure 3.1: User-item bipartite graph.

Because G is bipartite, any odd-step path starting from a vertex in U ends up in a vertex in I. Given $u_1 \in U$, we define $\mathcal{P}_{u_1} = \{\langle u_1, i_1, u_2, i_2\rangle | i_1 \in I_{u_1}, u_2 \in U_{i_1} \setminus \{u_1\}, i_2 \in I_{u_2} \setminus I_{u_1}, (u_1, i_1), (u_2, i_1), (u_2, i_2) \in E\}$. In fact, $\mathcal{P}_{u_1}$ represents the set of all three-step paths starting from user u_1 that end in an item not consumed by this user. From Figure 3.1, one can see $\mathcal{P}_{u_1} = \{\langle u_1, i_1, u_2, i_3\rangle, \langle u_1, i_2, u_2, i_3\rangle, \langle u_1, i_2, u_3, i_4\rangle\}$, where items i_3 and i_4 are not consumed by the target user u_1. Thus, we must exploit $\mathcal{P}_{u_1}$ to build a personalized recommendation list for u_1 which amounts to quantifying the likelihood of u_1 consuming a set of candidate items, such as $\{i_3, i_4\}$. Next, we present our method to exploit such paths and propose two models that account for the set of ratings. Loosely speaking, the models differ basically in the set of possible ratings and in the family of probability distributions.

3.1.1 Binary Rating Model

We assume that $R = \{0, 1\}$, where, given $u \in U, i \in I$, $r_{ui} = 1$ represents a positive assessment while $r_{ui} = 0$ represents a negative assessment. Thus, we define the set $R_i^+ = \{r_{ui} \in R_i | r_{ui} = 1\}$ of positive assessments and the set $R_i^- = \{r_{ui} \in R_i | r_{ui} = 0\}$ of negative assessments received by item i.

Given $i \in I$, let Y_i be a binary random variable that assumes 1 if i receives a positive assessment and 0 otherwise, where $\mathbb{P}(Y_i = 1) = \theta_i$. We place a Beta distribution as the prior of θ_i, where $\theta_i \sim \text{Beta}(\bar{a}, \bar{b})$. Intuitively, θ_i represents the unknown reliability of item i within the range $(0, 1)$, where larger values indicate more reliability. As $|R_i|$ increases, the Beta distribution shape tends to concentrate around its mean, then this notion of reliability turns out to be more precise. After observing R_i, we update our knowledge about θ with respect to item i, so that

$$\theta_i | R_i \sim \text{Beta}(a, b), \tag{3.1}$$

where $a = \bar{a} + |R_i^+|$ and $b = \bar{b} + |R_i^-|$. This result follows from the conjugacy property between Bernoulli and Beta distribution families.

3.1.2 Categorical Rating Model

We assume that R is a discrete set of ordered ratings, such as $R = \{1, 2, 3, 4, 5\}$. Without loss of generality, we assume $R = \{1, 2, \ldots, r\}$, so that $r = |R|$. Given $i \in I$, let $Y_i \sim \text{Cat}(r, \boldsymbol{\theta})$ be a categorical random variable parameterized by an r-dimensional vector $\boldsymbol{\theta} = \{\theta^1, \theta^2, \ldots, \theta^r\}$ where $\mathbb{P}(Y_i = k) = \theta^k$. Thus, $Y_i = k$ amounts to item i receiving a rating $k \in R$. We assume Y_i and Y_j are independent random variables for any $i, j \in I$. We place a Dirichlet distribution as the prior of $\boldsymbol{\theta}$, where $\boldsymbol{\theta} \sim \text{Dir}(r, \boldsymbol{\alpha})$. After observing R_i, we update our knowledge about $\boldsymbol{\theta}$ with respect to item i so that

$$\boldsymbol{\theta}_i | R_i \sim \text{Dir}(r, \boldsymbol{\alpha} + \boldsymbol{n}_i) \tag{3.2}$$

where $\boldsymbol{n}_i = (n_i^1, n_i^2, \ldots, n_i^r)$ is an r-dimensional vector so that $n_i^k = |\{r_{ui} \in R_i | r_{ui} = k\}|$; Equation (3.2) follows from the conjugacy property between categorical and Dirichlet distribution families. From Equation (3.2), we can obtain the posterior predictive distribution

$$\mathbb{P}(Y_i = k | R_i) = \frac{\alpha^k + n_i^k}{\sum_{a \in R} \alpha^a + n_i^a} \tag{3.3}$$

3.2 Algorithms

We now propose two different methods that leverage the models previously discussed. The methods we propose make a recommendation list for a target user $u \in U$ based on $\mathcal{P}_u$, where there might be several different paths ending in the same vertex. As an illustration, from Figure 3.1, one can see there are two paths in $\mathcal{P}_{u_1}$ where i_3 is the last vertex and one path that ends in i_4. The methods differ from each other in how they explore such paths. The first method resorts to random walk sampling to produce the final ranking. The second method, in turn, enumerates and scores all the paths starting from the target user.

3.2.1 Rating-based Random Walk

We propose Rating-based Random Walk (RRW), a random walk approach to make a recommendation list for a target user $u \in U$. Following the methods presented in [Christoffel et al., 2015; Cooper et al., 2014], RRW performs three-step random walks starting from u. In contrast, our method exploits distributional aspects with respect to the set of ratings for those items in the walk. Intuitively, a walker is more likely to move towards items with more positive ratings than those with equal amount

of ratings but with less positive evaluations. Next, we provide the details of our random walk algorithm.

input : $G = (U \cup I, E)$, $u_1 \in U$, κ
output: f_{u_1}
1 **for** $n \in \{1, 2, \ldots, \kappa.|I_{u_1}|\}$ **do**
2 | sample item $i_1 \in I_{u_1}$ according to Algorithm 2
3 | sample uniformly user $u_2 \in U_{i_1} \setminus \{u_1\}$
4 | sample item $i_2 \in I_{u_2} \setminus \{i_1\}$ according to Algorithm 2
5 | $f_{u_1}(i_2) := f_{u_1}(i_2) + 1$
6 **end**

Algorithm 1: Random walk algorithm for CF.

input : $G = (U \cup I, E)$, $u \in U$
output: $i \in I_u$
1 **for** $j \in I_u$ **do**
2 | sample rating r_j according to Equation (3.3)
3 **end**
4 sample an item $i \in I_u$ with probability proportional to r_i

Algorithm 2: Improved item sampling for CF random walks.

Algorithm 1 illustrates our random walk approach for CF. Starting from the target user u_1, we sample and move to an item $i_1 \in I_{u_1}$ in line 2. Next, we uniformly sample and move towards a user $u_2 \in U_{i_1} \setminus \{u_1\}$ in line 3. In line 4, we sample and move to a candidate item $i_2 \in I_{u_2} \setminus I_{u_1}$. Finally, we update the rank function $f_{u_1} : I \to \mathbb{N}$ for target user u_1 in line 5, where the candidate item's rank is proportional to the fraction of random walks that end in such item. The total number of random walks performed is given by $\kappa \cdot |I_u|$, where $\kappa \in \mathbb{N}$ is a parameter. Contrary to the uniform sampling for users, we sample an item $i \in I_u$ from a given user u with probability proportional to a rating sampled from Equation (3.3); this procedure is illustrated in Algorithm 2. Note also that the variance of Y_i decreases with $|R_i|$ and so, for items with a smaller number of ratings, the probability (3.3) is more similar to its prior $\boldsymbol{\alpha}$. Thus, we take into account the uncertainty over R_i as different ratings might be sampled with respect to the random walks once they reach u. RRW differs from those methods presented in [Christoffel et al., 2015; Cooper et al., 2014] in that the sampling process for items accounts for the uncertainty over the set of ratings whereas those methods uniformly sample both users and items.

3.2.2 Scaled Path Scoring

Differently from RRW, this method enumerates all the paths in $\mathcal{P}_u$ so that the candidate item i's rank for a target user u is proportional to the number of paths in $\mathcal{P}_u$ that end in i. As a concrete example, i_3 is likely to be ranked higher than i_4 in u_1's recommendation list from Figure 3.1. In addition, this method exploits the set of ratings by assigning a score to each path, where this score relies on scoring functions that compare one item to another in the light of the Bayesian paradigm.

For $u_1 \in U$, we define a ranking function $f_{u_1} : I \to \mathbb{R}_{>0}$ as $f_{u_1}(i_2) = \sum_{p:=\langle u_1,i_1,u_2,i_2\rangle \in \mathcal{P}_{u_1}} s(p)$ and a scoring function $s : \mathcal{P} \to \mathbb{R}_{>0}$, where i_2 is a candidate item for recommendation and $\mathcal{P} = \cup_{u \in U} \mathcal{P}_u$. In fact, $f_{u_1}(i_2)$ adds up the weights of all the three-step paths in $\mathcal{P}_{u_1}$ connecting u_1 and i_2, in other words, f_u shows an additive effect that accounts for the existence of such paths for scoring i_2. In turn, $s(\langle u_1, i_1, u_2, i_2\rangle)$ exploits distributional aspects of i_1 and i_2's ratings on the basis of the Bayesian paradigm; we present the Bayesian scoring functions in Section 3.2.3.

input : $G = (U \cup I, E)$, $u_1 \in U$, scoring function s
output: f_{u_1}
1 **for** $i_1 \in I_{u_1}$ **do**
2 **for** $u_2 \in U_{i_1} \setminus \{u_1\}$ **do**
3 **for** $i_2 \in I_{u_2} \setminus I_{u_1}$ **do**
4 $f_{u_1}(i_2) := f_{u_1}(i_2) + s(\langle u_1, i_1, u_2, i_2\rangle)$
5 **end**
6 **end**
7 **end**

Algorithm 3: Vanilla computation of f_{u_1} for target user u_1.

input : $G = (U \cup I, E)$, $u_1 \in U$, scoring function s, ψ
output: f_{u_1}
1 **for** $i_1 \in I_{u_1}$ **do**
2 **for** $u_2 \in U_{i_1} \setminus \{u_1\}$ **do**
3 **for** $i_2 \in I_{u_2} \setminus I_{u_1}$ **do**
4 $f_{u_1}(i_2) := f_{u_1}(i_2) + \psi \times s(\langle u_1, i_1, u_2, i_2\rangle)$
5 **end**
6 **end**
7 **end**

Algorithm 4: Similarity-based computation of f_{u_1} for target user u_1 where ψ is suitably instantiated among ψ_U, ψ_I or $\psi_U \times \psi_I$.

Algorithm 3 illustrates our first method. The algorithm receives as input the user-item undirected bipartite graph G, target user u_1, a scoring function s. In lines

1-3, we systematically iterate over each neighbor of the corresponding vertex. Thus, we enumerate all paths in $\mathcal{P}_{u_1}$. In line 4, we compute the value function f_{u_1} for the final vertex i_2 in the path $\langle u_1, i_1, u_2, i_2 \rangle \in \mathcal{P}_{u_1}$. As a result, the algorithm outputs the personalized ranking function f_{u_1} that spans all items not consumed by u_1 that are reachable within three steps starting from the target user. Hence, our method fundamentally explores a vicinity of the target user to reach candidate items. Motivated by user and item neighborhood-based approaches, we wonder whether it is possible to also exploit similarities between users or between items with respect to such vicinity at the hope of providing better recommendations.

In reality, users often resort to like-minded acquaintances for recommendations of items such as books, movies or TV series, and consume items similar to those they liked in the past [Sinha and Swearingen, 2001]. For example, we usually rely on a acquaintance's recommendation when selecting a movie to watch. In contrast, our method depicted in the Algorithm 3 does not take into account any similarity between users nor between items to produce the final ranking. Given $\langle u_1, i_1, u_2, i_2 \rangle \in \mathcal{P}_{u_1}$ starting from target user u_1, it is reasonable to assume: (i) the more u_1 and u_2 share similar preferences, the greater this path should score, and (ii) the more similar i_1 and i_2 are, the greater this path should score. Thus, we propose to incorporate user-user and/or item-item similarities when calculating the rank function f.

We exploit the number of users that both rated a pair of items as a correlation measure between items. Furthermore, we also consider the number of items in common rated by a pair of users as a similarity measure between users. To this end, we make use of the Jaccard similarity measure to exploit structural properties (graph statistics) of the user-item bipartite graph. We define an item correlation measure $\psi_I : I \times I \to \mathbb{R}_{>0}$ where $\psi_I(i_1, i_2) = |U_{i_1} \cap U_{i_2}|/|U_{i_1} \cup U_{i_2}|$, and a user correlation measure $\psi_U : U \times U \to \mathbb{R}_{>0}$ where $\psi_U(u_1, u_2) = |I_{u_1} \cap I_{u_2}|/|I_{u_1} \cup I_{u_2}|$. As a matter of fact, we propose three distinct strategies for scaling the scoring function s:

- U-strategy: ψ_U as a scale factor, where we only consider the effect of the similarity between users in the path,
- I-strategy: ψ_I as a scale factor, where we solely take into consideration the similarity between items in the path, and
- UI-strategy: $\psi_U \times \psi_I$ as a scale factor, where we study both effects in conjunction.

Thus, we take into account previous states of the path for recommending a candidate item.

Algorithm 4 illustrates the general algorithm where the parameter ψ is instantiated according to the scaling strategy. For instance, $\psi = \psi_U \times \psi_I$ amounts to the method where the UI-strategy is chosen for scaling purposes. For the sake of better understanding, we hereafter refer to the method presented in Algorithm 3 as the Vanilla-strategy, where no similarity measure is taken into account.

3.2.2.1 Complexity Analysis

Algorithm 3 incurs space complexity $O(|U|+|I|+|E|)$ and time complexity $O(|U| \cdot \Delta_u \cdot \Delta_I^2)$ at worst case scenario considering the set of target users U. Recalling the methods presented in [Christoffel et al., 2015; Cooper et al., 2014], they incur space complexity $\Theta((|U| + |I|)^2)$ and time complexity $\Theta((|I| + |U|)^3)$ at worst case scenario. Thus, our method are more computationally efficient compared to state-of-the-art graph-based approaches.

