

Relevance Models for Collaborative Filtering

Relevance Models for Collaborative Filtering

Proefschrift

ter verkrijging van de graad van doctor

aan de Technische Universiteit Delft,

op gezag van de Rector Magnificus prof.dr.ir. J. T. Fokkema,

voorzitter van het College voor Promoties,

in het openbaar te verdedigen op maandag 7 april 2008 om 12:30 uur

door Jun Wang

MSc in Computer Science from National University of Singapore, Singapore,

Bachelor in Electrical Engineering from Southeast University, China,

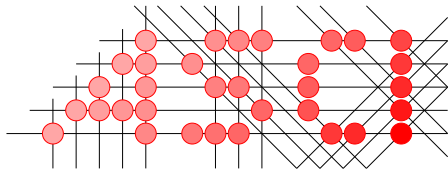
geboren te Jiangsu, China.

Dit proefschrift is goedgekeurd door de promotor:
Prof.dr.ir. M.J.T. Reinders

Toegevoegd promotor:
Dr.ir. A.P. de Vries

Samenstelling promotiecommissie:

| | |
|------------------------------|---|
| Rector Magnificus, | voorzitter |
| Prof.dr.ir. M.J.T. Reinders, | Technische Universiteit Delft, promotor |
| Dr.ir. A.P. de Vries, | CWI, toegevoegd promotor |
| Prof.dr.ir. G. Jongbloed, | Technische Universiteit Delft |
| Prof.dr. S.E. Robertson, | Microsoft Research, Cambridge, UK |
| Prof.dr.ir. M. van Steen, | Vrije Universiteit |
| Prof.dr. B. Berendt, | Katholieke Universiteit Leuven, Belgium |
| Dr.ir. D. Hiemstra, | Universiteit Twente |



Advanced School for Computing and Imaging

This work was carried out in the ASCI graduate school; ASCI dissertation series number 158. The research reported in this thesis was financed by the CACTUS and I-Share projects.

ISBN 978-90-9022932-4

Copyright © 2007 by Jun Wang

All rights reserved. No part of the material protected by this copyright notice may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without written permission from the copyright owner.

The Master said,
“When I walk along with two others,
they may serve me as my teachers.
I will select their good qualities and follow them,
their bad qualities and avoid them.”
The *Lunyu*: Book VII *Shu’er*
Confucius, 551 BCE - 479 BCE

to my family
for making it possible

Summary

Collaborative filtering is the common technique of predicting the interests of a user by collecting preference information from many users. Although it is generally regarded as a key information retrieval technique, its relation to the existing information retrieval theory is unclear. This thesis shows how the development of collaborative filtering can gain many benefits from information retrieval theories and models. It brings the notion of relevance into collaborative filtering and develops several relevance models for collaborative filtering. Besides dealing with user profiles that are obtained by explicitly asking users to rate information items, the relevance models can also cope with the situations where user profiles are implicitly supplied by observing user interactions with a system. Experimental results complement the theoretical insights with improved recommendation accuracy for both item relevance ranking and user rating prediction. Furthermore, the approaches are more than just analogy: our derivations of the unified relevance model show that popular user-based and item-based approaches represent only a partial view of the problem, whereas a unified view that brings these partial views together gives better insights into their relative importance and how retrieval can benefit from their combination.

Samenvatting

Collaborative filtering is een bekende techniek om de interesses van een gebruiker te voorspellen aan de hand van gegevens over de voorkeuren van een grote groep gebruikers. De relatie tussen collaborative filtering en de algemeen geaccepteerde theorie voor information retrieval is echter grotendeels onbekend. Dit proefschrift toont aan dat information retrieval modellen een grote bijdrage kunnen leveren aan de ontwikkeling van collaborative filtering. Het introduceert het begrip relevantie in collaborative filtering, en ontwikkelt vervolgens een reeks modellen voor dit relevantiebegrip. Naast gebruikersprofielen gebaseerd op expliciete voorkeursinformatie, verkregen door gebruikers te vragen objecten te beoordelen, kunnen deze modellen ook gebruik maken van impliciete indicatoren van relevantie op basis van waarnemingen over de interactie van gebruikers met objecten. Behalve theoretische inzichten, wijzen de experimentele resultaten uit dat deze modellen de gebruikersinteresse nauwkeuriger voorspellen, alsmede relevantere objecten aanbevelen. Bovendien blijkt uit de afleidingen in het proefschrift dat collaborative filtering met information retrieval theorie niet slechts een mooie metafoor is. De reeds bekende gebruikers- en objectbenaderingen van collaborative filtering blijken slechts een deel van het probleem te beschrijven, terwijl een verenigde kijk op beide deelbenaderingen een beter inzicht geeft in hun relatieve belang en hoe retrieval kan profiteren van hun combinatie.

Contents

| | |
|--|------------|
| Summary | vii |
| Samenvatting | ix |
| 1 Introduction | 1 |
| 1.1 Scope | 3 |
| 1.1.1 Collaborative Filtering Scenarios | 4 |
| 1.1.2 Putting Relevance into Collaborative Filtering | 4 |
| 1.1.3 Issues in Collaborative Filtering | 6 |
| 1.2 Thesis Outline | 7 |
| 1.3 Main Contributions | 8 |
| I Relevance Models | 11 |
| 2 Language Modelling Approaches | 13 |
| 2.1 Introduction | 14 |
| 2.2 Background | 14 |
| 2.2.1 Rating-based Collaborative Filtering | 14 |
| 2.2.2 Log-based Collaborative Filtering | 15 |
| 2.3 A User-Item Relevance Model | 17 |

| | | |
|--------------------------------|---|-----------|
| 2.3.1 | Item-based Generation | 18 |
| | Probability Estimation and Smoothing | 19 |
| | Linear Interpolation Smoothing | 19 |
| 2.3.2 | User-based Generation | 20 |
| 2.3.3 | Discussions | 21 |
| | Inverse Item Frequency | 21 |
| | Two Representations | 22 |
| 2.4 | Experiments | 22 |
| 2.5 | Conclusions | 25 |
| Commentary on Chapter 2 | | 27 |
| 3 | Probabilistic Relevance Ranking | 31 |
| 3.1 | Introduction | 32 |
| 3.2 | Related Work | 33 |
| 3.2.1 | Rating Prediction | 33 |
| 3.2.2 | Item Ranking | 34 |
| 3.3 | A Probabilistic Relevance Ranking Framework | 35 |
| 3.3.1 | Item-Based Relevance Model | 36 |
| | 3.3.1.1 Probability Estimation | 38 |
| 3.3.2 | User-Based Relevance Model | 43 |
| 3.3.3 | Discussions | 44 |
| 3.4 | Experiments | 44 |
| 3.4.1 | Data Sets | 44 |
| 3.4.2 | Experiment Protocols | 45 |
| 3.4.3 | Performance | 46 |
| 3.4.4 | Parameter Estimation | 49 |
| 3.5 | Conclusions | 50 |
| 3.A | The Okapi BM25 Document Ranking Score | 53 |

| | |
|---|-----------|
| Commentary on Chapter 3 | 55 |
| 4 Personalized Collaborative Tagging | 57 |
| 4.1 Introduction | 58 |
| 4.2 Related Work | 60 |
| 4.3 Personalization Models | 61 |
| 4.3.1 Indexing Phase | 63 |
| 4.3.1.1 Collaborative Indexing Model | 63 |
| 4.3.2 Exploratory Search Phase | 65 |
| 4.3.2.1 Collaborative Browsing Model | 65 |
| 4.3.2.2 Collaborative Item Search Model | 66 |
| 4.3.3 Discussions | 67 |
| 4.4 Experiments | 68 |
| 4.4.1 Data Set Preparation | 68 |
| 4.4.2 Evaluation Protocols | 69 |
| 4.4.3 Performance of Personalization Models | 71 |
| 4.4.4 Representation of User Profiles | 72 |
| 4.4.5 Impact of Parameters | 74 |
| 4.5 Conclusions | 76 |
| 4.A Probability Estimation | 77 |
| Commentary on Chapter 4 | 79 |
| II Unified Models | 81 |
| 5 Similarity Fusion | 83 |
| 5.1 Introduction | 84 |
| 5.2 Related Work | 85 |
| 5.3 Background | 86 |
| 5.3.1 User-based Collaborative Filtering | 86 |

| | | |
|----------|--|------------|
| 5.3.2 | Item-based Collaborative Filtering | 88 |
| 5.4 | Similarity Fusion | 89 |
| 5.4.1 | Individual Predictors | 89 |
| 5.4.2 | Probabilistic Fusion Framework | 90 |
| 5.4.3 | Probability Estimation | 92 |
| 5.4.4 | Discussions | 93 |
| 5.5 | Empirical Evaluation | 94 |
| 5.5.1 | Experimental Setup | 94 |
| 5.5.2 | Individual Predictors | 96 |
| 5.5.3 | Impact of Parameters | 96 |
| 5.5.4 | Data Sparsity | 97 |
| 5.5.5 | Comparison to Other Methods | 99 |
| 5.6 | Conclusions | 100 |
| 5.A | Normalization | 100 |
| 5.B | A Unified Weighting Function | 101 |
| 6 | Unified Relevance Models | 103 |
| 6.1 | Introduction | 104 |
| 6.2 | Related Work | 106 |
| 6.2.1 | Collaborative Filtering | 106 |
| 6.2.2 | Probabilistic Models for Information Retrieval | 107 |
| 6.3 | Background | 109 |
| 6.3.1 | User-based Collaborative Filtering | 110 |
| 6.3.2 | Item-based Collaborative Filtering | 111 |
| 6.3.3 | Combining User-based and Item-based Approaches | 112 |
| 6.4 | Probabilistic Relevance Prediction Models | 113 |
| 6.4.1 | Three Different Relevance Models | 114 |
| 6.4.1.1 | User-based Relevance Model | 115 |
| 6.4.1.2 | Item-based Relevance Model | 116 |