Our method only considers three-step paths since the exploitation of longer odd-step paths increases time complexity, which may render this approach infeasible for larger datasets. In fact, Cooper et al. [2014] show that considering three-step attains better results than considering five-step paths in their method. Algorithm 4 compared to Algorithm 3 has an additional time complexity cost due to similarity measure calculations. Since the similarity measures used are based on the intersection between two sets of elements, intersection computation incurs $\Theta(n \times \log(n))$ time complexity where n is the number of elements in the set. We provide implementation details so that this additional cost does not heavily impact our method. Namely, we propose the use of memoization.

3.2.3 Bayesian Scoring Functions

Starting from a target user u_1, there might be several three-step paths that end in a candidate item i_2 as previously illustrated in Figure 3.1. Our method takes into account the existence of such paths for scoring i_2, but we advocate these paths must not contribute evenly for the final score. To this end, we propose scoring functions that are based on the Bayesian paradigm.

One can wonder what the chances are user u_1 ends up consuming item i_2 given (i) users u_1 and u_2 consumed at least one item i_1 in common, and (ii) u_2 consumed i_2. One might compare i_2 to i_1 according to some criteria as an attempt to infer these chances. Thus, we propose next scoring functions so that each one exploits a different aspect by comparing i_2 to i_1 in the light of the our Bayesian models that explicitly exploit the items ratings.

3.2.3.1 Binary based Scoring

We here propose several scoring functions based on the model presented in Section 3.1.1.

Binary Inequality Scoring. We propose a scoring function dubbed Binary Inequality Scoring (BIS) where

$$s(\langle u_1, i_1, u_2, i_2\rangle) = \mathbb{P}(\theta_{i_2} \geq \theta_{i_1} | R_{i_2}, R_{i_1})$$

given $\langle u_1, i_1, u_2, i_2\rangle \in \mathcal{P}_{u_1}$. We can compute $\mathbb{P}(\theta_{i_2} > \theta_{i_1} | R_{i_2}, R_{i_1}) = \int_0^1 \int_0^1 1(\theta_2 \geq \theta_1) f(\theta_2 | a_2, b_2) f(\theta_1 | a_1, b_1) d\theta_2 d\theta_1$ from (3.1), where $f(.)$ represents the probability density function of the Beta distribution; we discuss how to compute $\mathbb{P}(\theta_{i_2} > \theta_{i_1} | R_{i_2}, R_{i_1})$ at the end of this section. Intuitively, BIS represents the probability of the reliability of candidate item i_2 being greater than the reliability of item i_1 in the user history.

Binary Joint Scoring. We propose a scoring function called Binary Joint Scoring (BJS) where

$$s(\langle u_1, i_1, u_2, i_2\rangle) = \mathbb{P}(Y_{i_2} = 1 | R_{i_2}) \times \mathbb{P}(Y_{i_1} = 1 | R_{i_1})$$

given $\langle u_1, i_1, u_2, i_2\rangle \in \mathcal{P}_{u_1}$. We can obtain $\mathbb{P}(Y_{i_2} = 1 | R_{i_2}) = \frac{a+1}{a+b+1}$ from (3.1). Intuitively, BJS represents the probability of both i_2 and i_1 receiving positive assessments where we assume Y_{i_2} and Y_{i_1} are independent, that is, $\mathbb{P}(Y_{i_2} = 1 | Y_{i_1} = 1, R_{i_2}, R_{i_1}) = \mathbb{P}(Y_{i_2} = 1 | R_{i_2})$.

Binary Odds Ratio Scoring. We propose a scoring function denominated Binary Odds Ratio Scoring (BORS) where

$$s(\langle u_1, i_1, u_2, i_2\rangle) = \frac{\eta_{i_2}}{\eta_{i_1}}$$

given $\langle u_1, i_1, u_2, i_2\rangle \in \mathcal{P}_{u_1}$, where we define $\eta_{i_1} = \frac{\mathbb{P}(Y_{i_1}=1|R_{i_1})}{\mathbb{P}(Y_{i_1}=0|R_{i_2})}$. Intuitively, BORS represents how large the odds of i_2 receiving a positive assessment is when compared to the odds of i_1 receiving a positive assessment.

3.2.3.2 Categorical based scoring

We here propose a scoring function based on the Bayesian model presented in Section 3.1.2. Thus, this scoring function overcomes the strong assumption about the rating scale from the previous scoring functions and then are applicable to a wider range of scenarios.

Categorical Inequality Scoring. We propose a scoring function named CIS

where

$$s(\langle u_1, i_1, u_2, i_2\rangle) = \mathbb{P}(Y_{i_2} \geq Y_{i_1}|R_{i_2}, R_{i_1})$$

given $\langle u_1, i_1, u_2, i_2\rangle \in \mathcal{P}_{u_1}$. From Equation (3.2), we can analytically compute $\mathbb{P}(Y_{i_2} \geq Y_{i_1}|R_{i_2}, R_{i_1})$:

$$\mathbb{P}(Y_{i_2} \geq Y_{i_1}|R_{i_2}, R_{i_1}) = \sum_{r_{i_2}\in R}\sum_{r_{i_1}\in R} 1(r_{i_2} \geq r_{i_1})\mathbb{P}(Y_{i_2} = r_{i_2}, Y_{i_1} = r_{i_1}|R_{i_2}, R_{i_1}) \tag{3.4}$$

$$= \sum_{r_{i_2}\in R}\sum_{r_{i_1}\in R} 1(r_{i_2} \geq r_{i_1})\mathbb{P}(Y_{i_2} = r_{i_2}|R_{i_2}) \times \mathbb{P}(Y_{i_1} = r_{i_1}|R_{i_1}) \tag{3.5}$$

$$= \sum_{r_{i_2}\in R}\sum_{r_{i_1}\in R} 1(r_{i_2} \geq r_{i_1})\frac{\alpha^{r_{i_2}} + n_{i_2}^{r_{i_2}}}{\sum_{a\in R}\alpha^a + n_{i_2}^a} \times \frac{\alpha^{r_{i_1}} + n_{i_1}^{r_{i_1}}}{\sum_{a\in R}\alpha^a + n_{i_1}^a} \tag{3.6}$$

where $1(\cdot)$ is an indicator function.

From Equation (3.4) to (3.5), we make use of the independence property between two independent random variables. CIS represents the probability of i_2's rating being greater than or equal to i_1's rating in the light of the posterior distribution. Intuitively, this scoring function favors paths where i_2 potentially provides a better shopping experience than i_1.

The scoring methods we propose are global for they do not depend on the target user. In contrast, the ranking function f_{u_1} is in fact personalized for each target user since the final result depends on the paths starting from vertex u_1. In conclusion, our ranking function exploits the neighborhood of the target user in the bipartite user-item graph.

Implementation Details. We represent G as an adjacency list and obtain $\mathcal{P}_u$ by systematically enumerating according to G all three-step paths starting from vertex u. Thus, our implementation for P^3_α and RP^3_β makes use of such enumeration instead of relying on matrix calculations or sampling as proposed by Christoffel et al. [2015] and Cooper et al. [2014]. Cook [2005] proposes an analytical approach to compute Beta inequalities. However, this approach is computationally expensive since it relies on recursive function calls. Thus, we resort to numerical integration methods to compute $\mathbb{P}(\theta_{i_2} \geq \theta_{i_1}|R_{i_2}, R_{i_1})$ from BIS. To this end, we make use of GNU Scientific Library [Galassi et al., 2009] as the numerical integration library for BIS and memoization for storing results and, then, avoid recomputing such values. To avoid the occurrence of probabilities equal to zero in the proposed scoring functions, we make use of pseudo

Table 3.1: Datasets properties.

Dataset	$\|D\|$	$\|U\|$	$\|I\|$	Δ_U	Δ_I	Γ_U	Γ_I	$\rho(\%)$
BX	42,137	1,842	2,065	964	225	22.87	20.40	98.75
CDs	445,412	15,592	16,184	2,069	603	28.56	27.52	98.81
Electronics	347,393	20,247	11,589	317	1,376	17.16	29.98	99.91
Epinions	300,304	10,706	8,945	525	1,491	28.05	33.57	99.73
Kindle	367,478	14,356	15,885	664	377	22.60	23.13	99.82
ML 1M	998,539	6,040	3,260	2,233	3,428	165.32	306.30	97.26

counts. For each item, we add one positive and one negative pseudo-assessment as if they both were given by a dummy user. The reason to add one for each end of the scale is to balance out the analysis so we do not induce bias. We also resort to memoization for storing similarity measure results, hence reducing time complexity of our method at the cost of increased memory consumption. In fact, similarity measures are computed if needed as we enumerate the paths.

3.3 Experimental Setup

Our experiments are designed to answer the following questions:

Q1. How do the scaling strategies and scoring functions impact recommendation accuracy?

Q2. How effective are RRW and our best enumeration-based method compared with each other and with state-of-the-art approaches?

Q3. How robust is our best method to various degrees of sparsity?

To answer these questions we make use of several freely available datasets and baselines in our experiments. In the following, we describe the setup of our empirical investigations.

3.3.1 Datasets

The BookCrossing (BX as an abbreviation) dataset [Ziegler et al., 2005] contains data about book lovers that exchange books all around the world. The original dataset contains 278,858 anonymous users providing 1,149,780 ratings about 271,379 books spanning a period of time from August to September 2004. Ratings are given by users

in a ten-star rating scale. In our experiments, we removed ratings related to implicit feedback.

The MovieLens datasets [Harper and Konstan, 2015] are widely used in the Recommender Systems literature and comprise four datasets of increasing sizes. Each dataset consists of users' preferences for movies expressed in a five-star rating scale. In this work, we consider the MovieLens 1M (ML 1M for short) dataset that comprises 1 million ratings from 6,000 users on 4,000 movies.

The Amazon dataset [McAuley et al., 2015] contains 142.8 million product reviews spanning May 1996 - July 2014 and product metadata from Amazon. Ratings are given by users in a five-star rating scale. In this work, we select three representative product categories from that dataset, namely, CDs & Vinyl (CDs as an abbreviation), Electronics and Kindle. In turn, the Epinions dataset [Massa and Avesani, 2007] contains 664,824 product ratings from 49,290 users on 139,738 products.

Since we do not focus on cold start issues, we keep only users and items with at least 10 ratings. As a result, the data used for our experiments differ from the original datasets and are summarized in Table 3.1, where ρ shows sparsity percentage. A common sense in the RS literature is that most users consume a small fraction of the items present in the catalogue and many items have few ratings [Park and Tuzhilin, 2008]. One can see that except for the MovieLens 1M dataset the maximum and average degree for items (Δ_I and Γ_I) and users (Δ_U and Γ_U) are significantly less than the corresponding total number in the dataset, which turns out to favor our path enumeration approaches; the datasets are severely sparse.

3.3.2 Evaluation Methodology

The datasets are not compliant with the assumption $R = \{0, 1\}$ posed in Section 3.1.1, which makes those scoring functions discussed in Section 3.2.3.1 inapplicable in these datasets. For instance, in the Amazon dataset $R = \{1, 2, 3, 4, 5\}$. In order to circumvent this limitation, we propose next a dataset transformation so that those scoring functions can be properly applied to such datasets.

Given a dataset and user u present on it, we calculate the average $\bar{r}_u$ of all ratings given by u. Assume that item i was graded $\hat{r}_{ui}$ by u. We define r_{ui} equals 1 if $\hat{r}_{ui} \geq \bar{r}_u$ or 0 otherwise and replace $(u, i, \hat{r}_{ui})$ in D by (u, i, r_{ui}). As a result, we make a transformation of the original set of ratings into a dichotomized set where the user bias is taken into account. We advocate each user has a personal understanding of the rating system, which motivates our decision of having the user bias dichotomize the original set of ratings. In contrast, the original rating scale in the dataset is used for

the methods based on the Bayesian model presented in Section 3.1.2.

We carried out experiments on the datasets described by using 5-fold cross validation. To this end, we randomly partitioned D into 5 subsets, where a subset T is retained for testing, and the remaining 4 subsets are used as training data. We repeated this process 5 times and report the average result across all the test folds in the 5 trials.

We further standardize the users' ratings in each dataset into a relevance scale personalized for each individual user. Precisely, given a user $u \in U$, we define the set $T_u \subset I_u$ of items rated by u and used for testing. We further partition $T_u = T_u^+ \cup T_u^-$, where $T_u^+ = \{i \in I_u | (u, i, r_{ui}) \in T, r_{ui} \geq \bar{r}_u\}$ is the set of items used for testing that u has rated above average and $T_u^- = T_u \setminus T_u^+$, in turn, is the set of items used for testing that u has rated below average. To express our lack of information about data we adopt uninformative prior distributions so that the Beta distribution reduces to the Uniform distribution and the Dirichlet distribution reduces to the Uniform distribution over the $(r-1)$-simplex.

3.3.3 Evaluation Criteria

Given $u \in U$, let L_u be the recommendation list non-increasingly ordered according to f_u. For a given $N \in \mathbb{N}$, let L_u^N be the recommendation list composed of the N highest scored items according to f_u. Given L_u, $L_u[i]$ represents the item at position i and the same applies for L_u^N. We report our results using three ranking evaluation metrics largely used in the RS literature [Shani and Gunawardana, 2011], as described next.

Precision (P). Given L_u, P@N is defined as:

$$P@N = \frac{|L_u^N \cap T_u^+|}{N} \tag{3.7}$$

Average Precision (AP). Given L_u, AP is given by the following equation:

$$AP = \sum_{i:=1}^{|L_u|} \frac{P@i \times 1(L_u[i] \in T_u^+)}{|T_u^+|} \tag{3.8}$$

where $1(\cdot)$ is an indicator function and $P@i$ is given by Equation (3.7) described above.

Normalized Discounted Cumulative Gain (nDCG). Given a ranking L_u, nDCG [Järvelin and Kekäläinen, 2002] measures the quality of a ranking based on the graded relevance of the recommended items with respect to an ideal ranking of items.

It is defined as:

$$\text{nDCG@}N = \frac{DCG@N}{IDCG@N} \tag{3.9}$$

where $DCG@N$ is defined as:

$$DCG@N = \text{rel}(L_u^N[1], T_u) + \sum_{i:=2}^{N} \frac{\text{rel}(L_u^N[i], T_u)}{\log_2(i)}$$

where $\text{rel} : I \times \mathcal{P}(I) \setminus \emptyset \rightarrow \{0, 1, 2\}$ is a function that indicates the relevance of an item and we define it as:

$$\text{rel}(i, T_u) = \begin{cases} 2, & \text{if } i \in T_u^+ \\ 1, & \text{if } i \in T_u^- \\ 0, & \text{otherwise} \end{cases}$$

In turn, $IDCG@N$ is the maximum possible $DCG@N$ where L_u^N is sorted in non-increasing order by item relevance. In this work, we consider items in T_u^+ as highly relevant, items in T_u^- as fairly relevant and those not in T_u as irrelevant items for target user u.

In the next section, we report our results as an average of P@5, nDCG@5, and AP (denoted MAP for Mean Average Precision) figures across all users in the test subsets of all cross-validation rounds. Statistically significant differences are verified using a two-tailed paired t-test. In particular, we use the symbols $\triangle$ ($\triangledown$) and $\blacktriangle$ ($\blacktriangledown$) to denote statistically significant increase (decrease) at the $p < 0.05$ and $p < 0.01$ levels, with the symbol $\circ$ used to denote no significant difference from a given baseline.

3.3.4 Recommendation Baselines and Parameter Tuning

We consider P_α^3, RP_β^3, Most Popular (MP) and WRMF as recommendation baselines in our investigations. P_α^3 and RP_β^3 are state-of-the-art graph-based CF approaches. MP, in turn, sorts items according to their global popularity and so the recommendation list does not depend on the target user. Finally, WRMF is a representative matrix-factorization approach for recommendation. In our experiments, we make use of the WRMF implementation provided in the Java port of the MyMediaLite [Gantner et al., 2011] recommender system framework.[1] Since P_α^3, RP_β^3, and WRMF are designed for implicit feedback contexts, all observed interactions $(u, i, r_{ui}) \in D$ are considered as positive preferences for these methods, so that rating values are completely discarded.

[1] https://github.com/jcnewell/MyMediaLiteJava

We performed 5-fold cross validation for tuning the parameters for RRW and baselines so that they provide the best possible results for the sake of a fair comparison. For RRW, we experimented with values of $\kappa \in \{1, 5, 10, 25, 50, 100\}$, recalling the number of random walks performed is given by $\kappa \cdot |I_u|$ where $u \in U$ is the target user. For P^3_α and RP^3_β, we tested values of $\alpha, \beta \in [0.2, 2.0]$ in steps of 0.2. Finally, for WRMF, we tested values of number of factors $f \in \{10, 20, 50, 100\}$ and regularization parameter $\lambda \in \{0.1, 1, 10, 100\}$. Table 3.2 shows the best parameter configuration obtained for each dataset with respect to MAP. Thus, we report the results for these methods on the basis of the best parameter configuration found.

Table 3.2: Best parameter configuration for each dataset.

Dataset	P^3_α	RP^3_β	WRMF	RRW
BX	$\alpha = 0.2$	$\beta = 0.2$	$f = 50, \lambda = 0.1$	$\kappa = 100$
CDs			$f = 50, \lambda = 0.1$	
Electronics			$f = 20, \lambda = 1$	
Epinions			$f = 50, \lambda = 1$	
Kindle			$f = 50, \lambda = 0.1$	
ML 1M			$f = 50, \lambda = 10$	

3.4 Experimental Results

In this section, we address each of the three research questions posed in Section 3.3 in turn. Since we evaluate four Bayesian scoring functions and four scaling strategies, we adopt a specific nomenclature to make clear which pair of scoring function and strategy is under consideration. Namely, CIS stands for CIS method where the Vanilla-strategy is used, CIS-U for our CIS with the U-strategy turned on, CIS-I where I-strategy is taken into account and CIS-UI where the UI-strategy is active; a similar nomenclature is applied for BIS, BJS and BORS.

Results are displayed in Tables 3.3–3.8. Each result entry shows a first significance symbol, metric value, in parenthesis the mean percentage improvement of our best result over the corresponding method in the entry and, when applicable, a second significance symbol. In fact, significance symbols are used to denote a statistically significant difference (or lack thereof) of the result in each entry compared to: a) our best result in case of the first symbol, and b) the Vanilla-strategy in case of the second symbol, which is applicable to our improved path scoring mechanisms. Finally, our best result is highlighted in bold.

3.4.1 Scaling Strategy and Scoring Functions Choice

To address research question Q1, we assess the performance of the four scaling strategies and scoring functions we discuss in this work. For the BX (Table 3.3) dataset, the UI-strategy provides the best results except for CIS-U and CIS-UI, where the U-strategy provides better results than the UI-strategy; the Vanilla-strategy shows the worst results. For the CDs (Table 3.4), Electronics (Table 3.5) and Kindle (Table 3.7) datasets, among all scaling strategies, the UI-strategy provides the best results while the Vanilla-strategy presents the worst results; the I-strategy outperforms the U-strategy. For the ML 1M dataset (Table 3.8), the U-strategy shows the best results except for BIS-U and BIS-UI, where the UI-strategy provides the best results in P@5 and nDCG@5; the I-strategy provides inferior results compared to the Vanilla-strategy except for BIS-I and BIS. Finally, for the Epinions dataset (Table 3.6), the U-strategy provides the best results except for BIS-U and BIS-UI, where the UI-strategy superior results compared to the U-strategy; the Vanilla-strategy provides the worst results. In summary, the introduction of any similarity measure provides statistically significant better results at the $p < 0.01$ level than the Vanilla-strategy in most settings, and our novel CIS provides the best results in all datasets.

Among our scoring functions, our novel CIS provides the best results across the datasets, followed by BJS, which in turn provides better results than BORS, while BIS provides the worst results. This result suggests our Binary Rating Model presented in Section 3.1.1 is not suitable for scenarios where the ratings scale is interval-based. In fact, we believe the transformation process discussed in Section 3.3.2 might incur information loss.

Recalling research question Q1, the introduction of similarities when calculating the rank function f boost the performance of our method. Moreover, the UI-strategy provides the best results in half of the cases while the U-strategy shows the best results in the remaining cases. These results attest the value of exploiting similarities compared to the Vanilla strategy for gains are at least 83.8% for BX, 88.1% for CDs, 42.1% for Electronics, 54.1% for Kindle, 22.2% for ML 1M and 73.6% for Epinions.

3.4.2 Recommendation Effectiveness

To address research question Q2, we contrast the effectiveness of RRW and our best enumeration-based method to that of the baselines. Recalling the baselines discussed in Section 3.3.4, P^3_α and RP^3_β are personalized graph-based CF approaches, MP is a non-personalized recommendation approach that ranks the candidate items according to popularity and WRMF is a state-of-the-art matrix-factorization approach for CF.

Table 3.3: Performance comparison of our method and baselines on BX dataset.

Method	MAP	P@5	nDCG@5
BIS	▼ 0.039 (227.7)	▼ 0.021 (276.6)	▼ 0.022 (306.3)
BIS-U	▼ 0.076 (68.8) ▲	▼ 0.043 (86.0) ▲	▼ 0.047 (90.5) ▲
BIS-I	▼ 0.060 (122.8) ▲	▼ 0.035 (137.4) ▲	▼ 0.039 (139.5) ▲
BIS-UI	▽ 0.090 (41.4) ▲	▼ 0.055 (44.6) ▲	▼ 0.061 (46.1) ▲
BJS	▼ 0.070 (83.8)	▼ 0.041 (92.9)	▼ 0.044 (101.8)
BJS-U	○ 0.110 (15.1) ▲	○ 0.067 (17.0) △	○ 0.075 (18.5) ▲
BJS-I	▼ 0.086 (52.5) ▲	▼ 0.055 (46.5) ▲	▼ 0.060 (50.9) ▲
BJS-UI	○ 0.112 (13.7) ▲	○ 0.073 (8.1) ▲	○ 0.080 (11.6) ▲
BORS	▼ 0.055 (131.0)	▼ 0.033 (139.8)	▼ 0.035 (153.0)
BORS-U	▽ 0.093 (36.1) ▲	▼ 0.056 (41.4) ▲	▽ 0.063 (41.8) ▲
BORS-I	▼ 0.073 (77.7) ▲	▼ 0.047 (72.2) ▲	▼ 0.052 (74.9) ▲
BORS-UI	▽ 0.101 (25.7) ▲	▽ 0.064 (23.1) ▲	○ 0.072 (23.3) ▲
CIS	▼ 0.082 (56.1)	▼ 0.050 (59.1)	▼ 0.054 (66.1)
CIS-U	**0.128** ▲	**0.079** ▲	**0.089** ▲
CIS-I	▼ 0.084 (55.0) ▲	▼ 0.055 (47.3) ○	▼ 0.061 (49.3) ▲
CIS-UI	○ 0.112 (13.3) ▲	○ 0.071 (10.8) ▲	○ 0.080 (11.2) ▲
RRW	▼ 0.064 (98.5)	▼ 0.040 (99.3)	▼ 0.043 (108.5)
MP	▼ 0.026 (388.4)	▼ 0.014 (450.3)	▼ 0.015 (479.8)
P^3_α	▼ 0.039 (226.0)	▼ 0.023 (243.7)	▼ 0.026 (250.0)
RP^3_β	▼ 0.021 (508.9)	▼ 0.011 (594.5)	▼ 0.013 (603.7)
WRMF	▼ 0.077 (65.5)	▼ 0.047 (68.1)	▼ 0.053 (68.2)

RRW. Compared to P^3_α, RP^3_β and MP, RRW provides better results across the datasets in all metrics. This result shows the effectiveness of incorporating the uncertainty with the set of ratings into random walks compared to P^3_α and RP^3_β, which are the state-of-the-art random walk approaches for graph-based CF. Notwithstanding, RRW is not able to provide better results than WRMF.

Best enumeration-based method. For the BX (Table 3.3), CDs (Table 3.4), Electronics (Table 3.5), Epinions (Table 3.6) and Kindle (Table 3.7) datasets, our best method in each dataset provides statistically significant increase at the $p < 0.01$ level over the baselines for all but one metric. To be precise, our best method in the Epinions dataset provides statistically significant increase at the $p < 0.05$ level for WRMF in nDCG@5. In particular, gains are of at least 65.5% for BX, 65.7% for CDs, 74.6% for Electronics, 179.7% for Kindle and 57.7% for Epinions. In contrast, for the ML 1M dataset (Table 3.8), though our best method surpasses P^3_α, RP^3_β and MP, where gains are of at least 40.9%, it provides inferior results compared to WRMF. In fact, our best

Table 3.4: Performance comparison of our method and baselines on CDs dataset.

Method	MAP	P@5	nDCG@5
BIS	▼ 0.028 (256.7)	▼ 0.021 (264.2)	▼ 0.020 (301.4)
BIS-U	▼ 0.046 (121.4) ▲	▼ 0.032 (132.7) ▲	▼ 0.032 (151.2) ▲
BIS-I	▼ 0.065 (55.7) ▲	▼ 0.049 (54.1) ▲	▼ 0.049 (64.6) ▲
BIS-UI	▼ 0.078 (28.6) ▲	▼ 0.058 (28.6) ▲	▼ 0.059 (35.5) ▲
BJS	▼ 0.054 (88.1)	▼ 0.039 (90.7)	▼ 0.040 (98.6)
BJS-U	▼ 0.073 (38.1) ▲	▼ 0.053 (41.1) ▲	▼ 0.055 (44.7) ▲
BJS-I	▼ 0.086 (17.4) ▲	▼ 0.063 (19.9) ▲	▼ 0.065 (22.5) ▲
BJS-UI	○ 0.095 (5.9) ▲	○ 0.070 (6.7) ▲	○ 0.074 (8.3) ▲
BORS	▼ 0.046 (120.9)	▼ 0.034 (118.6)	▼ 0.035 (129.4)
BORS-U	▼ 0.063 (59.9) ▲	▼ 0.047 (60.0) ▲	▼ 0.048 (66.4) ▲
BORS-I	▼ 0.077 (30.8) ▲	▼ 0.058 (29.4) ▲	▼ 0.060 (33.4) ▲
BORS-UI	▼ 0.087 (15.6) ▲	▼ 0.066 (13.4) ▲	▼ 0.069 (16.5) ▲
CIS	▼ 0.060 (66.9)	▼ 0.044 (71.9)	▼ 0.045 (77.4)
CIS-U	▼ 0.079 (27.9) ▲	▼ 0.057 (30.8) ▲	▼ 0.060 (32.5) ▲
CIS-I	○ 0.094 (7.6) ▲	▽ 0.069 (9.6) ▲	▽ 0.073 (10.4) ▲
CIS-UI	**0.101** ▲	**0.075** ▲	**0.080** ▲
RRW	▼ 0.057 (78.4)	▼ 0.045 (67.1)	▼ 0.047 (69.4)
MP	▼ 0.007 (1335.1)	▼ 0.005 (1299.2)	▼ 0.005 (1372.5)
P^3_α	▼ 0.028 (258.2)	▼ 0.022 (238.0)	▼ 0.023 (243.7)
RP^3_β	▼ 0.011 (786.0)	▼ 0.007 (998.5)	▼ 0.007 (1048.9)
WRMF	▼ 0.056 (80.1)	▼ 0.045 (68.4)	▼ 0.048 (65.7)

method presents results up to 27.2% inferior compared to WRMF. Finally, compared to RRW, our best enumeration-based method provides statistically significant increase at the $p < 0.01$ level over the baselines for all metrics, where gains are at least 19.8%. Recalling research question Q2, these results attest the effectiveness of our method for it provides the best results for five among six datasets in all metrics.