| | | |
|--------------------------------------|--|------------|
| 6.4.1.3 | Unified Relevance Model | 116 |
| 6.4.2 | Probability Estimation | 117 |
| 6.4.2.1 | Density Estimation for Rating Models | 117 |
| 6.4.2.2 | Density Estimation for Preference Models | 118 |
| 6.4.3 | Rating Predictions | 120 |
| 6.4.4 | Cross-validated EM algorithm | 122 |
| 6.4.5 | A Generalised Distance Measure | 122 |
| 6.4.6 | Discussions | 125 |
| 6.4.7 | Computational Complexity | 127 |
| 6.4.7.1 | Offline Computation | 128 |
| 6.4.7.2 | Online Computation | 129 |
| 6.5 | Experiments | 130 |
| 6.5.1 | Data Sets | 130 |
| 6.5.2 | Evaluation Protocols | 130 |
| 6.5.3 | Results | 131 |
| 6.5.3.1 | Parameter Estimation | 131 |
| 6.5.3.2 | Sparsity | 134 |
| 6.5.3.3 | Comparison to other approaches | 140 |
| 6.6 | Conclusions and Future Work | 142 |
| 6.A | Cross-validated EM algorithm | 144 |
| Commentary on Chapter 6 | | 149 |
| III Applications | | 151 |
| 7 Peer-to-Peer Recommendation | | 153 |
| 7.1 | Introduction | 154 |
| 7.2 | Related Work | 156 |
| 7.2.1 | Collaborative Filtering | 156 |

| | | |
|----------|--|------------|
| 7.2.2 | Peer-to-Peer Networks | 156 |
| 7.3 | User-oriented Overlay Network | 157 |
| 7.3.1 | User-based Ranking Model | 157 |
| 7.3.2 | Decentralized Ranking | 159 |
| 7.4 | Item-oriented Overlay Network | 163 |
| 7.4.1 | Item-based Ranking Model | 163 |
| 7.4.1.1 | Incorporation of User Profiles | 164 |
| 7.4.2 | Self-organizing Distributed Buddy Tables | 165 |
| 7.4.2.1 | Dynamically Updating Relevance Ranks | 165 |
| 7.4.2.2 | Distributed Item-to-Item Relevance Ranking | 167 |
| 7.4.3 | Distributed Recommendation | 170 |
| 7.4.4 | Experiments | 171 |
| 7.4.5 | Self-organizing Relevance Links | 177 |
| 7.4.6 | Recommendation Performance | 178 |
| 7.5 | Applications | 181 |
| 7.5.1 | The Tribler System | 181 |
| 7.5.2 | Wi-Fi Walkman | 183 |
| 7.6 | Conclusions | 185 |
| 8 | Discussions | 187 |
| 8.1 | Scenarios | 187 |
| 8.2 | Relevance | 188 |
| 8.2.1 | Two views | 188 |
| 8.3 | Data Sparsity | 189 |
| 8.4 | Future Research | 189 |
| 8.4.1 | Discovering More from Information Retrieval Models | 189 |
| 8.4.2 | Beyond Collaborative Filtering | 190 |
| | Bibliography | 191 |

| | |
|-------------------------|------------|
| Acknowledgements | 203 |
| Curriculum Vitae | 205 |

Chapter 1

Introduction

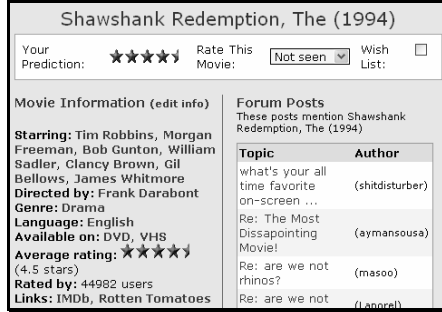
Technological breakthroughs in the last half of the 19th century, such as the telephone and telegraphy, have revolutionized communications and the spread of information. The subsequent digital revolution in the 20th century on information storage and transmission further increased the amount of information that we deal with in our daily lives. Although we enjoy the entertainment and convenience brought to us by such a variety of sources, the volume of information is increasing far more quickly than our ability to digest it. For instance, the Internet has become the most significant media source and is growing at an exponential speed. But users' ability to obtain useful information from the Internet grows at a slow rate. Tools that support the effective retrieval of relevant information are still primitive. Most information retrieval systems rely heavily on textual queries by users to identify their information needs. Queries constructed using keywords alone, however, are not powerful enough to express both semantically and contextually the needs of a particular user. To see this, consider the following common search scenario: "Find movies showing this weekend in nearby cinemas that I am most likely to enjoy." Such a user information need requires the retrieval system to at least be able to capture user interest ("most likely to enjoy"). Unfortunately, most existing retrieval models and search technologies are incapable of achieving such a realistic retrieval goal, because they focus only on building *correspondence* between textual queries and documents and lack the mechanisms to model individual users who issue queries. Hence, it is essential to accurately model various user information needs beyond queries. With recent advances in Human-Computer Interfacing and sensor technologies that make use of cameras, motion detectors, voice captures, GPSs etc., we have witnessed a research transition from information (document)-centric computing to user-centric computing - for example user profiling, which attempts to broadly understand users' various interests,

intentions, etc. on the basis of recorded human-computer interactions.

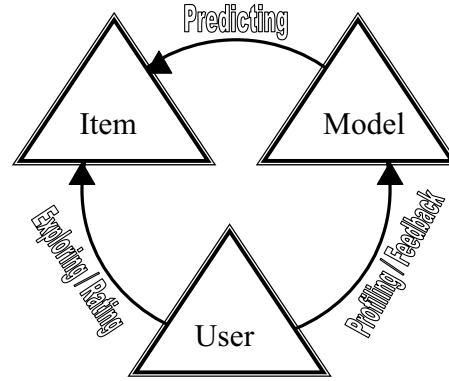
In addition, large amounts of information exist in a dynamic form. To process streams of incoming data, we need an information system that can play a more active role during information-seeking process. Therefore, information-filtering systems arise [5]. In contrast to most retrieval systems that passively wait for user queries to respond accordingly, information-filtering systems aim to actively filter out, refine and systematically represent the relevant information, intuitively ignoring superfluous computations on redundant data.

The potential combination of these two demands has created an increasing interest in building a filtering system that can steer users towards their personal interests and actively filter relevant information items on the users' behalf. As one of the dominant forms, recommender systems have appeared in the domains of Information Retrieval (IR) and Human-Computer Interaction (HCI). These systems attempt to filter information items such as books, CDs, DVDs, movies, TV programmes, and electronics, based on a history of the user's likes and dislikes. Examples include the Amazon's book-recommendation engine (amazon.com) and the Netflix DVD-recommendation engine (netflix.com). It is believed that recommender systems will eventually support companies to realize a shift from offering mass products and services to offering customized goods and services that efficiently satisfy the desires and needs of individual users. An extensive survey of recommendation algorithms can be found at [1]. For a recent overview of the personalization process in Web applications, we refer to [73].

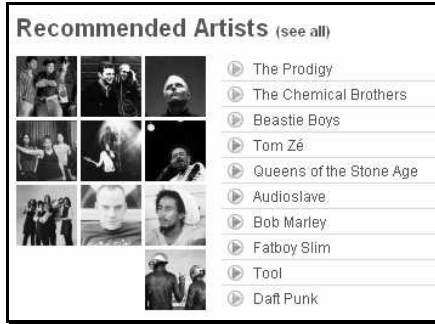
There are two prevalent approaches to recommending information items to a given user: content-based filtering [5] and collaborative filtering [30]. Content-based approaches usually make use of content descriptions, such as titles, genres, and synopses, to make the recommendation. However, content descriptions are not always available. In situations where information items are non-textual, content-based approaches in general have to dig deep, directly extracting and analyzing low-level features (measurable properties) from the content. For instance, colors, textures, and edges are the common features of image and video items [34, 101], while frequency-related features are usually adopted for audio and music items [54, 110]. Nonetheless, trying to bridge the so-called semantic gap between low-level features and high-level user needs causes many researchers to look beyond traditional low-level features. Collaborative filtering approaches are such methods that address this problem from a user perspective. In these approaches, the recommendations are calculated on the basis of a collection of user preferences, which are either obtained by explicitly asking users to rate items or implicitly learned from users' historic interactions with them. The work described in this thesis is one of the efforts undertaken in this direction.



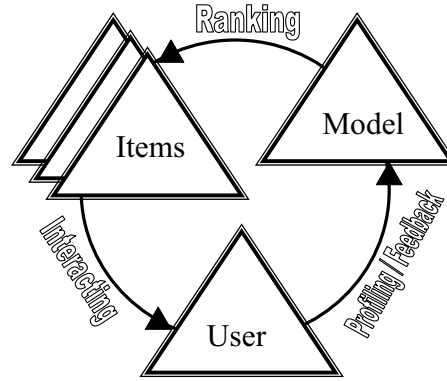
(a) A snapshot of the MovieLens System



(b) A diagram of rating prediction.



(c) A snapshot of the Last.FM system



(d) A diagram of item ranking.

Figure 1.1: The Two Forms of Recommendation.

1.1 Scope

This Ph.D thesis focuses on the theoretical understanding of the underlying collaborative filtering mechanisms. To gain an insight into the problems, we introduce the notion of relevance into collaborative filtering. We approach it from several aspects, mostly employing statistical models. Our intention is that, besides the theoretical contributions, the resulting recommendation models and techniques should cope with several major issues in recommender systems. Meanwhile, they should be applicable to many practical applications where recommendation and personalization are needed, such as in the areas of e-commerce, the Internet (e.g. web search, user-generated content, social media), information retrieval, and P2P content discovery.

To detail the scope of the thesis, this section starts with the scenarios of collaborative filtering, identifying opportunities and challenges in collaborative filtering research.