3.4.3 Recommendation Robustness

To address research question Q3, we assess the robustness of our method when recommending for users with various levels of sparsity. Figures 3.2 and 3.3 provide performance breakdowns (with respect to MAP) for our best enumeration-based method and baselines in all datasets. Each figure shows five bins representing user groups that rated an increasing amount of items organized according to the 10th, 35th, 65th and 90th percentiles of the corresponding dataset.

From Figure 3.2, one can clearly see our method provides the best recommen-

Table 3.5: Performance comparison of our method and baselines on Electronics dataset.

Method	MAP	P@5	nDCG@5
BIS	▼ 0.022 (76.3)	▼ 0.013 (84.4)	▼ 0.013 (103.2)
BIS-U	▼ 0.024 (64.2) ○	▼ 0.014 (72.5) △	▼ 0.014 (87.5) △
BIS-I	▼ 0.032 (21.8) ▲	▼ 0.020 (24.5) ▲	▼ 0.020 (29.9) ▲
BIS-UI	▽ 0.033 (15.9) ▲	▼ 0.021 (16.6) ▲	▼ 0.021 (21.6) ▲
BJS	▼ 0.027 (42.1)	▼ 0.017 (44.0)	▼ 0.017 (53.4)
BJS-U	▼ 0.029 (33.1) △	▼ 0.019 (33.2) △	▼ 0.018 (40.6) ▲
BJS-I	○ 0.036 (6.4) ▲	○ 0.023 (8.5) ▲	▽ 0.023 (10.3) ▲
BJS-UI	○ 0.038 (2.1) ▲	○ 0.024 (3.7) ▲	○ 0.025 (4.7) ▲
BORS	▼ 0.025 (56.1)	▼ 0.016 (57.7)	▼ 0.015 (69.0)
BORS-U	▼ 0.026 (46.1) ○	▼ 0.017 (47.0) △	▼ 0.016 (55.9) △
BORS-I	▽ 0.034 (15.1) ▲	▼ 0.021 (16.7) ▲	▼ 0.021 (20.4) ▲
BORS-UI	○ 0.035 (9.9) ▲	○ 0.022 (10.3) ▲	▽ 0.023 (13.0) ▲
CIS	▼ 0.028 (37.9)	▼ 0.018 (39.0)	▼ 0.017 (47.1)
CIS-U	▼ 0.030 (28.6) △	▼ 0.019 (27.5) △	▼ 0.019 (33.8) ▲
CIS-I	○ 0.037 (3.8) ▲	○ 0.024 (4.1) ▲	○ 0.024 (5.1) ▲
CIS-UI	**0.039** ▲	**0.025** ▲	**0.026** ▲
RRW	▼ 0.021 (83.3)	▼ 0.014 (77.8)	▼ 0.014 (80.9)
MP	▼ 0.017 (125.7)	▼ 0.010 (147.9)	▼ 0.010 (160.3)
P^3_α	▼ 0.015 (154.8)	▼ 0.010 (150.4)	▼ 0.010 (154.2)
RP^3_β	▼ 0.004 (860.1)	▼ 0.001 (2045.1)	▼ 0.001 (2163.3)
WRMF	▼ 0.023 (68.8)	▼ 0.014 (74.6)	▼ 0.015 (74.9)

dations in all the bins for the BX, CDs, Electronics and Kindle datasets. Recalling Q3, we conclude that our method is robust to the degree of sparsity of different users, with consistent improvements across these datasets. From Figure 3.3, one can see our method provides the best recommendations compared to P^3_α, RP^3_β and MP for the ML 1M and Epinions datasets. In particular, when compared to WRMF, our method provides the best recommendations for the two first bins in the Epinions dataset. One can clearly see WRMF systematically provides the best recommendations in all the bins for the ML 1M dataset. Next, we contrast the performance of our method and WRMF in finer degrees of sparsity.

Despite the consistent improvements observed across the user bases for all four datasets in Figure 3.2, an interesting observation can be drawn by zooming into highly active users on the CDs dataset. As shown in Figure 3.4a, our method provides the best results only in the first two bins (short history users) whereas WRMF provides the best recommendations in the last three bins (active users). This finding suggests

Table 3.6: Performance comparison of our method and baselines on Epinions dataset.

Method	MAP	P@5	nDCG@5
BIS	▼ 0.022 (214.0)	▼ 0.016 (206.1)	▼ 0.014 (295.8)
BIS-U	▼ 0.036 (90.5) ▲	▼ 0.026 (85.1) ▲	▼ 0.024 (128.2) ▲
BIS-I	▼ 0.027 (154.4) ▲	▼ 0.020 (143.8) ▲	▼ 0.019 (201.8) ▲
BIS-UI	▼ 0.041 (67.7) ▲	▼ 0.030 (60.3) ▲	▼ 0.029 (92.0) ▲
BJS	▼ 0.036 (90.2)	▼ 0.027 (73.6)	▼ 0.028 (97.8)
BJS-U	○ 0.066 (3.9) ▲	○ 0.046 (3.1) ▲	○ 0.053 (6.1) ▲
BJS-I	▼ 0.039 (77.3) ▲	▼ 0.029 (64.4) ▲	▼ 0.030 (86.9) ▲
BJS-UI	○ 0.060 (13.8) ▲	○ 0.045 (5.0) ▲	○ 0.048 (14.2) ▲
BORS	▼ 0.033 (106.8)	▼ 0.026 (85.6)	▼ 0.026 (114.8)
BORS-U	○ 0.058 (16.6) ▲	○ 0.042 (12.7) ▲	○ 0.047 (19.2) ▲
BORS-I	▼ 0.035 (98.0) ▲	▼ 0.026 (83.3) ▲	▼ 0.027 (106.8) ▲
BORS-UI	○ 0.053 (27.5) ▲	○ 0.040 (17.4) ▲	○ 0.043 (28.4) ▲
CIS	▼ 0.037 (85.2)	▼ 0.028 (71.6)	▼ 0.029 (90.2)
CIS-U	**0.068** ▲	**0.048** ▲	**0.056** ▲
CIS-I	▼ 0.039 (76.5) ▲	▼ 0.030 (64.0) ▲	▼ 0.032 (80.3) ▲
CIS-UI	○ 0.060 (12.2) ▲	○ 0.045 (4.4) ▲	○ 0.050 (9.9) ▲
RRW	▼ 0.026 (163.5)	▼ 0.023 (112.8)	▼ 0.024 (135.9)
MP	▼ 0.015 (360.1)	▼ 0.011 (316.8)	▼ 0.015 (273.2)
P^3_α	▼ 0.021 (230.3)	▼ 0.017 (180.1)	▼ 0.020 (181.3)
RP^3_β	▼ 0.007 (869.8)	▼ 0.004 (1068.2)	▼ 0.004 (1145.3)
WRMF	▼ 0.038 (80.9)	▼ 0.030 (60.0)	▽ 0.035 (57.7)

that WRMF accuracy increases as the level of sparsity decreases. In fact, Liang et al. [2016] claim matrix-factorization approaches usually show limited accuracy for short history users, as they cannot accurately infer latent factors for such users. In contrast, Figure 3.4b shows WRMF provides the best recommendations even for short history users on ML 1M, which turns out to conflict with the previous finding at first look. In fact, for users who evaluated up to sixteen items, the average (median) number of evaluations these items have received equals 915 (706) on ML 1M and 29 (52.64) on CDs. On the one hand, this finding suggests WRMF is not able to provide the best recommendations for short-history users on CDs, on the other hand it is able to do so on ML 1M as there are plenty of data to accurately infer latent factors for the items. Likewise, we can draw a similar conclusion for WRMF performance for the first two bins on BX, where the first bin shows better results compared to the second bin as the items in the former have greater average and median number of evaluations than those in the latter.

Table 3.7: Performance comparison of our method and baselines on Kindle dataset.

Method	MAP	P@5	nDCG@5
BIS	▼ 0.065 (167.0)	▼ 0.046 (186.4)	▼ 0.044 (209.9)
BIS-U	▼ 0.083 (109.2) ▲	▼ 0.060 (120.3) ▲	▼ 0.058 (135.3) ▲
BIS-I	▼ 0.134 (29.9) ▲	▼ 0.099 (31.8) ▲	▼ 0.100 (36.0) ▲
BIS-UI	▼ 0.143 (21.9) ▲	▼ 0.106 (23.9) ▲	▼ 0.107 (27.1) ▲
BJS	▼ 0.113 (54.1)	▼ 0.081 (61.0)	▼ 0.083 (64.7)
BJS-U	▼ 0.126 (38.8) ▲	▼ 0.092 (42.7) ▲	▼ 0.094 (44.4) ▲
BJS-I	○ 0.165 (5.7) ▲	○ 0.123 (6.2) ▲	○ 0.126 (7.8) ▲
BJS-UI	○ 0.169 (3.3) ▲	○ 0.127 (3.5) ▲	○ 0.130 (4.5) ▲
BORS	▼ 0.089 (96.1)	▼ 0.065 (102.7)	▼ 0.065 (109.9)
BORS-U	▼ 0.104 (67.2) ▲	▼ 0.077 (70.7) ▲	▼ 0.078 (74.9) ▲
BORS-I	▼ 0.145 (20.2) ▼	▼ 0.109 (20.2) ▼	▼ 0.112 (21.9) ▼
BORS-UI	▼ 0.152 (14.9) ▲	▼ 0.114 (15.0) ▲	▼ 0.117 (16.1) ▲
CIS	▼ 0.122 (42.8)	▼ 0.087 (50.2)	▼ 0.089 (52.7)
CIS-U	▼ 0.131 (33.2) ▲	▼ 0.096 (36.7) ▲	▼ 0.099 (37.8) ▲
CIS-I	○ 0.174 (0.3) ▲	○ 0.131 (0.4) ▲	○ 0.135 (1.0) ▲
CIS-UI	**0.175** ▲	**0.131** ▲	**0.136** ▲
RRW	▼ 0.101 (73.2)	▼ 0.077 (69.2)	▼ 0.079 (73.5)
MP	▼ 0.006 (2642.5)	▼ 0.004 (3256.2)	▼ 0.004 (3360.9)
P^3_α	▼ 0.057 (206.2)	▼ 0.041 (216.5)	▼ 0.042 (225.0)
RP^3_β	▼ 0.027 (543.8)	▼ 0.018 (624.7)	▼ 0.018 (653.6)
WRMF	▼ 0.061 (185.9)	▼ 0.047 (179.7)	▼ 0.048 (184.7)

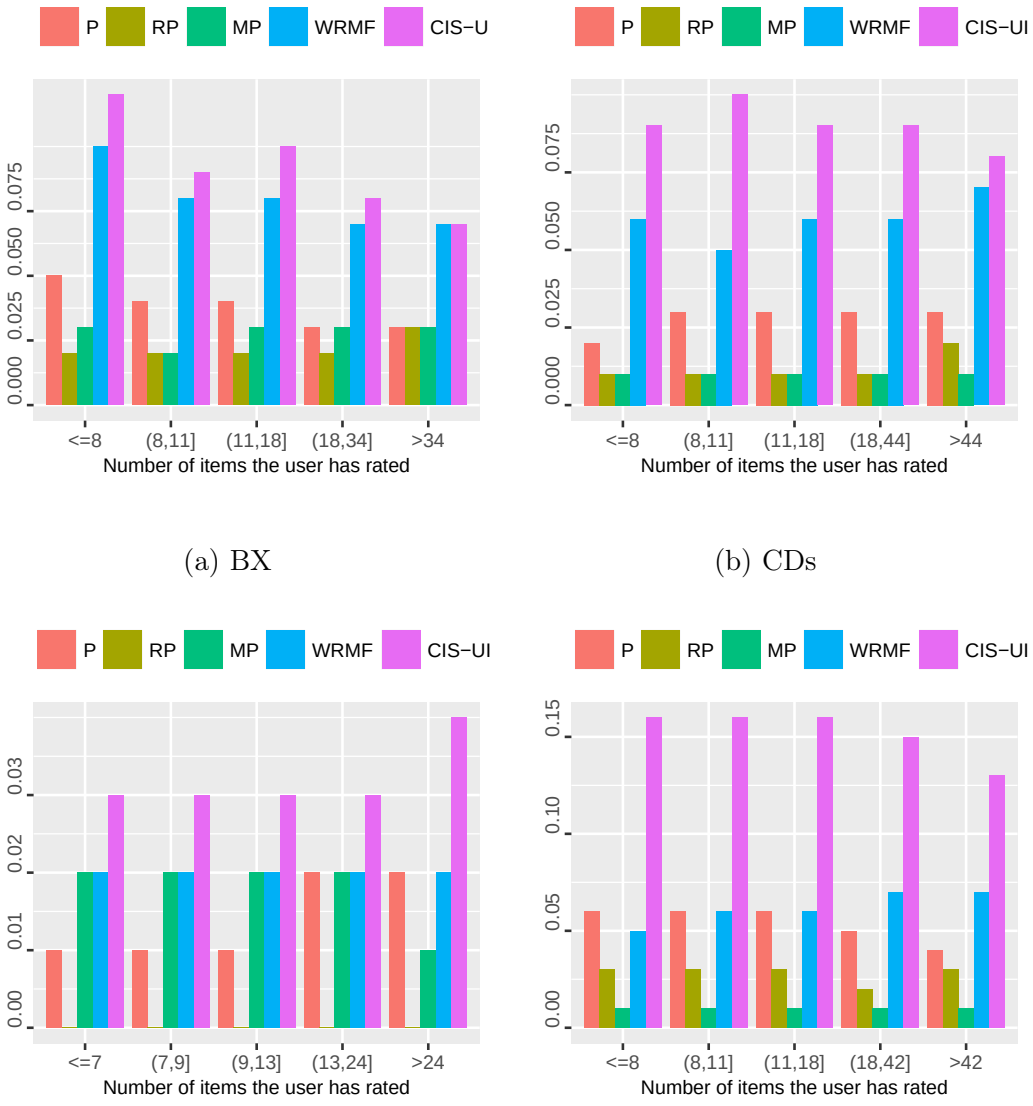
3.5 Discussion

In this chapter, we presented the results regarding our graph-based approaches and baselines in several datasets. We posed three research questions to guide our analysis. Our results show: i) the scaling strategies outperform the Vanilla counterpart, ii) our scoring function based on the Categorical Bayesian Model provides the best results compared to those provided by the scoring function based on the Binary Bayesian Model, iii) our random walk sampling algorithm outperforms the state-of-the art graph-based approaches, which attest the effectiveness of considering the items' rating, iv) our best path enumeration method outperforms our random walking sampling algorithm and baselines in all but one dataset, where a state-of-the-art MF approach provides better recommendations, and v) our best method is able to improve recommendation accuracy in sparser settings. Hence, we confirm our main hypothesis that the set of items' ratings contains a valuable source of latent information that short-length graph-

Table 3.8: Performance comparison of our method and baselines on MovieLens 1M dataset.

Method	MAP	P@5	nDCG@5
BIS	▼ 0.096 (53.1)	▼ 0.152 (52.4)	▼ 0.143 (61.8)
BIS-U	▼ 0.110 (33.5) △	▼ 0.164 (41.1) ▲	▼ 0.157 (46.8) ▲
BIS-I	▼ 0.095 (55.9) ○	▼ 0.159 (45.6) ▲	▼ 0.150 (55.0) ▲
BIS-UI	▼ 0.106 (39.1) ▲	▼ 0.170 (36.2) ▲	▼ 0.161 (43.7) ▲
BJS	▼ 0.116 (26.6)	▼ 0.189 (22.2)	▼ 0.187 (23.4)
BJS-U	○ 0.143 (3.1) ▲	▽ 0.223 (3.9) ▲	○ 0.224 (3.3) ▲
BJS-I	▼ 0.102 (46.9) ▼	▼ 0.175 (33.7) ▼	▼ 0.172 (36.1) ▼
BJS-UI	▼ 0.119 (24.2) ▲	▽ 0.201 (15.8) ▲	▽ 0.199 (16.8) ▲
BORS	▼ 0.111 (32.3)	▼ 0.184 (26.1)	▼ 0.181 (27.5)
BORS-U	▼ 0.130 (12.7) ▲	▼ 0.208 (11.0) ▲	▼ 0.209 (10.6) ▲
BORS-I	▼ 0.102 (46.1) ▼	▼ 0.172 (35.3) ▼	▼ 0.167 (39.6) ▼
BORS-UI	▼ 0.115 (28.5) ▲	▼ 0.192 (20.9) ▲	▼ 0.188 (23.9) ▲
CIS	▼ 0.119 (23.6)	▼ 0.196 (18.0)	▼ 0.193 (20.0)
CIS-U	**0.147** ▲	**0.231** ▲	**0.231** ▲
CIS-I	▼ 0.104 (43.5) ▼	▼ 0.180 (29.9) ▼	▼ 0.177 (32.1) ▼
CIS-UI	▽ 0.123 (20.7) ▲	○ 0.208 (12.2) ▲	▽ 0.206 (13.1) ▲
RRW	▼ 0.116 (26.6)	▼ 0.193 (19.8)	▼ 0.189 (22.2)
MP	▼ 0.097 (50.9)	▼ 0.164 (40.9)	▼ 0.164 (40.9)
P^3_α	▼ 0.097 (51.1)	▼ 0.169 (37.7)	▼ 0.170 (36.2)
RP^3_β	▼ 0.077 (90.9)	▼ 0.131 (76.4)	▼ 0.134 (72.0)
WRMF	▲ 0.230 (-36.0)	▲ 0.318 (-27.2)	▲ 0.327 (-29.4)

based CF should take into account for improving recommendation. Thus, P^3_α and RP^3_β are not suitable for explicit feedback contexts.



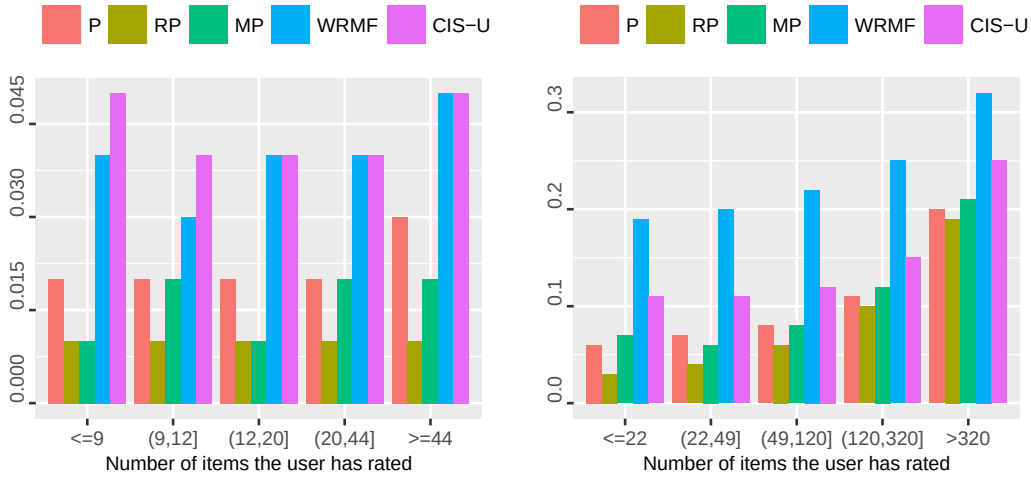
(a) BX

(b) CDs

(c) Electronics

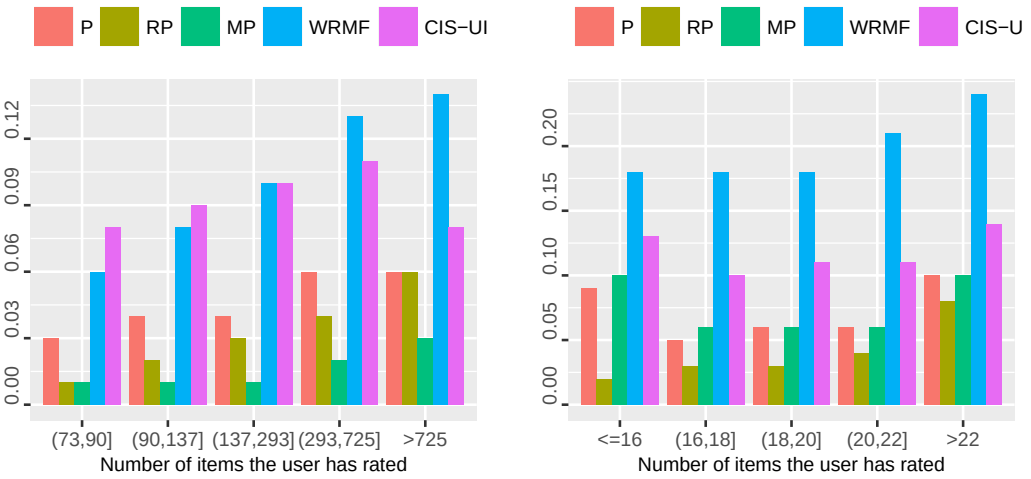
(d) Kindle

Figure 3.2: MAP breakdown for users with various levels of sparsity.



(a) Epinions (b) ML 1M

Figure 3.3: MAP breakdown for users with various levels of sparsity.



(a) CDs (b) ML 1M

Figure 3.4: Zoom into MAP breakdown for users with various levels of sparsity.

Chapter 4

SMF: Similarity-based Matrix Factorization

In this chapter, we present our MF proposal, SMF, to solve the ranking (top-N) formulation of the recommendation problem. SMF simultaneously factorizes the user-item rating matrix and also the user-user and item-item similarity matrices. To this end, we make use of user-user and item-item similarity measures to regularize respectively user and item latent factors. Experiments using publicly available datasets from several recommendation domains demonstrate the effectiveness of SMF, with significant improvements compared to state-of-the-art MF baselines from the literature.

4.1 SMF Model

In explicit feedback domains, we have a set U of users ($m = |U|$), a set I of items ($n = |I|$), and a set V of possible rating values. User-item interactions are represented as a sparse matrix $R = (r_{ui})_{m \times n}$, where $r_{ui} \in V$ denotes user u's rating on item i.

4.1.1 Probabilistic Matrix Factorization

In Probabilistic Matrix Factorization (PMF) [Salakhutdinov and Mnih, 2008], each user u is associated with a user latent vector $p_u \in \mathbb{R}^f$, where $P = (p_u)_{f \times m}$, each item i is associated with an item latent vector $q_i \in \mathbb{R}^f$, where $Q = (q_i)_{f \times n}$, and f is the number of factors. The graphical model of PMF is illustrated in Figure 4.1. The model

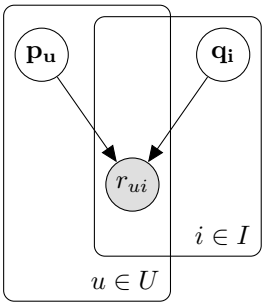


Figure 4.1: Graphical model for PMF.

is formally defined by the following equations:

$$\mathbb{P}(R|P,Q,\sigma^2) = \prod_{u:=1}^{m}\prod_{i:=1}^{n}[\mathcal{N}(r_{ui}|p_u^\top q_i,\sigma^2)]^{1_{ui}} \tag{4.1}$$

$$\mathbb{P}(P|\sigma_P^2\mathrm{E}) = \prod_{u:=1}^{m}\mathcal{N}(p_u|0,\sigma_P^2\mathrm{E}) \tag{4.2}$$

$$\mathbb{P}(Q|\sigma_Q^2\mathrm{E}) = \prod_{i:=1}^{n}\mathcal{N}(q_i|0,\sigma_Q^2\mathrm{E}) \tag{4.3}$$

where 1_{ui} is an indicator function that equals 1 if user u has rated item i or 0 otherwise, and E is the $f \times f$ identity matrix. Maximizing the log-posterior is equivalent to minimizing the following objective function:

$$\begin{aligned}\mathcal{L}_{PMF} = \frac{1}{2}\sum_{u:=1}^{m}\sum_{i:=1}^{n}1_{ui}(r_{ui}-p_u^\top q_i)^2 + \\ \frac{\lambda_P}{2}\sum_{u\in U}\|p_u\|_2^2 + \frac{\lambda_Q}{2}\sum_{i\in I}\|q_i\|_2^2\end{aligned} \tag{4.4}$$

where $\lambda_P = (\sigma^2/\sigma_P^2)$ and $\lambda_Q = (\sigma^2/\sigma_Q^2)$ are regularization parameters to avoid overfitting. After learning P and Q, a missing feedback r_{ui} is estimated as $\hat{r}_{ui} = p_u^\top q_i$.

.

4.1.2 Similarity-based Matrix Factorization

We extend PMF by embedding user-user and item-item similarity matrices to make more accurate recommendations. Given user-user $S_U = (s_{uv})_{m\times m}$ and item-item $S_I = (s_{ij})_{n\times n}$ similarity matrices, we propose to decompose them into the product of latent

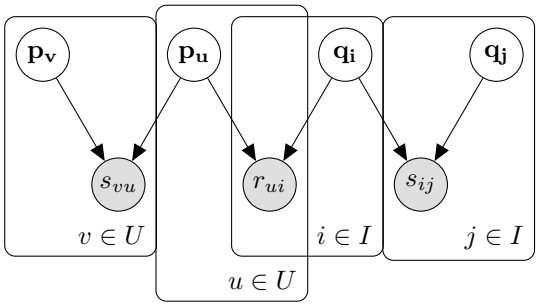


Figure 4.2: Graphical model for SMF.

vectors so that:

$$\mathbb{P}(S_U|P,\sigma^2) = \prod_{u:=1}^{m} \prod_{v:=u+1}^{m} \mathcal{N}(s_{uv}|p_u^\top p_v, \sigma^2) \tag{4.5}$$

$$\mathbb{P}(S_I|Q,\sigma^2) = \prod_{i:=1}^{n} \prod_{j:=i+1}^{n} \mathcal{N}(s_{ij}|q_i^\top q_j, \sigma^2) \tag{4.6}$$

SMF is defined by Equations (4.1)–(4.3), (4.5) and (4.6), with its graphical model shown in Figure 4.2. As with PMF, maximizing the log-posterior is equivalent to minimizing the following objective function:

$$\begin{aligned}\mathcal{L}_{SMF} = & \overbrace{\frac{1}{2}\sum_{u:=1}^{m}\sum_{i:=1}^{n} 1_{ui}\left(r_{ui} - p_u^\top q_i\right)^2}^{\text{MF}} + \\ & \frac{\lambda_P}{2}\sum_{u\in U}\|p_u\|_2^2 + \frac{\lambda_Q}{2}\sum_{i\in I}\|q_i\|_2^2 + \\ & \overbrace{\frac{1}{2}\sum_{u:=1}^{m}\sum_{v:=u+1}^{m}\left(s_{uv} - p_u^\top p_v\right)^2}^{\text{user-user similarity factorization}} + \\ & \overbrace{\frac{1}{2}\sum_{i:=1}^{n}\sum_{j:=i+1}^{n}\left(s_{ij} - q_i^\top q_j\right)^2}^{\text{item-item similarity factorization}}\end{aligned} \tag{4.7}$$

$\mathcal{L}_{SMF}$ only differs from $\mathcal{L}_{PMF}$ in the last two regularization terms, which account for user-user and item-item similarities, respectively. In particular, p_u and q_i are latent factors shared between user-item rating and similarity matrices. The key difference between our model and PMF lies in that the user factors p_u (item factors q_i) account for both user-item interactions and user-user (item-item) similarities. Through the regularization terms, SMF forces not only similar users (items) to have close user

latent vectors but also dissimilar users (items) to have near-orthogonal or near-opposite latent vectors. From Equation (2.6), one can see CoFactor differs from SMF since: i) it makes use of additional item latent vectors (w) and item bias latent vectors (b and c) to factorize the item similarity matrix, ii) it is designed for implicit feedback and so adopt weights (c) to balance unobserved feedbacks, and iii) it only embeds item-item similarities.