1.1.1 Collaborative Filtering Scenarios

Although collaborative filtering exists in various forms in practice, its purposes can be generally regarded as “item ranking” and “rating prediction”. These are illustrated in Fig 1.1. The rating prediction (see Fig 1.1 (a) and (b)) aims to predict an unknown rating of an item for a target user, with the requirement that the user has to explicitly rate a certain number of items. This type of recommendation has been widely conceived and well studied in the research literature since the pioneering work on the MovieLens systems (<http://movielens.umn.edu>) - from the early work on filtering netnews [81] and the movie recommender systems [37] to the latest Netflix competition (<http://www.netflixprize.com/>), most approaches accept by default that rating prediction is the basic core task for recommender systems. However, we shall see in this thesis that, in many practical systems such as Amazon (<http://amazon.com>) and Last.fm (<http://last.fm>) it is sometimes more favorable to formulate collaborative filtering as an item-ranking problem, because we often face a situation where our ultimate task is to generate the top-N list of the end user’s most favorite items (see Fig 1.1 (c) and (d)).

This thesis takes this difference into account when seeking statistical models for collaborative filtering. To cope with differing scenarios and data types, the methods proposed in each chapter, together with their evaluations, are specifically designed for one of the particular scenarios. For instance in Chapters 2, 3, and 4, we will first look at the item ranking problem of the Last.Fm scenario and then in Chapters 5 and 6 we will deal with the rating prediction problem of the MovieLens scenario.

1.1.2 Putting Relevance into Collaborative Filtering

The notion of relevance has a long history [72]; it is a key concept in information retrieval and facilitates the understanding of retrieval mechanisms and the development of retrieval models [91, 112]. To help understanding recommendation mechanisms and the development of collaborative filtering models, we shall introduce the notion of relevance into collaborative filtering research.

In information (text) retrieval, relevance is generally considered as, among other similar definitions, the correspondence between a user information need and an information item (e.g. a document) returned by the system in response to that need. Since the true user information need is quite often a hidden variable and in most cases impossible to extract, the relevance in many retrieval models is constructed at a surface level, i.e. by considering the evidence of user information needs such as users’ queries and relevance feedbacks. Owing to the absence of evidence about the difference between users, before relevance

feedback, the retrieval system may provide the same response to any two users who issue the same query.

In collaborative filtering and recommender systems, the underlying information need may have slightly different meanings. It represents a user interest or preference for a set of information items. The user interest may slowly vary over time as the user's goals, knowledge and conditions change, but nonetheless it remains relatively stable and can be inferred through user profiling. In this regard, we can think of the relevance of a recommender system as a correspondence between a *long-term* user interest and information items returned by the recommender system in response to that interest. The task of a recommender system is to find information items that are “relevant” to a user interest.

This school of thinking lays a foundation for our study of recommendation algorithms. We shall see it provides a theoretical framework to study the problem of collaborative filtering.

By doing this, we relate the problem of information retrieval and that of collaborative filtering at a conceptual level. Yet, at the modelling level they are quite apart from each other, as their input data and purposes are completely different. Consequently, applying the information retrieval (relevance) models to collaborative filtering is not trivial. The difficulty lies in the fact that in text retrieval both queries and documents are represented by texts, which provide an important information channel to link queries (user needs) and documents. Due to the lack of relevance observations, the retrieval models in text retrieval shift their focus from directly estimating the correspondence (relevance) between user needs (queries) and documents to estimating word statistics in the documents and/or queries and then building up the link through these statistics. Conversely, in collaborative filtering, in most cases, we do not have such extra information to relate users and information items. Instead, recorded in the system are only user preferences, which are thought of as indirect observations of the relevance between a user interest and an information item. Thus, the central question in modelling the relevance in collaborative filtering is how to relate users and items through this usually very sparse user-item matrix. We shall see that in this thesis we achieve this by establishing user representations, item representations or both on the basis of the user preferences.

The relevance between information items and user interests can be estimated through a variety of information channels. A general view of relevance would be to treat it as a hidden random variable, and subsequently, our belief about it is sequentially updated each time a new observation is made. In Chapters 2, 3, and 4, we will focus our study on evidence from users' interactions (e.g. users' playlists), while in Chapters 5 and 6, we will concentrate on explicit user ratings. In addition, to obtain task-focused recommendations, we shall see in Chapter 4 that keyword queries from users are effective in inferring and learning

aspects of user interests, therefore building up a more close relationship with text retrieval.

1.1.3 Issues in Collaborative Filtering

Once we introduce the notation of relevance and establish the correspondence between items and users to formulate collaborative filtering, we are equipped to formally address some practical issues of collaborative filtering. In this thesis we will concentrate on the following aspects:

Collaborative filtering suffers seriously from the data sparsity problem. That is, the number of information items is large, and most recommendation systems do not have enough observations, either explicitly or implicitly, about user preferences for these items. The resulting user preference data is therefore sparse, and there is no guarantee to find a set of users who have similar tastes. This typically happens when the ratio between the number of items and the number of users is high or when the recommender system is in the initial stage of use. In this regard, solving the data sparsity problem becomes an important objective in seeking appropriate models for collaborative filtering in this thesis.

It is highly desirable in practice to infer user preferences by implicitly observing user interactions with the recommender systems. Unfortunately, academic research into this implicit user-profiling for collaborative filtering has so far been limited, while a large body of research focuses on rating data. Thus, it is worthwhile investigating a formal model for those recommender systems that require implicit user profiles.

Recommendation without specifying user information needs (queries) is less task-focused. We need to have a flexible recommendation model that can incorporate more data about user information needs whenever it is available. As a case study, we will study the personalized retrieval problem in the context of collaborative tagging systems [32]. We will show that user input in the form of tags (few keywords) could provide an effective channel to infer and learn user information needs, resulting in more specific and task-focused recommendations.

Peer-to-peer (P2P) networks are becoming more and more popular for sharing information items. As the amount of data overwhelms the local storage, it is necessary to share the storage and filter relevant information in a personalized way. Our relevance model of collaborative filtering requires a centralized database to hold user preferences. Within a peer-to-peer network, however, such a centralized database is not readily available. Thus, we need to propose a fully distributed relevance model that can facilitate the distribution of computation loads and data into the network.

1.2 Thesis Outline

This thesis consists of a list of papers that have been published or submitted to international journals or conferences. The chapters are self-contained texts, which can be read independently. They are organized in such a way so as to reflect our thread in the quest to obtain collaborative filtering models. We begin with a “partial” view of relevance models, depicted from either a user perspective or an item perspective. We then move forward to a unified view, introducing unified (relevance) models that combine the two partial views. Finally, we investigate their applicability in distributed environments.

Chapter 2 investigates recommendations based on user preferences extracted from user interaction data. It introduces the basic relevance models within the context of collaborative filtering. To formally model the task of item ranking, we establish the log-ratio of relevance on the basis of the Probability Ranking Principle (PRP) of information retrieval [112]. This chapter constrains its discussion to the relevance case only and leaves a full discussion of the non-relevance case to Chapter 3. Two ranking models are introduced independently, namely the user-based relevance model and the item-based relevance model. The analysis and discussion of the two models supply a basis and motivation for the unified models that will be developed in the later chapters (Chapter 5 and Chapter 6).

Chapter 3 proposes a more advanced item-ranking framework, inspired by the BM25 formula in text retrieval [87]. This framework not only allows us to make use of frequency counts for modelling implicit user preferences but has room to model non-relevance in a formal way. However, data sparsity makes probability estimations less reliable. Thus, we extend the BM25 ranking formula by viewing the probabilities of (non-)relevance in the models as parameters and apply Bayesian inference to enforce different prior knowledge into the probability estimations. This will prove to be crucial for the accuracy of recommendation.

One of the drawbacks of existing collaborative filtering methods including those based on the techniques introduced in Chapters 2 and 3, is that the item ranking is made independently from the user’s task. To enforce the dependency of the ranking on the user’s task at hand, Chapter 4 therefore proposes a task-focused collaborative filtering framework incorporating tags as a task (or aspect) indicator. In the framework, we extend the approaches proposed in Chapter 2 and consider further types of generative processes in the tagging data, where smoothing methods are naturally integrated to cope with the problem of data sparsity.

Next we start to look at the rating prediction problem.

Chapter 5 considers a unified framework for the purpose of alleviating data sparsity. We view rating prediction in collaborative filtering as a voting mech-

anism from a pool of predictors. We identify three types of predictors and show that a carefully formulated, complete model for combining them leads to a robust prediction that is more tolerant to data sparsity.

Chapter 6 goes further regarding this topic and presents a unified relevance framework for rating prediction. We first establish a link between information retrieval and collaborative filtering, arguing that current memory-based collaborative filtering approaches represent only a partial view of the prediction problem and share the same drawback as the two common views on information retrieval. We then set up a unified probabilistic relevance framework to exploit more of the data available in the user-item matrix.

Chapter 7 addresses collaborative filtering from an application perspective, more specifically regarding its applicability for peer-to-peer networks. To eliminate its reliance upon centralized databases, we present two distributed approaches. First, from a user-oriented view of the network, we propose the *Buddycast* algorithm, which effectively exchanges user preferences using exploitation and exploration principles. We then present an item-oriented view and introduce a technique to distribute the computation of item-to-item similarity throughout the network, without sacrificing efficiency. At the conclusion of this chapter, we describe two peer-to-peer recommendation systems: the *Tribler* system and the *Wi-Fi Walkman* system.

Chapter 8 concludes the thesis with a discussion of future work.

1.3 Main Contributions

This thesis is devoted to various relevance models for collaborative filtering, reflecting different types of data inputs and different recommendation scenarios. Modelling relevance in collaborative filtering provides a general yet concrete solution for the development of collaborative filtering and recommender systems both effectively and efficiently. The main contributions are summarized as follows:

Collaborative filtering has often been formulated as a self-contained problem, parallel to the classic information retrieval problem (i.e. ad hoc text retrieval). This thesis relates collaborative filtering to text retrieval (Chapters 2, 3, 4, and 6). The way of thinking expands our methodologies towards collaborative filtering, providing a flexible framework in which to try out more of the techniques that have been used in text retrieval for the related problem of collaborative filtering.

The problem of data sparsity is tackled in a formal way (Chapters 2, 3, 4, and 6). In our relevance frameworks, the probabilities of (non-)relevance have to

be estimated from the severely under-sampled input data. This largely reduces the reliability of the estimations. The Bayesian inference framework that we adopt offers an elegant way of converting the problem of estimation (and thus the data sparsity problem) to that of choosing a proper prior, which in turn can be estimated from the whole data collection.

Chapters 5 and 6 provide a unified view of and solution to the prediction problem. In information retrieval, there have been two separate views on how to assign a probability of relevance for a document to a user need have existed, namely the *document-oriented* and the *query-oriented* views; the classic probabilistic retrieval model of information retrieval [86] takes the query-oriented view while the language modelling approach to information retrieval [39, 55, 58, 78] builds upon the document-oriented view. Neither represents the problem of information retrieval completely [85]. Likewise, we discover the same drawback in popular user-based and item-based collaborative filtering approaches (Chapters 5 and 6). We thus propose a *unified relevance model* of collaborative filtering (Chapter 6) by applying kernel density estimation and as a result, providing a practical solution for the unification advocated in [85] in the context of collaborative filtering.

A practical contribution lies in a method for implementing distributed collaborative filtering. Collaborative filtering requires a centralized user profile database, but within a peer-to-peer network such a centralized database is not readily available. The proposed novel scheme, called *BuddyCast*, builds such a social network for a user by exchanging user profiles using exploitation and exploration principles (Chapter 7). As one of the core algorithms for a peer-to-peer file sharing system (tribler.org), it has been released to the public at large.

Part I

Relevance Models

Chapter 2

Language Modelling Approaches

In this paper, we follow a formal approach of text retrieval to reformulate the collaborative filtering problem. Based on the classic probability ranking principle, we propose a probabilistic user-item relevance model. Under this formal model, we show that user-based and item-based approaches are only two different factorizations with different independence assumptions. Moreover, we show that smoothing is an important aspect in estimating the parameters of the models, because of data sparsity. By adding linear interpolation smoothing, the proposed model provides a probabilistic justification for using TF \times IDF-like item ranking in collaborative filtering. Besides offering the insight into the problem of collaborative filtering, we also show experiments in which the proposed method provides a better recommendation performance on a music playlist data set.

This work has been published as “A user-item relevance model for log-based collaborative filtering” by J. Wang, A. P. de Vries, and M. J. Reinders, in Proc. of European Conference on Information Retrieval (ECIR06), London, UK, 2006. See also [114].

2.1 Introduction

Generally, a collaborative filtering algorithm uses a collection of user profiles to identify interesting “information” for these users. A particular user acquires a recommendation based on the user profiles of other similar users. User profiles are commonly obtained by explicitly asking users to rate the items. Collaborative filtering has often been formulated as a self-contained problem, distinct from the classic information retrieval problem (i.e. ad hoc text retrieval). Research started with heuristic implementations of “Word of Mouth” (e.g. user-based approaches [9]) and moved to item-based approaches [92], and more recently, various model-based approaches have been introduced [42, 66].

Previous research [15] has shown that users are very unlikely to provide an explicit rating. Asking the user to rate items is annoying and should be avoided whenever possible. Alternatively, user profiles can also be obtained by implicitly observing user interactions with the system. For instance, a music playlist indicates the music tastes of a user, and web query logs could indicate the interest of a user in certain web sites. The implicit acquisition of user preferences makes the “log-based” collaborative filtering more favorable in practice (see Section 2.2).

This paper therefore focuses on log-based collaborative filtering. We identify a close relationship between log-based collaborative filtering and text information retrieval. We build a user-item relevance model to reformulate collaborative filtering under the classic probability ranking principle. Given our user-item relevance models, we also introduce the linear interpolation *smoothing* into collaborative filtering. We show that the smoothing is important in estimating the model parameters correctly, because of data sparsity. Similar to the situation in text retrieval, the user-item relevance model provides a probabilistic justification for using $\text{TF} \times \text{IDF}$ -like item weighting in collaborative filtering.

2.2 Background

2.2.1 Rating-based Collaborative Filtering

Preference information about items can be based either on user ratings (explicit interest functions) or log-archives (implicit interest functions). Their differences lead, in our view, to two distinct ways of approaching collaborative filtering: *rating-based* and *log-based*. Rating-based collaborative filtering is based on user profiles that contain rated items. The majority of the literature addresses rating-based collaborative filtering, which has been studied in depth [66]. Rating-based approaches are often classified as memory-based or model-

based. In the memory-based approach, all rating examples are stored as is into memory (in contrast to learning an abstraction). In the prediction phase, similar users or items are sorted based on the memorized ratings. Based on the ratings of these similar users or items, a recommendation for the target user can be generated. Examples of memory-based collaborative filtering include item correlation-based methods [92], user clustering [123] and locally weighted regression [9]. The advantage of the memory-based methods over their model-based alternatives is that they have fewer parameters to be tuned, while the disadvantage is that the approach cannot deal with data sparsity in a principled manner.

In the model-based approach, training examples are used to generate a model that is able to predict the ratings for items that a target user has not previously rated. Examples include decision trees [9], latent class models [42], and factor models [10]. The “compact” models in these methods could solve the data sparsity problem to a certain extent. However, the requirement of tuning an often significant number of parameters or hidden variables has prevented these methods from practical usage.

Recently, to overcome the drawbacks of these approaches to collaborative filtering, researchers have started to combine memory-based and model-based approaches [77, 118].

2.2.2 Log-based Collaborative Filtering

Implicit interest functions can be represented by binary-valued preferences. That is a one indicates a “file is downloaded”, or a “web-site is visited”. Few log-based collaborative filtering approaches that deal with such data have been developed thus far. Two examples are the item-based top-N collaborative filtering approach [23, 52] and Amazon’s item-based collaborative filtering [61].

The following characteristics make log-based collaborative filtering more similar to the problem of text retrieval than the rating-based approaches:

- The simplest way to employ log-based user profiles, e.g. playlists, is to assume they are binary-valued. Usually, one means “relevance” or “likeness”, and zero indicates “non-relevance” or “non-likeness”. Moreover, in most situations, non-relevance and non-likeness are rarely observed. This is similar to the concept of relevance in text retrieval.
- A common goal for rating-based collaborative filtering is to predict the rating of users, while for log-based algorithms it is desirable to rank the items to the user in order of decreasing relevance. As a result, their evaluation differs. In rating-based collaborative filtering, the mean square

error (MSE) of the predicted rating is usually adopted, while in log-based collaborative filtering, recall and precision metrics are employed.

This paper therefore proposes to apply the probabilistic framework developed for text retrieval to log-based collaborative filtering. We consider the following formal setting. The information that has to be filtered, e.g. images, movies, or audio files, is represented as a set of *items*. We introduce discrete random variables $U \in \{u_1, \dots, u_K\}$ and $I \in \{i_1, \dots, i_M\}$ to represent a user and an item in the collection, respectively. K is the number of users while M is the number of items in the collection. Let L_{u_k} denote a user profile list for user u_k . L_{u_k} is a set of items in which user u_k has previously shown interest. $L_{u_k}(i_m) = 1$ (or $i_m \in L_{u_k}$) indicates that item i_m is in the list while $L_{u_k}(i_m) = 0$ (or $i_m \notin L_{u_k}$) indicates otherwise. The number of items in the list is denoted as $|L_{u_k}|$.

The purpose of log-based collaborative filtering is to rank the relevance of a target item to a user. This could be represented by the retrieval status value (RSV) of a target item for a user, denoted as: $RSV_{u_k}(i_m)$. Heuristic implementations of “Word of Mouth” introduced in [23, 37] provide the following basic item-based and user-based approaches for calculating the RSV when we consider the binary case:

$$\begin{aligned} \text{User-based : } RSV_{u_k}(i_m) &= \sum_{\text{Top-N similar } u_b} s_U(u_k, u_b) L_{u_b}(i_m) \\ \text{Item-based : } RSV_{u_k}(i_m) &= \sum_{\forall i_b: i_b \in L_{u_k}} s_I(i_b, i_m) \end{aligned} \quad (2.1)$$

where s_I and s_U are the two similarity measures between two items and two users, respectively. The two commonly used similarity measures are the Pearson correlation and the cosine similarity [9]. Frequency counting has been used as an alternative basis for similarity measures in [23, 52]. To suppress the influence of items that are being purchased frequently, these studies have introduced a TF×IDF-like weighting (similarity) function:

$$s_I(i_b, i_m) = \frac{c(i_b, i_m)/c(i_m)}{c(i_b)^\alpha} \quad (2.2)$$

where $c(i_b, i_m) = \sum_{k=1}^K |L_{u_k}(i_m) \cap L_{u_k}(i_b)|$ is the number of user profiles in which both items i_b and i_m exist (i.e., items that *co-occur*); and $c(i)$ is the number of user profiles containing item i . α is a tuning parameter.

2.3 A User-Item Relevance Model

In log-based collaborative filtering, users want to know which items fit their interests best. This section adopts the probabilistic relevance model proposed in the text retrieval domain [56, 78] to measure the relevance between user interests and items. We intend to answer the following basic question:

- What is the probability that *this* item is relevant to *this* user, given his or her profile?

To answer this question, we first define the sample space of relevance: Φ_R . It has two values: “relevant” r and “non-relevant” \bar{r} . Let R be a random variable over the sample space Φ_R . Likewise, let U be a discrete random variable over the sample space of *user ids*: $\Phi_U = \{u_1, \dots, u_K\}$ and let I be a random variable over the sample space of *item ids*: $\Phi_I = \{i_1, \dots, i_M\}$, where K is the number of users and M the number of items in the collection. In other words, U refers to the user identifiers and I refers to the item identifiers.