4.1.3 Inference

A local minimum of the objective function given by Equation (4.7) can be found by performing Stochastic Gradient Descent (SGD) in P and Q where the gradients are given by:

$$\begin{aligned}\frac{\partial \mathcal{L}_{SMF}}{\partial p_u} = &- \sum_{i:=1}^{n} 1_{ui} \left(r_{ui} - p_u^\top q_i\right) q_i \\ &- \sum_{v \in U: v \neq u} \left(s_{uv} - p_u^\top p_v\right) p_v + \lambda_P \, p_u\end{aligned} \tag{4.8}$$

$$\begin{aligned}\frac{\partial \mathcal{L}_{SMF}}{\partial q_i} = &- \sum_{u:=1}^{m} 1_{ui} \left(r_{ui} - p_u^\top q_i\right) p_u \\ &- \sum_{j \in I: j \neq i} \left(s_{ij} - q_i^\top q_j\right) q_j + \lambda_Q \, q_i\end{aligned} \tag{4.9}$$

The update for each latent factor is performed by moving in the opposite direction of the gradient, which is scaled by a learning rate $\alpha \in \mathbb{R}_{>0}$. In fact, both f, α, λ_P and λ_Q are determined via parameter tuning.

4.1.4 Complexity Analysis and Implementation Details

The computation of the gradients incurs time complexity $O((m+n)^2 \, f)$ in the worst case scenario. Since R is usually very sparse, the computation of the gradients is very fast in practice. Our SGD implementation performs uniform sampling without repetition. The number of epochs is fixed at 100. Finally, we only embed into SMF pairs of users and pairs of items whose similarity values are different from zero.

4.2 Experimental Setup

Our experiments are designed to answer the following research questions:

Q1. How do different user-user and item-item similarity measures impact the effectiveness of SMF?

Q2. How effective is SMF compared to MF approaches with and without similarity regularization?

Q3. How robust is SMF to various degrees of sparsity?

To answer these questions, we use several datasets, baselines and evaluation metrics in our experiments.

Datasets. We make use of the BX, CDs, Epinions and ML 1M datasets, which were previously presented in Section 3.3.1. Since we do not focus on cold start issues, we only keep users and items with at least 10 ratings. Table 3.1 summarizes key properties of these datasets after preprocessing.

Baselines. We compare SMF to PMF [Salakhutdinov and Mnih, 2008], which does not use either user-user or item-item similarity matrices for further regularization. Additionally, we compare our approach to two variants of SMF where: (i) only the item-item similarity matrix is jointly factorized (SMF_I for short), which is strongly related to the CoFactor model proposed by Liang et al. [2016] and discussed in Section 2.3.2, and (ii) only the user-user similarity matrix is jointly factorized (SMF_U as an abbreviation). Thus, we can thoroughly assess the advantage of the joint factorization of both user-user and item-item similarity matrices in SMF when compared to the three baselines.

Evaluation Methodology. Our experiments are conducted using a 5-fold cross validation on each dataset. To this end, we randomly partitioned the datasets into 5 subsets, where a subset T is retained for testing, and the remaining 4 subsets are used as training data. We repeated this process 5 times and report the average result across all the test subsets in 5 trials. We further standardize the users' ratings in each dataset into a relevance scale personalized for each individual user. Precisely, given a user $u \in U$, we define the set $T_u \subset I_u$ of items rated by u and used for testing. We further partition $T_u = T_u^+ \cup T_u^-$, where $T_u^+ = \{i \in I_u | r_{ui} \geq \bar{r}_u\}$ is the set of items used for testing that u has rated above average and $T_u^- = T_u \setminus T_u^+$, in turn, is the set of items used for testing that u has rated below average.

Evaluation Criteria. We adopt the evaluation metrics discussed in Section 3.3.3.

Parameter Tuning. We empirically tune the hyper-parameters for our methods and the baselines for the sake of a fair comparison. We tested values of number of factors $f \in \{10, 20, 50\}$, regularization parameter $\lambda \in \{0.1, 1, 10, 100\}$, where we set $\lambda = \lambda_P = \lambda_Q$, and learning rate $\alpha \in \{0.0001, 0.001, 0.01\}$. We report the results for all methods with their best configuration found with respect to MAP.

4.3 Experimental Results

In the following, we address each of the three research questions posed in Section 4.2 in turn.

Table 4.1: Performance comparison of our method and baselines on BX dataset.

Method	S_U	S_I	Parameters	MAP	P@5	NDCG@5
PMF	–	–	f=50,λ=0.1,α=0.001	▼ 0.017 (353.2)	▼ 0.008 (559.1)	▼ 0.007 (659.2)
SMF_I	–	P	f=50,λ=1,α=0.001	▼ 0.024 (239.7)	▼ 0.012 (315.6)	▼ 0.011 (426.7)
	–	J	f=50,λ=1,α=0.001	▼ 0.047 (68.9)	▼ 0.024 (108.9)	▼ 0.026 (109.9)
	–	C	f=50,λ=1,α=0.001	▼ 0.054 (46.1)	▼ 0.032 (55.1)	▼ 0.032 (71.3)
SMF_U	P	–	f=50,λ=1,α=0.001	▼ 0.039 (104.2)	▼ 0.023 (119.6)	▼ 0.023 (134.7)
	J	–	f=50,λ=1,α=0.001	▼ 0.052 (50.6)	▼ 0.031 (63.1)	▼ 0.033 (62.2)
	C	–	f=50,λ=1,α=0.01	▼ 0.028 (178.1)	▼ 0.017 (201.9)	▼ 0.016 (243.7)
SMF	P	P	f=50,λ=1,α=0.001	▼ 0.037 (112.1)	▼ 0.022 (128.1)	▼ 0.023 (138.8)
	P	J	f=50,λ=1,α=0.001	▼ 0.049 (60.6)	▼ 0.030 (69.2)	▼ 0.031 (73.6)
	P	C	f=50,λ=1,α=0.001	▼ 0.067 (17.6)	▼ 0.043 (17.7)	▼ 0.045 (21.2)
	J	P	f=50,λ=1,α=0.001	▼ 0.044 (76.3)	▼ 0.026 (90.1)	▼ 0.029 (85.5)
	J	J	f=50,λ=1,α=0.001	▼ 0.066 (18.2)	▼ 0.042 (19.4)	▼ 0.047 (15.4)
	J	C	f=50,λ=1,α=0.001	**0.078**	**0.050**	**0.054**
	C	P	f=50,λ=0.1,α=0.0001	▼ 0.024 (232.3)	▼ 0.016 (218.7)	▼ 0.016 (255.9)
	C	J	f=50,λ=1,α=0.001	▼ 0.042 (86.9)	▼ 0.027 (87.1)	▼ 0.029 (86.6)
	C	C	f=50,λ=10,α=0.01	▼ 0.051 (54.2)	▼ 0.032 (59.2)	▼ 0.034 (57.8)

4.3.1 Similarity Choice

To address research question Q1, we assess the impact of each of the three similarity measures described in Section 2.3.1: Cosine (Equation (2.2)), Jaccard (Equation (2.3)) and Pearson (Equation (2.1)). Results for the BX, CDs, Epinions and ML 1M datasets are presented in Tables 4.1–4.4, respectively. In each table, the second and third columns show the similarity measures used to instantiate respectively the user-user (S_U)

Table 4.2: Performance comparison of our method and baselines on CDs dataset.

Method	S_U	S_I	Parameters	MAP	P@5	NDCG@5
PMF	–	–	f=50,λ=0.1,α=0.001	▼ 0.013 (331.1)	▼ 0.008 (466.1)	▼ 0.007 (524.4)
	–	P	f=50,λ=10,α=0.01	▼ 0.006 (875.5)	▼ 0.004 (1129.0)	▼ 0.004 (1122.1)
SMF_I	–	J	f=50,λ=0.1,α=0.001	▼ 0.021 (166.7)	▼ 0.012 (266.1)	▼ 0.012 (270.5)
	–	C	f=50,λ=1,α=0.001	▼ 0.037 (52.4)	▼ 0.028 (60.8)	▼ 0.028 (66.3)
	P	–	f=50,λ=0,0.1,α=0.001	▼ 0.011 (414.4)	▼ 0.008 (464.7)	▼ 0.008 (509.7)
SMF_U	J	–	f=50,λ=0,0.1,α=0.001	▼ 0.028 (99.7)	▼ 0.022 (104.4)	▼ 0.022 (107.5)
	C	–	f=50,λ=0,1,α=0.01	▼ 0.028 (95.8)	▼ 0.022 (105.7)	▼ 0.021 (114.6)
	P	P	f=50,λ=0.1,α=0.0001	▼ 0.006 (803.6)	▼ 0.006 (684.3)	▼ 0.006 (718.4)
	P	J	f=50,λ=1,α=0.001	▼ 0.013 (320.8)	▼ 0.010 (331.8)	▼ 0.010 (352.9)
	P	C	f=50,λ=1,α=0.001	▼ 0.030 (86.3)	▼ 0.026 (72.0)	▼ 0.026 (77.0)
	J	P	f=50,λ=1,α=0.001	▼ 0.007 (656.2)	▼ 0.005 (736.3)	▼ 0.006 (720.9)
SMF	J	J	f=50,λ=1,α=0.001	▼ 0.041 (37.2)	▼ 0.033 (37.9)	▼ 0.033 (37.6)
	J	C	f=50,λ=1,α=0.001	**0.056**	**0.045**	**0.046**
	C	P	f=50,λ=0.1,α=0.0001	▼ 0.010 (493.3)	▼ 0.010 (338.4)	▼ 0.010 (348.2)
	C	J	f=50,λ=0.1,α=0.0001	▼ 0.022 (152.2)	▼ 0.023 (100.0)	▼ 0.023 (97.4)
	C	C	f=50,λ=1,α=0.01	▼ 0.034 (62.9)	▼ 0.030 (50.5)	▼ 0.030 (50.4)

and item-item (S_I) similarity matrices considered in the joint factorization performed by SMF in Equation (4.7). In particular, (i) C stands for Cosine similarity measure, (ii) J stands for Jaccard similarity measure, and (iii) P stands for Pearson similarity measure. Note that no similarity measure is used for the PMF baseline, as it does not perform any similarity-based regularization. For improved reproducibility, the fourth column lists the optimized hyper-parameter settings obtained for each considered model through cross-validation. Lastly, in the remaining three columns, the significance symbols are used to denote a statistically significant difference (or lack thereof) of the result in each row compared to the best result in the column, which is highlighted in bold.

For the ML 1M dataset (Table 4.4), among all possible configurations in SMF where S_U is fixed and S_I varies, in most of the cases, $S_I = J$ shows the best results. In contrast, when S_I is kept fixed and S_U varies, $S_U = J$ provides the best results in most of the configurations. For the BX, CDs and Epinions datasets (Tables 4.1–4.3), $S_U = J$, in turn $S_I = C$, shows the best results when the counterpart similarity is kept fixed. Recalling question Q1, while the Jaccard similarity measure consistently provides the best instantiation for S_U, effective instantiations for S_I are obtained using either the Jaccard or the Cosine measure depending on the considered dataset.

Table 4.3: Performance comparison of our method and baselines on Epinions dataset.

Method	S_U	S_I	Parameters	MAP	P@5	NDCG@5
PMF	–	–	f=50,λ=0.1,α=0.001	▼ 0.005 (643.1)	▼ 0.003 (1247.8)	▼ 0.002 (1451.4)
SMF_I	–	P	f=50,λ=0.1,α=0.0001	▼ 0.003 (1312.5)	▼ 0.002 (1301.0)	▼ 0.003 (1294.0)
	–	J	f=50,λ=1,α=0.001	▼ 0.009 (336.8)	▼ 0.006 (450.4)	▼ 0.006 (510.8)
	–	C	f=50,λ=1,α=0.01	▼ 0.028 (44.9)	▼ 0.022 (52.0)	▼ 0.023 (49.6)
SMF_U	P	–	f=50,λ=0.1,α=0.0001	▼ 0.015 (160.2)	▼ 0.012 (174.1)	▼ 0.012 (194.8)
	J	–	f=50,λ=0.1,α=0.001	▼ 0.020 (103.1)	▼ 0.015 (121.9)	▼ 0.014 (145.7)
	C	–	f=50,λ=1,α=0.0001	▼ 0.012 (243.6)	▼ 0.008 (301.0)	▼ 0.007 (365.3)
SMF	P	P	f=50,λ=0.1,α=0.0001	▼ 0.013 (203.3)	▼ 0.011 (200.0)	▼ 0.011 (223.7)
	P	J	f=50,λ=0.1,α=0.0001	▼ 0.015 (173.4)	▼ 0.012 (179.6)	▼ 0.012 (196.9)
	P	C	f=50,λ=1,α=0.001	▼ 0.021 (91.2)	▼ 0.018 (81.1)	▼ 0.019 (83.7)
	J	P	f=50,λ=0.1,α=0.001	▼ 0.004 (916.8)	▼ 0.002 (1388.0)	▼ 0.002 (1548.3)
	J	J	f=50,λ=0.1,α=0.001	▼ 0.025 (59.7)	▼ 0.023 (47.0)	▼ 0.023 (48.4)
	J	C	f=50,λ=1,α=0.001	**0.040**	**0.033**	**0.035**
	C	P	f=50,λ=0.1,α=0.0001	▼ 0.010 (325.1)	▼ 0.008 (300.2)	▼ 0.007 (375.3)
	C	J	f=50,λ=1,α=0.0001	▼ 0.021 (88.5)	▼ 0.019 (76.3)	▼ 0.019 (82.8)
	C	C	f=50,λ=1,α=0.01	▼ 0.031 (29.8)	▼ 0.027 (25.7)	▼ 0.029 (20.8)

4.3.2 Recommendation Effectiveness

To address research question Q2, we contrast the effectiveness of our proposed SMF model to that of the PMF baseline (Equation (4.4)), which performs no similarity-based regularization. As additional baselines that exploit similarities for jointly factorizing the user-item rating matrix, we consider two SMF variants: SMF_U, which performs user-user similarity regularization, and SMF_I, which performs item-item similarity regularization, and therefore closely resembles the CoFactor regularization approach recently proposed by Liang et al. [2016] and previously discussed in Section 2.3.2.