We then denote P as a probability function on the joint sample space $\Phi_U \times \Phi_I \times \Phi_R$. In a probability framework, we can answer the above basic question by estimating the probability of relevance $P(R = r|U, I)$. The relevance rank of items in the collection Φ_I for a given user $U = u_k$ can be formulated as the odds of the relevance:

$$RSV_{u_k}(i_m) = \ln \frac{P(r|u_k, i_m)}{P(\bar{r}|u_k, i_m)} \quad (2.3)$$

For simplicity, $R = r$, $R = \bar{r}$, $U = u_k$, and $I = i_m$ are denoted as r , \bar{r} , u_k , and i_m , respectively.

Hence, the evidence for the relevance of an item for a user is based on both the positive evidence (indicating the relevance) as well as the negative evidence (indicating the non-relevance). Once we know, for a given user, the RSV of each item I in the collection (excluding the items in which the user has already expressed interest), we sort these items in decreasing order. The highest-ranked items are then recommended to the user.

In order to estimate the conditional probabilities in Eq. 2.3, i.e. the relevance and non-relevance between the user and the item, we need to factorize the equation along the item or the user dimension. We propose to consider both *item-based generation* (i.e. using items as features to represent the user) and *user-based generation* (i.e. treating users as features to represent an item).

2.3.1 Item-based Generation

By factorizing $P(\bullet|u_k, i_m)$ with $\frac{P(u_k|i_m, \bullet)P(\bullet|i_m)}{P(u_k|i_m)}$, the following log-odds ratio can be obtained from Eq. (2.3):

$$RSV_{u_k}(i_m) = \ln \frac{P(r|i_m, u_k)}{P(\bar{r}|i_m, u_k)} = \ln \frac{P(u_k|i_m, r)}{P(u_k|i_m, \bar{r})} + \ln \frac{P(i_m|r)P(r)}{P(i_m|\bar{r})P(\bar{r})} \quad (2.4)$$

Without explicit evidence for non-relevance, and following the language modelling approach to information retrieval [56], we now assume: 1) independence between u_k and i_k in the non-relevance case (\bar{r}), i.e., $P(u_k|i_m, \bar{r}) = P(u_k|\bar{r})$; and, 2) equal priors for both u_k and i_m , given that the item is non-relevant. Then the two non-relevance terms can then be removed and the RSV becomes:

$$RSV_{u_k}(i_m) = \ln P(u_k|i_m, r) + \ln P(i_m|r) \quad (2.5)$$

Note that the two negative terms in Eq. (2.4) can always be added to the model, when the negative evidences are captured.

To estimate the conditional probability $P(u_k|i_m, r)$ in Eq. 2.5, consider the following. Instead of placing users in the sample space of user ids, we can also use the set of items that the user likes (L_{u_k}) to represent the user (u_k). This step is similar to using a “bag-of-words” representation of queries or documents in the text retrieval domain [90]. This implies: $P(u_k|i_m, r) = P(L_{u_k}|i_m, r)$. We call these representing items *query items*. Note that, unlike the target item i_m , the query items do not need to be ranked since the user has already expressed interest in them.

Furthermore, we assume that the items in the user profile list L_{u_k} (query items) are conditionally independent from each other. Although this naive Bayes assumption does not hold in many real situations, it has been empirically shown to be a competitive approach (e.g. in text classification [27, 112]). Under this assumption, Eq. 2.5 becomes:

$$\begin{aligned} RSV_{u_k}(i_m) &= \ln P(L_{u_k}|i_m, r) + \ln P(i_m|r) \\ &= \sum_{\forall i_b: i_b \in L_{u_k}} \ln P(i_b|i_m, r) + \ln P(i_m|r) \end{aligned} \quad (2.6)$$

The conditional probability $P(i_b|i_m, r)$ corresponds to the relevance of an item i_b , given that another item i_m is relevant. This probability can be estimated by counting the number of user profiles that contain both items i_b and i_m , divided by the total number of user profiles in which i_m exists (see also [52]):

$$P_{ml}(i_b|i_m, r) = \frac{P(i_b, i_m|r)}{P(i_m|r)} := \frac{c(i_b, i_m)}{c(i_m)} \quad (2.7)$$

Probability Estimation and Smoothing

Using the frequency count in Eq. 2.7 to estimate the above probability corresponds to using its maximum likelihood estimator. However, many item-to-item co-occurrence counts will be zero, because of the sparseness of the user-item matrix. Therefore, we apply a smoothing technique to adjust the maximum likelihood estimation [112].

In information retrieval [126], most smoothing methods apply two distinct distributions: one for the words that occur in the document, and one for the words that do not. Here, we also adopt this formulation. To estimate $P(i_b|i_m, r)$, we use $P_s(i_b|i_m, r)$ when $c(i_b, i_m) > 0$, while when $c(i_b, i_m) = 0$ (i.e. i_b and i_m do not co-occur in any of the user profiles), we assume the probability is proportional to the general frequency of i_b for the whole user profile set. That is, $P(i_b|i_m, r) = \alpha_{i_m} P(i_b|r)$, where α_{i_m} depends on item i_m . Then, the conditional probability between a user and an item can be formulated as follows:

$$\begin{aligned}
\ln P(u_k|i_m, r) &= \ln P(L_{u_k}|i_m, r) \\
&= \sum_{\forall i_b: i_b \in L_{u_k} \cap c(i_b, i_m) > 0} \ln P_s(i_b|i_m, r) + \sum_{\forall i_b: i_b \in L_{u_k} \cap c(i_b, i_m) = 0} \ln \alpha_{i_m} P(i_b|r) \\
&= \sum_{\forall i_b: i_b \in L_{u_k} \cap c(i_b, i_m) > 0} \ln \frac{P_s(i_b|i_m, r)}{\alpha_{i_m} P(i_b|r)} + \sum_{\forall i_b: i_b \in L_{u_k}} \ln \alpha_{i_m} P(i_b|r) \quad (2.8) \\
&= \sum_{\forall i_b: i_b \in L_{u_k} \cap c(i_b, i_m) > 0} \ln \frac{P_s(i_b|i_m, r)}{\alpha_{i_m} P(i_b|r)} + |L_{u_k}| \ln \alpha_{i_m} + \sum_{\forall i_b: i_b \in L_{u_k}} \ln P(i_b|r)
\end{aligned}$$

Since the last term is independent of the target item i_m , it can be dropped when we calculate the RSV of item i_m . Combining Eq. (2.6) and Eq. (2.8), we obtain the following:

$$\begin{aligned}
RSV_{u_k}(i_m) &= \sum_{\forall i_b: i_b \in L_{u_k} \cap c(i_b, i_m) > 0} \ln \frac{P_s(i_b|i_m, r)}{\alpha_{i_m} P(i_b|r)} + |L_{u_k}| \ln \alpha_{i_m} + \ln P(i_m|r) \quad (2.9)
\end{aligned}$$

Eq. 2.9 provides a generative ranking formula. Next, we consider a special case: a linear interpolation smoothing.

Linear Interpolation Smoothing

A linear interpolation smoothing can be defined as a linear interpolation between the maximum likelihood estimation and the background model. To use

it, if we define:

$$\begin{aligned} P_s(i_b|i_m, r) &= (1 - \lambda)P_{ml}(i_b|i_m, r) + \lambda P(i_b|r) \\ \alpha_{i_m} &= \lambda \end{aligned} \quad (2.10)$$

where $P_{ml}(i_b|i_m, r)$ is the maximum likelihood estimation as given in Eq. 2.7. Item prior probability $P(i_b|r)$ is used as a background model. Furthermore, the parameter $\lambda \in [0, 1]$ is a parameter that balances the maximum likelihood estimation and background model (a larger λ means more smoothing). Usually, the best value for λ is found from a training data. The linear interpolation smoothing leads to the following RSV:

$$\begin{aligned} RSV_{u_k}(i_m) &= \sum_{\forall i_b: i_b \in L_{u_k} \cap c(i_b, i_m) > 0} \ln(1 + \frac{(1 - \lambda)P_{ml}(i_b|i_m, r)}{\lambda P(i_b|r)}) + \ln P(i_m|r) \end{aligned} \quad (2.11)$$

2.3.2 User-based Generation

By factorizing $P(\bullet|u_k, i_m)$ with $P(i_m|u_k, \bullet)P(\bullet|u_k)/P(i_m|u_k)$, the following log-odds ratio can be obtained from Eq. 2.3:

$$\begin{aligned} RSV_{u_k}(i_m) &= \ln \frac{P(r|i_m, u_k)}{P(\bar{r}|i_m, u_k)} \\ &= \ln \frac{P(i_m|u_k, r)}{P(i_m|u_k, \bar{r})} + \ln \frac{P(u_k|r)P(r)}{P(u_k|\bar{r})P(\bar{r})} \\ &\propto \ln \frac{P(i_m|u_k, r)}{P(i_m|u_k, \bar{r})} \end{aligned} \quad (2.12)$$

When the non-relevance evidence is absent, and following the language model [56], we now assume equal priors for i_m in the non-relevant case. The non-relevance term can then be removed and the RSV becomes:

$$RSV_{u_k}(i_m) = \ln P(i_m|u_k, r) \quad (2.13)$$

Instead of using the item list to represent the user, we use each user's judgment as a feature to represent an item. For this, we introduce a list L_{i_m} for each item i_m , where $m = \{1, \dots, M\}$. This list enumerates the users who have expressed interest in the item i_m . $L_{i_m}(u_k) = 1$ (or $u_k \in L_{i_m}$) denotes that user u_k is in the list, while $L_{i_m}(u_k) = 0$ (or $u_k \notin L_{i_m}$) indicates otherwise. The number of users in the list corresponds to $|L_{i_m}|$.

Replacing i_m with L_{i_m} , after we assume that each user's judgment of a particular item is independent, we obtain:

$$RSV_{u_k}(i_m) = \ln P(i_m|u_k, r) = \sum_{\forall u_b: u_b \in L_{i_m}} \ln P(u_b|u_k, r) \quad (2.14)$$

Similar to the item-based generation, when we use linear interpolation smoothing to estimate $P(u_b|u_k, r)$, we obtain the final ranking formula:

$$\begin{aligned} RSV_{u_k}(i_m) &= \sum_{\forall u_b: u_b \in L_{i_m}} \ln P(u_b|u_k, r) \\ &\propto \sum_{\forall u_b: u_b \in L_{i_m} \cap c(u_b, u_k) > 0} \ln \left(1 + \frac{(1 - \lambda)P_{ml}(u_b|u_k, r)}{\lambda P(u_b|r)} \right) + |L_{i_m}| \ln \lambda \end{aligned} \quad (2.15)$$

where $\lambda \in [0, 1]$ is the smoothing parameter.

2.3.3 Discussions

Inverse Item Frequency

The usage of TF×IDF-like ranking shown in Eq. 2.2 was studied in [23] and has been shown to display the best performance. However, [23] does not provide justification about the usage of the *inverse item frequency* ($1/P(i_b|r)$) through probability theory. By considering log-based collaborative filtering probabilistically and proposing the linear interpolation smoothing, our user-item relevance model in Eq. 2.11 provides a probabilistic justification. Our ranking formula can directly be interpreted as TF×IDF-like ranking, since

$$P_{ml}(i_b|i_m, r) \propto c(i_b, i_m)/c(i_m) \text{ and } P(i_b|r) \propto c(i_b) \quad (2.16)$$

In addition, Eq. 2.11 offer a very intuitive understanding of the statistical ranking mechanisms that play a role in log-based collaborative filtering:

- The relevance rank of a target item i_m is the sum of both its popularity (prior probability $P(i_m|r)$) and its co-occurrence (the first term in Eq. 2.11) with the items (i_b) in the profile list of the target user. The co-occurrence is higher if more users express interest in the target item (i_m) as well as item i_b . However, the co-occurrence should be suppressed more when the popularity of the item in the profile of the target user ($P(i_b|r)$) is higher.

- When λ approaches 0, smoothing from the background model is minimal. It emphasizes the co-occurrence count, and the ranking becomes equivalent to *coordination level matching* [40], which is simply counting the number of times for which $c(i_b, i_m) > 0$. When the λ is equal to 0, the model reduces to the traditional item-based approach [61]. When the λ approaches 1, the model is more smooth, emphasizing the background model.

Two Representations

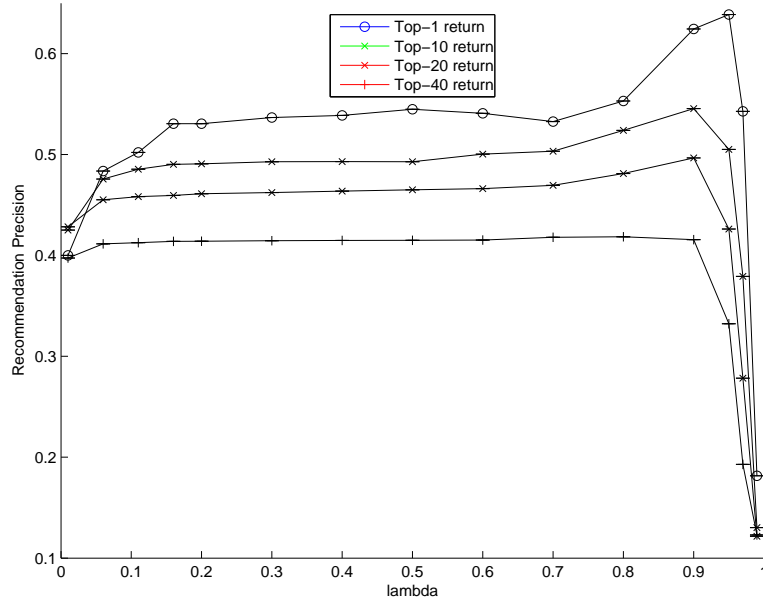
Traditionally, collaborative filtering makes a distinction between user-based and item-based approaches. Our probabilistic user-item relevance model, derived with an information retrieval view to collaborative filtering, demonstrates that the user-based (Eq. 2.15) and item-based (Eq. 2.11) models are, in fact, equivalent from the probabilistic point of view, since they have actually been derived from the same generative relevance model (Eq. 2.3). The only difference in their derivation corresponds to the choice of independence assumptions, leading to the two distinct factorizations.

Consequently, this formula offers a much better understanding of the underlying statistical assumptions that are made in these two approaches. In the user-based approach, a target item is assumed to be judged or rated independently, while in the item-based approach, a target user is assumed independently to judge or rate each query item. Besides the differences in the number of users (K) and the number of items (M), we believe that these underlying assumptions are the major factors influencing the performances of these two approaches in practice.

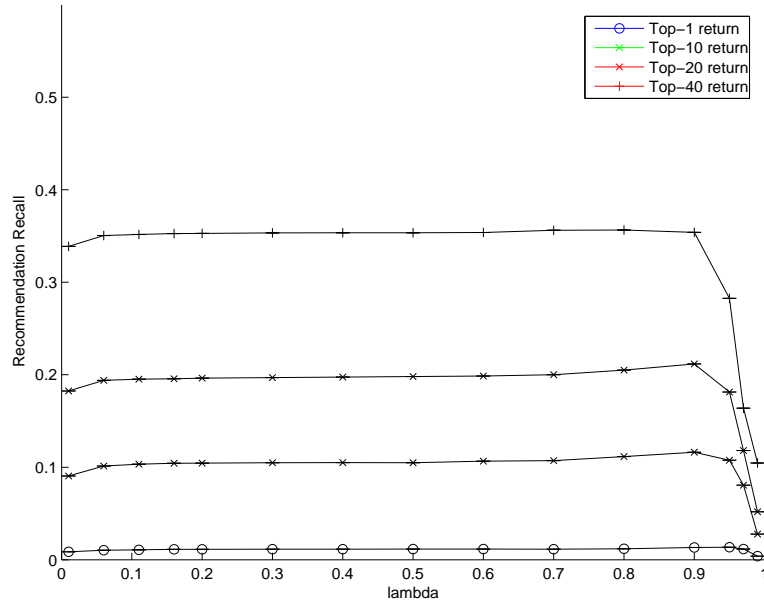
2.4 Experiments

The standard data set used in the evaluation of collaborative filtering algorithms (MovieLens) is rating-based, which is not suitable for testing our method using log-based user profiles. The user logs we employed were collected from the *Audioscrobber*¹ community. The Audioscrobber data set is collected from the playlists of the users in the community by using a plug-in in the users' media players (for instance, Winamp, iTunes, and XMMS). Plug-ins send the title (song name and artist name) of every song that users play to the Audioscrobber server, which updates the user's musical profile with the new song. That is, when a user plays a song in a certain time, this transaction is recorded as a tuple in the form of $\{\text{userID}, \text{itemID}, t\}$ in the database.

¹Audioscrobber can be found at <http://last.fm>.



(a) Precision



(b) Recall

Figure 2.1: Performance of the linear interpolation smoothing.

For computational reasons, we randomly sampled the data set to limit the number of users to 428 and the number of items to 516. The sparsity (percentage of 0 values in the user-item matrix) is 96.86% .

For cross-validation, we randomly divided this data set into a training set (80% of the users) and a test set (20% of the users). Results are obtained by averaging five separate runs (sampling of training/test set). The training set was used to estimate the model. The test set was used for evaluating the accuracy of the recommendations on the new users, whose user profiles were not in the training set. For each test user, 50% of the items of a test user were placed into the user profile list. The other 50% of the items were used to test the recommendations. By doing so, the number of items in the user profiles reflects the distribution in the overall data set.

The effectiveness of the log-based collaborative filtering experiments can be measured using the *precision* and *recall* of the recommendations. Precision measures the proportion of recommended items that are ground truth items. The recall measures the proportion of the ground truth items that are recommended. In the case of making recommendations, precision seems more important than recall. However, to analyze the behavior of our method, we report both metrics in our experimental results.

We first studied the behavior of the linear interpolation smoothing. For this, we plotted the average precision and recall rate for the different values of the smoothing parameter λ . This is shown in Fig. 2.1.

Fig. 2.1 (a) and (b) show that both precision and recall drop when λ reaches its extreme values 0 and 1. The precision is sensitive to λ , especially the early precision (when only a small number of items are recommended). Recall is less sensitive to the actual value of this parameter, having its optimum at a wide range of values. Effectiveness tends to be higher in both metrics when λ is large; when λ is approximately 0.9, the precision seems optimal. An optimal range of λ near 1 can be explained by the sparsity of user profiles, causing the prior probability $P(i_b|r)$ to be much smaller than the conditional probability $P_{ml}(i_b|i_m, r)$. The background model is therefore only emphasized for values of λ closer to 1. In combination with the experimental results that we obtained, this suggests that smoothing the co-occurrence probabilities with the background model (prior probability $P(i_b|r)$) improves recommendation performance.