From Tables 4.1–4.4, we note that the best SMF configuration provides significant gains when compared to PMF across all datasets according to all evaluation metrics. Such gains are of at least 353.2% for BX, 331.1% for CDs, 643.1% for Epinions and 238.7% for ML 1M. Compared to SMF_U, SMF provides gains starting from 50.6% for BX, 95.8% for CDs, 103.1% for Epinions and 33.7% for ML 1M. Finally, compared to SMF_I, our SMF variant that simulates the CoFactor model [Liang et al., 2016], gains start from 46.1% for BX, 52.4% for CDs, 44.9% for Epinions and 51.1% for ML 1M. Recalling research question Q2, these results attest the effectiveness of SMF compared to the standard PMF model. Moreover, they attest the value of exploiting both user-

Table 4.4: Performance comparison of our method and baselines on MovieLens 1M dataset.

Method	S_U	S_I	Parameters	MAP	P@5	NDCG@5
PMF	–	–	f=50,λ=0.1,α=0.01	▼ 0.062 (282.8)	▼ 0.097 (238.7)	▼ 0.086 (273.2)
SMF_I	–	P	f=50,λ=0.1,α=0.0001	▼ 0.106 (122.2)	▼ 0.177 (84.4)	▼ 0.168 (90.3)
	–	J	f=50,λ=0.1,α=0.001	▼ 0.143 (64.6)	▼ 0.216 (51.1)	▼ 0.200 (59.8)
	–	C	f=50,λ=1,α=0.0001	▼ 0.130 (81.2)	▼ 0.213 (53.4)	▼ 0.205 (55.7)
SMF_U	P	–	f=50,λ=0.1,α=0.0001	▼ 0.115 (105.2)	▼ 0.192 (69.7)	▼ 0.181 (76.9)
	J	–	f=50,λ=0.1,α=0.001	▼ 0.163 (44.2)	▼ 0.232 (40.9)	▼ 0.220 (45.1)
	C	–	f=50,λ=1,α=0.0001	▼ 0.161 (45.9)	▼ 0.244 (33.7)	▼ 0.232 (38.0)
SMF	P	P	f=50,λ=0.1,α=0.0001	▼ 0.145 (61.8)	▼ 0.246 (32.7)	▼ 0.234 (36.9)
	P	J	f=50,λ=0.1,α=0.001	▼ 0.151 (55.9)	▼ 0.237 (37.9)	▼ 0.224 (42.6)
	P	C	f=50,λ=0.1,α=0.0001	▼ 0.150 (56.4)	▼ 0.245 (33.1)	▼ 0.231 (38.2)
	J	P	f=50,λ=0.1,α=0.001	▼ 0.214 (10.2)	▼ 0.309 (5.5)	▼ 0.304 (5.3)
	J	J	f=50,λ=0.1,α=0.001	**0.235**	**0.326**	**0.320**
	J	C	f=50,λ=0.1,α=0.001	▼ 0.204 (15.1)	▼ 0.302 (8.1)	▼ 0.286 (11.6)
	C	P	f=50,λ=1,α=0.0001	▼ 0.145 (61.9)	▼ 0.166 (96.9)	▼ 0.151 (112.3)
	C	J	f=50,λ=1,α=0.0001	▼ 0.195 (20.4)	▼ 0.291 (12.2)	▼ 0.277 (15.4)
	C	C	f=50,λ=1,α=0.001	▼ 0.124 (89.4)	▼ 0.206 (58.4)	▼ 0.203 (57.7)

user and item-item similarities conjointly when factorizing user-item ratings.

4.3.3 Recommendation Robustness

To address question Q3, we assess the robustness of SMF when recommending for users with various levels of sparsity. Figures 4.3a-4.3d provide performance breakdowns (in terms of MAP) for PMF, all configurations of SMF_U and SMF_I, and the best configuration of SMF in all four datasets. Each figure shows five bins representing user groups that rated an increasing amount of items organized according to the 10th, 35th, 65th and 90th percentiles of the corresponding dataset. In the figures, SMF_U, SMF_I, and SMF are denoted by two letters, the first representing the measure used for user-user similarities, and the second the measure used for item-item similarities. Once again, 'P' stands for Pearson, 'J' for Jaccard, and 'C' for Cosine. For example, 'J–' denotes SMF_U with user-user Jaccard similarities (and no item-item similarities), '–P' denotes SMF_I with item-item Pearson similarities (and no user-user similarities), and 'JC' denotes the full SMF model with user-user Jaccard similarities and item-item Cosine similarities.

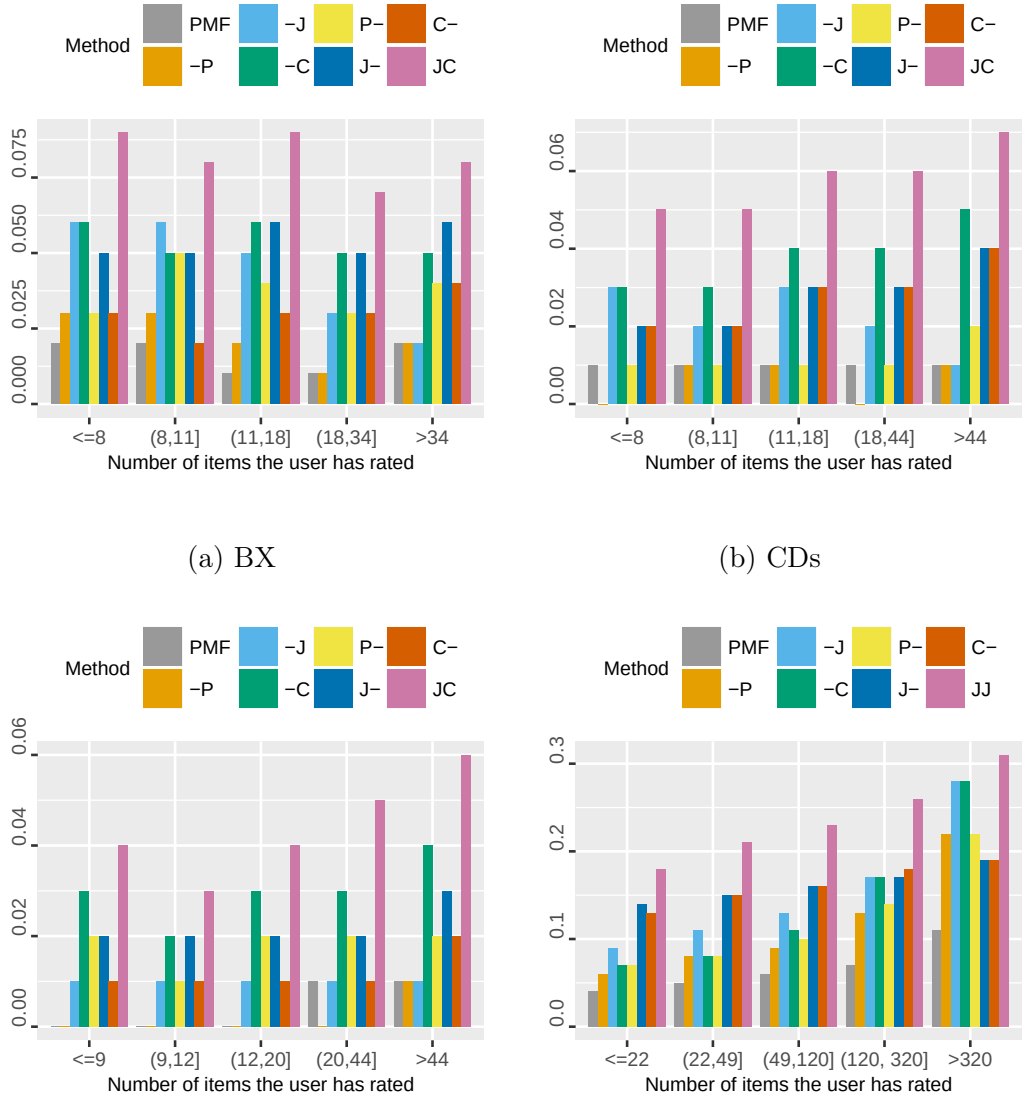
From Figures 4.3a-4.3d, one can clearly see that SMF produces the best recom-

mendations in all the bins for the ML 1M, CDs, BX, and Epinions datasets. PMF, in turn, cannot accurately infer latent factors for short history users, especially in the Epinions dataset (Figure 4.3c) where its accuracy significantly increases only in the fourth bin. Additionally, SMF outperforms both SMF_U and SMF_I in all the bins for all datasets. Therefore, making use of additional information from both user-user and item-item similarity matrices plays a key role in improving recommendations, especially for users who have consumed a small number of items. Recalling Q3, we conclude that SMF is robust to the degree of sparsity of different users, with consistent improvements across all considered datasets.

Despite the consistent improvements observed across most of the user base for all four datasets in Figures 4.3a-4.3d, an interesting observation can be drawn by zooming into highly active users on the ML 1M dataset, which account for 3.86% of the users in the dataset. As shown in Figure 4.4, SMF provides the best results only in the first bin. In fact, the best SMF_I configuration outperforms SMF in the last four bins, with SMF results lying between the results for the best configurations of SMF_U and SMF_I. This finding suggests that embedding user-user similarity for very active users into SMF may in fact introduce noise and degrade recommendation accuracy for highly active users as the preferences for such users are well defined. While these users account for the minority of the user base, such an accuracy degradation might be alleviated by a suitably defined weighting strategy in the user-user similarity-based regularization.

4.4 Discussion

In this chapter, we presented the results regarding SMF and baselines in several datasets. We posed three research questions to guide our analysis. Our results show: i) SMF is sensitive to the choice of user-user and item-item similarity measures, so similarity measures have to be suitably chosen for each dataset, ii) the joint factorization of both user-user and item-item similarity matrices provides significant gains compared to the canonical MF and scenarios where either user-user or item-item similarity matrix is jointly decomposed, which attest the value of incorporating both user-user and item-item similarities into MF and iii) SMF provides gains for users with various levels of sparsity, but it might introduce noise and then degrade recommendation accuracy for highly-active users as shown in the ML 1M dataset. Hence, we confirm our hypothesis that MF is able to produce more accurate recommendations by embedding both user-user and item-item similarities.



(a) BX

(b) CDs

(c) Epinions

(d) ML 1M

Figure 4.3: MAP breakdown for users with various levels of sparsity.

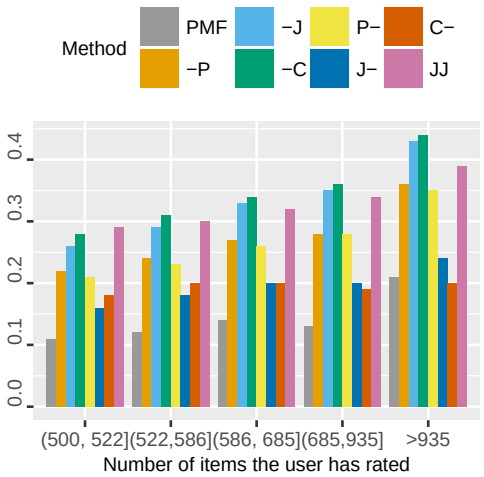


Figure 4.4: MAP breakdown for active users on ML 1M.

Chapter 5

Conclusion and Future Work

5.1 Concluding Remarks

In this work, we studied ranking algorithms for personalized recommendation. In the context of graph-based approaches, we proposed novel algorithms and scoring methods based on the Bayesian paradigm that exploit distributional aspects of ratings. Our methods make use of three-step paths starting from the target user in the user-item graph to score items. The first method relies on random walk simulations and takes into account the ratings for sampling items in the process. In contrast, the second method enumerates all three-step paths, which are in turn scored by means of a Bayesian scoring function; we propose to consider user and/or item similarity measures to boost recommendation accuracy. In addition, we show our path enumeration strategy is more efficient than the matrix multiplication strategy.

In the context of MF approaches, we proposed a MF model that jointly decomposes the user-item rating matrix and both user-user and item-item similarity matrices. In fact, SMF makes use of shared latent factors to decompose such matrices. Different from most works found in the literature, our approach makes no use of additional data, which makes it applicable to a wider range of scenarios where no further information about individual users or items is available. In both contexts, we carried out experiments using several publicly available datasets to assess the effectiveness of the recommendation methods.