Next, we compared our user-item relevance model to other log-based collaborative filtering approaches. Our goal here was to see, using our user-item relevance model, whether the smoothing and inverse item frequency should improve recommendation performance with respect to the other methods. For this, we focused on the item-based generation (denoted as UIR-Item). We set λ to the optimal value 0.9. We compared our results to those obtained with the top-N SUGGEST recommendation engine, a well-known log-based collaborative filtering implementation [52, 60]. This engine implements a variety of log-based recommendation algorithms. We compared our own results

| | Top-1 Item | Top-10 Item | Top-20 Item | Top-40 Item |
|--------------------|------------|-------------|-------------|-------------|
| UIR-Item | 0.62 | 0.52 | 0.44 | 0.35 |
| Item-TFIDF | 0.55 | 0.47 | 0.40 | 0.31 |
| Item-CosSim | 0.56 | 0.46 | 0.38 | 0.31 |
| Item-CorSim | 0.50 | 0.38 | 0.33 | 0.27 |
| User-CosSim | 0.55 | 0.42 | 0.34 | 0.27 |

(a) Precision

| | Top-1 Item | Top-10 Item | Top-20 Item | Top-40 Item |
|--------------------|------------|-------------|-------------|-------------|
| UIR-Item | 0.02 | 0.15 | 0.25 | 0.40 |
| Item-TFIDF | 0.02 | 0.15 | 0.26 | 0.41 |
| Item-CosSim | 0.02 | 0.13 | 0.22 | 0.35 |
| Item-CorSim | 0.01 | 0.11 | 0.19 | 0.31 |
| User-CosSim | 0.02 | 0.15 | 0.25 | 0.39 |

(b) Recall

Table 2.1: Comparison of Recommendation Performance.

with both the item-based $\text{TF} \times \text{IDF}$ -like version (denoted ITEM-TFIDF) and the user-based cosine similarity method (denoted User-CosSim), setting the parameters to the optimal ones according to the user manual. Additionally, for item-based approaches, we also used other similarity measures: the commonly used cosine similarity (denoted Item-CosSim) and Pearson correlation (denoted Item-CorSim). The results are shown in Table 2.1. For the precision, our user-item relevance model with the item-based generation (UIR-Item) outperforms other log-based collaborative filtering approaches for all four different numbers of returned items. Overall, $\text{TF} \times \text{IDF}$ -like ranking ranks second. The obtained experimental results demonstrate that smoothing contributes to a better recommendation precision in the two ways that were also found by [126]. On the one hand, smoothing compensates for missing data in the user-item matrix, and on the other hand, it plays the role of inverse item frequency to emphasize the weight of the items with the best discriminative power. With respect to recall, all four algorithms perform almost identically. This is consistent with our first experiment, which determined that recommendation precision is sensitive to the smoothing parameters while the recommendation recall is not.

2.5 Conclusions

This paper has identified a close relationship between log-based collaborative filtering and the methods developed for text information retrieval. We have built a user-item relevance model to reformulate the collaborative filtering problem under the classic probability ranking principle. Using this probabilistic framework

of user-item relevance models, we introduced linear interpolation *smoothing* in collaborative filtering. We showed that smoothing is an important aspect in estimating models, because of data sparsity. Similar to the situation in text retrieval, the user-item relevance model provides a probabilistic justification for using TF \times IDF-like item weighting in collaborative filtering.

Our further research aims to introduce relevance feedback into collaborative filtering. One of the powerful characteristics of linear interpolation smoothing is that we can vary the smoothing parameter: $\lambda \rightarrow \lambda(i_b)$ for the various items i_b in the user profile. It can then be treated as the importance of the query item. Initially, all the items in the user profile are treated equally. From relevance feedback, the importance value for various query items can be updated by using EM algorithm [40].

Commentary on Chapter 2

Event Space

In Chapter 2, we defined users and items in their identity spaces, and only when making probability estimations did we introduce their feature representations. The introduction of the “identity spaces”, however, is unnecessary and may be difficult to follow. Here, we provide a more general treatment.

First, information items and users are represented by their measurable properties, commonly called features. Mathematically, for each item, we have a K -dimensional random vector $\mathbf{I} = (I_1, \dots, I_k, \dots, I_K)$ over the sample space of the item properties (item features) $\Phi_{\mathbf{I}}$. Likewise, for each user, we have a M -dimensional random vector $\mathbf{U} = (U_1, \dots, U_m, \dots, U_M)$ over the sample space of user properties $\Phi_{\mathbf{U}}$. U_m and I_k are the elements of the feature vectors. The resulting definitions are of a general nature, because we can employ any feature representations that we can think of, as long as they are available. The possible features representing users include user demographics, user preferences, and relevance feedback, while those for items include content descriptions, low-level features, users’ ratings of them. In this thesis, we have limited our discussions to user profile data only, where users are represented by their preferences while items are characterized by users’ judgments of them.

Second, to indicate the correspondence between users and items, we define a sample space of relevance Φ_R . Let R be a random variable over the relevance space Φ_R . The random variable can be binary, multiple-valued, or even multivariate, depending on the practical situations. For instance, in Chapters 1 and 2 we assumed that it is binary when aiming at item relevance ranking, while in Chapter 6 we assume that it has multiple values when focusing on rating prediction. For the moment, let us consider the binary case $\Phi_R = \{r, \bar{r}\}$, in which R is either “relevant” r or “non-relevant” \bar{r} . In most cases, we do not have a direct observation of the relevancy between users and items. A simple solution, as described in Chapters 1 and 2, would be to consider an item relevant to a user if this user frequently plays or visits the item.

Third, a probability measure on the joint sample space $\Phi_U \times \Phi_I \times \Phi_R$ models the random behavior of the random vector $(\mathbf{U}, \mathbf{I}, R)$. An item recommendation model can thus be built by estimating the log odds of the relevance:

$$o_{\mathbf{u}}(\mathbf{i}) = \ln \frac{P(r|\mathbf{u}, \mathbf{i})}{P(\bar{r}|\mathbf{u}, \mathbf{i})}$$

where $\mathbf{u} = (u_1, \dots, u_m, \dots, u_M)$ and $\mathbf{i} = (i_1, \dots, i_k, \dots, i_K)$ are, respectively, the two instantiations of the random vectors \mathbf{U} and \mathbf{I} . For simplicity, the propositions $R = r$, $R = \bar{r}$, $\mathbf{U} = \mathbf{u}$ and $\mathbf{I} = \mathbf{i}$ are denoted as r , \bar{r} , \mathbf{u} , and \mathbf{i} , respectively.

We then obtain the following dual solutions if we apply Bayes' rule differently:

$$\begin{aligned} o_{\mathbf{u}}(\mathbf{i}) &= \ln \frac{P(\mathbf{u}|r, \mathbf{i})}{P(\mathbf{u}|\bar{r}, \mathbf{i})} + \ln \frac{P(r|\mathbf{i})}{P(\bar{r}|\mathbf{i})} && \text{(the item-based)} \\ o_{\mathbf{u}}(\mathbf{i}) &= \ln \frac{P(\mathbf{i}|r, \mathbf{u})}{P(\mathbf{i}|\bar{r}, \mathbf{u})} + \ln \frac{P(r|\mathbf{u})}{P(\bar{r}|\mathbf{u})} && \text{(the user-based)} \end{aligned}$$

The vectors \mathbf{u} and \mathbf{i} are quite often high-dimensional, making the conditional probabilities difficult to estimate. A common practice [25, 59] is to assume conditional independence among the element features. This leads to:

$$\begin{aligned} o_{\mathbf{u}}(\mathbf{i}) &= \ln \frac{P(\mathbf{u}|r, \mathbf{i})}{P(\mathbf{u}|\bar{r}, \mathbf{i})} + \ln \frac{P(r|\mathbf{i})}{P(\bar{r}|\mathbf{i})} = \sum_m \ln \frac{P(u_m|r, \mathbf{i})}{P(u_m|\bar{r}, \mathbf{i})} + \ln \frac{P(r|\mathbf{i})}{P(\bar{r}|\mathbf{i})} && \text{(the item-based)} \\ o_{\mathbf{u}}(\mathbf{i}) &= \ln \frac{P(\mathbf{i}|r, \mathbf{u})}{P(\mathbf{i}|\bar{r}, \mathbf{u})} + \ln \frac{P(r|\mathbf{u})}{P(\bar{r}|\mathbf{u})} = \sum_k \ln \frac{P(i_k|r, \mathbf{u})}{P(i_k|\bar{r}, \mathbf{u})} + \ln \frac{P(r|\mathbf{u})}{P(\bar{r}|\mathbf{u})} && \text{(the user-based)} \end{aligned}$$

Clearly, the conditional independence assumption may not hold in practice. For instance, items in a music playlist may be dependent upon each other. A possible solution to alleviate the dependency would be to project the feature vectors into an uncorrelated low-dimensional space by applying dimension reduction techniques such as PCA (Principle Component Analysis) [25].

Generative models

In Chapter 2 we have proposed a generative model for ranking items. The idea is that for each target item we construct a generative model (see $P(\mathbf{L}_{u_k}|i_m, r)$ in Eq. 2.6) to describe its relevant user profiles. The presence of an item in the user profile is the output of a generative process associated with that model. We then considered the user profile as a sequence of independent items, leading to a *multinomial* generative model. In Chapter 2, we binarized the data

and considered only the presence/absence counts when we estimated the generative model. Nonetheless, it is possible and straightforward to estimate the multinomial distribution without binarizing the user profile data.

The method introduced in Chapter 3 is another way to model the frequency counts in user profiles. The model has a more complete view of modelling relevance because it not only incorporates the estimation of the probability of relevance but also takes into account the non-relevance and absence counts.

Probabilistic Relevance Ranking

Collaborative filtering is concerned with making recommendations about items to users. Most formulations of the problem are specifically designed for predicting user ratings, assuming past data of explicit user ratings is available. However, in practice we may only have implicit evidence of user preference; and furthermore, a better view of the task is of generating a top-N list of items that the user is most likely to like. In this regard, we argue that collaborative filtering can be directly cast as a *relevance* ranking problem. We begin with the classic Probability Ranking Principle of information retrieval, proposing a probabilistic item ranking framework. In the framework, we derive two different ranking models, showing that despite their common origin, different factorizations reflect two distinctive ways to approach item ranking. For the model estimations, we limit our discussions to implicit user preference data, and adopt an approximation method introduced in the classic text retrieval model (i.e. the Okapi BM25 formula) to effectively decouple frequency counts and presence/absence counts in the preference data. Furthermore, we extend the basic formula by proposing the Bayesian inference to estimate the probability of relevance (and non-relevance), which largely alleviates the data sparsity problem. Apart from a theoretical contribution, our experiments on real data sets demonstrate that the proposed methods perform significantly better than other strong baselines.

This work has been accepted by Information Retrieval, Springer, 2008. The authors are J. Wang, S. Roberston, A. P. de Vries, and M. J. T. Reinders. See also [120].