In the context of graph-based recommendation, we empirically showed that our methods clearly outperform the graph-based approaches considered in all datasets and metrics. We also compared our methods against a representative matrix factorization method and the results show our best method provides superior recommendations in all but one dataset. Our path enumeration method clearly benefits from the intro-

Table 5.1: Comparison between CIS and WRMF for each dataset.

Data Set	CIS-U	CIS-UI	WRMF
BX	✓		
CDs		✓	
Electronics		✓	
Epinions	✓		
Kindle		✓	
ML 1M			✓

duction of similarity measures; one can see that our method is flexible in that other similarity measures can be easily embedded depending on the context. In turn, our random walk method benefits from the set of ratings to produce more accurate recommendations than related baselines. Thus, we confirm our specific hypothesis that short-length graph-based approaches can take advantage of the set of ratings and that these approaches can benefit from the introduction of similarity measures. Likewise, our work shows the power of graph-based approaches, especially considering the fact that these approaches have been neglected since matrix factorization has been considered the main technique in CF. Finally, our results show that there is no single silver bullet for all datasets, providing yet another example of the no free lunch theorem from [Wolpert, 1996]; Table 5.1 illustrates the best method for each dataset. Thus, one has to try out several approaches to determine the method that yields the best results for each dataset. However, our experiments suggest our methods are more suitable for sparser contexts and datasets, which provides another example of the fact graph-based approaches are more effective for sparse ratings matrices [Aggarwal, 2016].

In the context of MF recommendation, we showed that SMF improves recommendation accuracy in all datasets and metrics compared to the baselines. To be precise, we compared SMF against the canonical MF and two variants that jointly factorize either user-user or item-item similarity matrix. Thus, we attest the value of embedding both user-user and item-item similarity matrices into the MF approach. On the other hand, we empirically showed that SMF potentially degrades recommendation accuracy for highly-active users, which might require suitably designed strategies to cope with this issue. For instance, while these users account for the minority of the user base, such an accuracy degradation might be alleviated by a suitably defined weighting strategy in the user-user similarity-based regularization.

In this dissertation we proposed graph-based and MF algorithms that consider similarities to improve recommendation accuracy. Since these approaches greatly differ,

Table 5.2: Comparison between CIS and SMF for each dataset.

Data Set	CIS	SMF
BX	✓	
CDs	✓	
Epinions	✓	
ML 1M		✓

we compare the best results for CIS (graph-based approach) and SMF (MF approach) where Table 5.2 shows the best choice for each dataset. One can see our graph-based approach is the best choice for the sparser datasets, while our MF approach stands out in denser settings. As a result, our findings provide another example of the fact that graph-based approaches are more effective for sparse ratings matrices whereas MF approaches are more effective for dense scenarios. However, although graph-based approaches are not suitable for dense contexts, they might be useful in online recommendation as they do not require an offline learning step. Finally, while the best results of our graph-based approach are sensitive to the similarity strategy, our MF approach shows the best results when embedding both user-user and item-item similarities.

5.2 Future Directions

In this section, we describe next activities that can be carried out. Regarding our graph-based approaches, we envision some future research avenues:

- Graph-based CF in the context of cross-domain recommendations. We hope structural information from the user-item graph and our Bayesian scoring functions may improve recommendation accuracy in a cross-domain scenarios. However, several issues have to be addressed. First, the user-item bipartite graph may not be suitable for representing user-item interactions, then we may investigate multivariate graphs or even model item-item interactions. Additionally, our Bayesian scoring functions may exploit distributional aspects between items in different domains, similarity measures or kernel functions suitably designed for cross-domain may be proposed.

- Quality measures like novelty, diversity and unexpectedness have been moving into the focus of researchers in recent years [Jannach et al., 2016]. In this context, a promising direction includes the proposition of novel scoring functions that take both novelty and diversity into account.

- Graph-based approaches are more effective for sparse ratings matrices whereas MF approaches are more effective for dense matrices. Since our graph-based approach outperforms WRMF in several datasets, a possible direction includes the proposition of a hybrid recommender that accounts for both methods or even the embedding of our scoring function values into MF.

Regarding SMF, we describe some future work:

1. Preference-based filtering techniques focus on predicting the correct relative order of the items rather than their individual ratings. In this context, a promising direction includes the joint factorization of similarity matrices into preference-based MF, such as BPRMF.

2. Contextualization has also become a common feature in real applications [Jannach et al., 2016]. Thus, we can have SMF incorporate context and side information to produce more accurate recommendation.

Bibliography

Adams, R., Dahl, G., and Murray, I. (2010). Incorporating side information in probabilistic matrix factorization with gaussian processes. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, pages 1--9. AUAI Press.

Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734--749.

Agarwal, D. and Chen, B.-C. (2009). Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 19--28. ACM.

Aggarwal, C. C. (2016). *Recommender Systems: The Textbook*. Springer International Publishing, Switzerland.

Aggarwal, C. C., Wolf, J. L., Wu, K.-L., and Yu, P. S. (1999). Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 201--212, New York, NY, USA. ACM.

Baeza-Yates, R. A. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Baluja, S., Seth, R., Sivakumar, D., Jing, Y., Yagnik, J., Kumar, S., Ravichandran, D., and Aly, M. (2008). Video suggestion and discovery for youtube: Taking random walks through the view graph. In *Proceedings of the 17th International Conference on World Wide Web*, pages 895--904, New York, NY, USA. ACM.

Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 43--52, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Christakopoulou, K. and Banerjee, A. (2015). Collaborative ranking with a push at the top. In *Proceedings of the 24th International Conference on World Wide Web*, pages 205--215, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Christoffel, F., Paudel, B., Newell, C., and Bernstein, A. (2015). Blockbusters and wallflowers: Accurate, diverse, and scalable recommendations with random walks. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 163–170, New York, NY, USA. ACM.

Cook, J. (2005). Exact calculation of beta inequalities. Technical report, Department of Biostatistics.

Cooper, C., Lee, S. H., Radzik, T., and Siantos, Y. (2014). Random walks in recommender systems: Exact computation and simulations. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 811–816, New York, NY, USA. ACM.

Cremonesi, P., Koren, Y., and Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the 4th ACM Conference on Recommender Systems*, pages 39–46, New York, NY, USA. ACM.

Das, A. S., Datar, M., Garg, A., and Rajaram, S. (2007). Google news personalization: Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*, pages 271--280, New York, NY, USA. ACM.

David, E. and Jon, K. (2010). *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, New York, NY, USA.

Desrosiers, C. and Karypis, G. (2011). *A Comprehensive Survey of Neighborhood-based Recommendation Methods*, pages 107--144. Springer US, Boston, MA, USA.

Fouss, F., Pirotte, A., Renders, J.-m., and Saerens, M. (2007). Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369.

Fouss, F., Pirotte, A., and Saerens, M. (2005). A novel way of computing similarities between nodes of a graph, with application to collaborative recommendation. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 550–556, Washington, DC, USA. IEEE Computer Society.

Galassi, M. et al. (2009). *GNU Scientific Library Reference Manual*. Network Theory Ltd., 3rd edition.

Gantner, Z., Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. (2011). Mymedialite: A free recommender system library. In *Proceedings of the 5th ACM Conference on Recommender Systems*, pages 305–308, New York, NY, USA. ACM.

Gori, M. and Pucci, A. (2007). Itemrank: A random-walk based scoring algorithm for recommender engines. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2766–2771, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Harper, F. M. and Konstan, J. A. (2015). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):1–19.

Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 230--237, New York, NY, USA. ACM.

Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 263–272, Washington, DC, USA. IEEE Computer Society.

Jamali, M. and Ester, M. (2009). Trustwalker: A random walk model for combining trust-based and item-based recommendation. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 397--406, New York, NY, USA. ACM.

Jamali, M. and Ester, M. (2010). A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the 4th ACM Conference on Recommender Systems*, pages 135--142. ACM.

Jannach, D., Resnick, P., Tuzhilin, A., and Zanker, M. (2016). Recommender systems — beyond matrix completion. *Communications of the ACM*, 59(11):94--102.

Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. (2010). *Recommender Systems: An Introduction*. Cambridge University Press, New York, NY, USA.

Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Lee, J., Bengio, S., Kim, S., Lebanon, G., and Singer, Y. (2014). Local collaborative ranking. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 85--96, New York, NY, USA. ACM.

Lee, S., Park, S., Kahng, M., and Lee, S.-g. (2012). Pathrank: A novel node ranking measure on a heterogeneous graph for recommender systems. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 1637--1641, New York, NY, USA. ACM.

Liang, D., Altosaar, J., Charlin, L., and Blei, D. M. (2016). Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of the 10th ACM Conferece on Recommender Systems*, pages 59--66, New York, NY, USA. ACM.

Linden, G., Smith, B., and York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76--80.

Lopes, R., Assunção, R., and Santos, R. L. T. (2016). Efficient bayesian methods for graph-based recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 333--340, New York, NY, USA. ACM.

Lops, P., de Gemmis, M., and Semeraro, G. (2011). *Content-based Recommender Systems: State of the Art and Trends*, pages 73--105. Springer US, Boston, MA, USA.

Ma, H., King, I., and Lyu, M. R. (2009). Learning to recommend with social trust ensemble. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 203--210. ACM.

Ma, H., Yang, H., Lyu, M. R., and King, I. (2008). SoRec: Social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, pages 931--940. ACM.

Massa, P. and Avesani, P. (2007). Trust-aware recommender systems. In *Proceedings of the 1st ACM Conference on Recommender Systems*, pages 17--24, New York, NY, USA. ACM.

McAuley, J., Pandey, R., and Leskovec, J. (2015). Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM.

Norris, J. (1998). *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab.

Park, Y. and Tuzhilin, A. (2008). The long tail of recommender systems and how to leverage it. In *Proceedings of the 2nd ACM Conference on Recommender Systems*, pages 11–18, New York, NY, USA. ACM.

Paudel, B., Christoffel, F., Newell, C., and Bernstein, A. (2016). Updatable, accurate, diverse, and scalable recommendations for interactive applications. *ACM Transactions on Interactive Intelligent Systems*, 7(1):1--34.

Poole, D. (2006). *Linear Algebra: A Modern Introduction*. Thomson Brooks/Cole.

Porteous, I., Asuncion, A., and Welling, M. (2010). Bayesian matrix factorization with side information and dirichlet process mixtures. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 563--568. AAAI Press.

Rajaraman, A. and Ullman, J. D. (2011). *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA.

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2009). Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pages 452–461, Arlington, Virginia, United States. AUAI Press.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, pages 175--186, New York, NY, USA. ACM.

Ricci, F., Rokach, L., and Shapira, B. (2011). *Introduction to Recommender Systems Handbook*, pages 1--35. Springer US, Boston, MA, USA.

Salakhutdinov, R. and Mnih, A. (2008). Probabilistic matrix factorization. In *Proceedings of The Conference on Neural Information Processing Systems*.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, pages 285--295, New York, NY, USA. ACM.

Sarwar, B. M., Karypis, G., Konstan, J., and Riedl, J. T. (2000). Application of Dimensionality Reduction in Recommender System – A Case Study. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Workshop on Web Mining for E–Commerce – Challenges and Opportunities*, Boston, MA, USA.

Schafer, J. B., Konstan, J., and Riedl, J. (1999). Recommender systems in e-commerce. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 158--166, New York, NY, USA. ACM.

Schafer, J. B., Konstan, J. A., and Riedl, J. (2001). E-commerce recommendation applications. *Data Mining Knowledge Discovery.*, 5(1-2):115--153.

Shan, H. and Banerjee, A. (2010). Generalized probabilistic matrix factorizations for collaborative filtering. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, pages 1025--1030. IEEE Computer Society.

Shani, G. and Gunawardana, A. (2011). *Recommender Systems Handbook*, chapter Evaluating Recommendation Systems, pages 257–297. Springer US, Boston, MA.

Shardanand, U. and Maes, P. (1995). Social information filtering: Algorithms for automating “word of mouth”. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 210--217, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Oliver, N., and Hanjalic, A. (2012). Climf: Learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the 6th ACM Conference on Recommender Systems*, pages 139–46, New York, NY, USA. ACM.

Shi, Y., Larson, M., and Hanjalic, A. (2013). Mining contextual movie similarity with matrix factorization for context-aware recommendation. *ACM Transactions on Intelligent Systems and Technology*, 4(1):1--19.

Singh, A. P. and Gordon, G. J. (2008). Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 650--658. ACM.

Singh, A. P., Gunawardana, A., Meek, C., and Sudendran, A. C. (2007). Recommendations using absorbing random walks. In *North East Student Colloquium on Artificial Intelligence*.

Sinha, R. R. and Swearingen, K. (2001). Comparing recommendations made by online systems and friends. In *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*.

Wei, K., Huang, J., and Fu, S. (2007). A survey of e-commerce recommender systems. In *International Conference on Service Systems and Service Management*, pages 1--5.

Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341--1390.

Xiang, L., Yuan, Q., Zhao, S., Chen, L., Zhang, X., Yang, Q., and Sun, J. (2010). Temporal recommendation on graphs via long- and short-term preference fusion. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 723--732, New York, NY, USA. ACM.

Yang, B., Lei, Y., Liu, D., and Liu, J. (2013). Social collaborative filtering by trust. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence*, pages 2747--2753. AAAI Press.

Yao, Y., Tong, H., Yan, G., Xu, F., Zhang, X., Szymanski, B. K., and Lu, J. (2014). Dual-regularized one-class collaborative filtering. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 759--768. ACM.

Yildirim, H. and Krishnamoorthy, M. S. (2008). A random walk method for alleviating the sparsity problem in collaborative filtering. In *Proceedings of the 2nd ACM Conference on Recommender Systems*, pages 131--138, New York, NY, USA. ACM.

Ziegler, C., McNee, S. M., Konstan, J. A., and Lausen, G. (2005). Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, pages 22--32, New York, NY, USA. ACM.