3.1 Introduction

Collaborative filtering aims at identifying interesting information items (e.g. movies, books, websites) for a set of users, given their user profiles. Different from its counterpart, content-based filtering [5], it utilizes other users' preferences to perform predictions, thus making direct analysis of content features unnecessary.

User profiles can be explicitly obtained by asking users to rate items that they know. However these explicit ratings are hard to gather in a real system [15]. It is highly desirable to infer user preferences from implicit observations of user interactions with a system. These implicit interest functions usually generate frequency-counted profiles, like “playback times of a music file”, or “visiting frequency of a web-site” etc.

So far, academic research into frequency-counted user profiles for collaborative filtering has been limited. A large body of research work for collaborative filtering by default focuses on rating-based user profiles [1, 66]. Research started with memory-based approaches to collaborative filtering [37, 92, 116, 123] and lately came with model-based approaches [41, 66, 50].

In spite of the fact that these rating-based collaborative filtering algorithms lay a solid foundation for collaborative filtering research, they are specifically designed for rating prediction, making them difficult to apply in many real situations where frequency-counted user profiling is demanded. Most importantly, the purpose of a recommender system is to suggest to a user items that he or she might be interested in. The user decision on whether accepting a suggestion (i.e. to review or listen to a suggested item) is a binary one. As already demonstrated in [23, 69], directly using predicted ratings as ranking scores may not accurately model this common scenario.

This motivated us to conduct a formal study on probabilistic item ranking for collaborative filtering. We start with the Probability Ranking Principle of information retrieval [83] and introduce the concept of “binary relevance” into collaborative filtering. We directly model how likely an item might be relevant to a given user (profile), and for the given user we aim at presenting a list of items in rank order of their predicted relevance. To achieve this, we first establish an item ranking framework by employing the log-odd ratio of relevance and then derive two ranking models from it, namely an *item-based relevance model* and *user-based relevance model*. We then draw an analogy between the classic text retrieval model [87] and our models, effectively decoupling the estimations of frequency counts and (non-)relevance counts from implicit user preference data. Because data sparsity makes the probability estimations less reliable, we finally extend the basic log-odd ratio of relevance by viewing the probabili-

ties of relevance and non-relevance in the models as parameters and apply the Bayesian inference to enforce different prior knowledge and smoothing into the probability estimations. This proves to be effective in two real data sets.

The remainder of the paper is organized as follows. We first describe related work and establish the log-odd ratio of relevance ranking for collaborative filtering. The resulting two different ranking models are then derived and discussed. After that, we provide an empirical evaluation of the recommendation performance and the impact of the parameters of our two models, and finally conclude our work.

3.2 Related Work

3.2.1 Rating Prediction

In the memory-based approaches, all rating examples are stored as-is into memory (in contrast to learning an abstraction), forming a heuristic implementation of the “Word of Mouth” phenomenon. In the rating prediction phase, similar users or (and) items are sorted based on the memorized ratings. Relying on the ratings of these similar users or (and) items, a prediction of an item rating for a test user can be generated. Examples of memory-based collaborative filtering include user-based methods [9, 37, 81], item-based methods [23, 92] and unified methods [115, 116]. The advantage of the memory-based methods over their model-based alternatives is that less parameters have to be tuned; however, the data sparsity problem is not handled in a principled manner.

In the model-based approaches, training examples are used to generate an “abstraction” (model) that is able to predict the ratings for items that a test user has not rated before. In this regard, many probabilistic models have been proposed. For example, to consider user correlation, [77] proposed a method called personality diagnosis (PD), treating each user as a separate cluster and assuming a Gaussian noise applied to all ratings. It computes the probability that a test user is of the same “personality type” as other users and, in turn, the probability of his or her rating to a test item can be predicted. On the other hand, to model item correlation, [9] utilizes a Bayesian Network model, in which the conditional probabilities between items are maintained. Some researchers have tried mixture models, explicitly assuming some hidden variables embedded in the rating data. Examples include the aspect models [41, 50], the cluster model [9] and the latent factor model [10]. These methods require some assumptions about the underlying data structures and the resulting ‘compact’ models solve the data sparsity problem to a certain extent. However, the need to tune an often significant number of parameters has prevented these methods

from practical usage. For instance, in the aspect models [41, 50], an EM iteration (called "fold-in") is usually required to find both the hidden user clusters or/and hidden item clusters for any new user.

3.2.2 Item Ranking

Memory-based approaches are commonly used for rating prediction, but they can be easily extended for the purpose of item ranking. For instance, a ranking score for a target item can be calculated by a summation over its similarity towards other items that the target user liked (i.e. in the user preference list). Taking this item-based view, we formally have the following basic ranking score:

$$o_{u_k}(i_m) = \sum_{i_{m'} \in L_{u_k}} s_i(i_{m'}, i_m) \quad (3.1)$$

where u_k and i_m denote the target user and item respectively, and $i_{m'} \in L_{u_k}$ denotes any item in the preference list of user u_k . S_i is the similarity measure between two items, and in practice cosine similarity and Pearson's correlation are generally employed. To specifically target the item ranking problem, researchers in [23] proposed an alternative, TFxIDF-like similarity measure, which is shown as follows:

$$s_i(i_{m'}, i_m) = \frac{Freq(i_{m'}, i_m)}{Freq(i_{m'}) \times Freq(i_m)^\alpha} \quad (3.2)$$

where $Freq$ denotes the frequency counts of an item $Freq(i_{m'})$ or co-occurrence counts for two items $Freq(i_{m'}, i_m)$. α is a free parameter, taking a value between 0 and 1. On the basis of empirical observations, they also introduced two normalization methods to further improve the ranking.

In [114], we proposed a language modelling approach for the item ranking problem in collaborative filtering. The idea is to view an item (or its presence in a user profile) as the output of a generative process associated with each user profile. Using a linear smoothing technique [126], we have the following ranking formula:

$$o_{u_k}(i_m) = \sum_{i_{m'} \in L_{u_k}} \ln \left(\lambda P(i_{m'} | i_m) + (1 - \lambda) P(i_{m'}) \right) + \ln P(i_m) \quad (3.3)$$

where the ranking score of a target item is essentially a combination of its popularity (expressed by the prior probability $P(i_m)$) and its co-occurrence with the items in the preference list of the target user (expressed by the conditional probability $P(i_{m'} | i_m)$). $\lambda \in [0, 1]$ is used as a linear smoothing parameter to further smooth the conditional probability from a background model ($P(i_{m'})$).

Nevertheless, our formulations in [114] only take the information about presence/absence of items into account when modelling implicit user preference data, completely ignoring other useful information such as frequency counts (i.e. the number of visiting/playing times). We shall see that the probabilistic relevance framework proposed in this paper effectively extends the language modelling approaches of collaborative filtering. It not only allows us to make use of frequency counts for modelling implicit user preferences but has room to model non-relevance in a formal way. They prove to be crucial to the accuracy of recommendation in our experiments.

3.3 A Probabilistic Relevance Ranking Framework

The task of information retrieval aims to rank documents on the basis of their relevance (usefulness) towards a given user need (query). The Probability Ranking Principle (PRP) of information retrieval [83] implies that ranking documents in descending order by their probability of relevance produces optimal performance under a “reasonable” assumption, i.e. the relevance of a document to a user information need is independent of other documents in the collection [112].

By the same token, our task for collaborative filtering is to find items that are relevant (useful) to a given user interest (implicitly indicated by a user profile). The PRP applies directly when we view a user profile as a query to rank items accordingly. Hereto, we introduce the concept of “relevancy” into collaborative filtering. By analogy with the relevance models in text retrieval [56, 86, 108], the top- N recommendation items can then be generated by ranking items in order of their probability of relevance to a user profile or the underlying user interest.

To estimate the probability of relevance between an item and a user (profile), let us first define a sample space of relevance: Φ_R and let R be a random variable over the relevance space Φ_R . R is either ‘relevant’ r or ‘non-relevant’ \bar{r} . Secondly, let U be a discrete random variable over the sample space of *user id*’s: $\Phi_U = \{u_1, \dots, u_K\}$ and let I be a random variable over the sample space of *item id*’s: $\Phi_I = \{i_1, \dots, i_M\}$, where N is the number of users and M the number of items in the collection. In other words, U refers to the user identifiers and I refers to the item identifiers.

We then denote P as a probability function on the joint sample space $\Phi_U \times \Phi_I \times \Phi_R$. The PRP now states that we can solve the ranking problem by estimating the probability of relevance $P(R = r|U, I)$ and non-relevance $P(R = \bar{r}|U, I)$. The relevance ranking of items in the collection Φ_I for a given user $U = u_k$ can

be formulated as the log odds of the relevance:

$$o_{u_k}(i_m) = \ln \frac{P(r|u_k, i_m)}{P(\bar{r}|u_k, i_m)} \quad (3.4)$$

For simplicity, the propositions $R = r$, $R = \bar{r}$, $U = u_k$ and $I = i_m$ are denoted as r , \bar{r} , u_k , and i_m , respectively.

3.3.1 Item-Based Relevance Model

Two different models can be derived if we apply Bayes' rule differently. This section introduces the item-based relevance model, leaving the derivations of the user-based relevance model in Section 3.3.2.

By factorizing $P(\bullet|u_k, i_m)$ with $P(u_k|i_m, \bullet)P(\bullet|i_m)/P(u_k|i_m)$, the following log-odds ratio can be obtained from Eq. 3.4:

$$o_{u_k}(i_m) = \ln \frac{P(u_k|i_m, r)}{P(u_k|i_m, \bar{r})} + \ln \frac{P(r|i_m)}{P(\bar{r}|i_m)} \quad (3.5)$$

Notice that, in the ranking model shown in Eq. 3.5, the target user is defined in the user *id* space. For a given new user, we do not have any observations about his or her relevancy towards an unknown item. This makes the probability estimations unsolvable. In this regard, we need to build a feature representation of a new user by his or her user profile so as to relate the user to other users that have been observed from the whole collection.

This paper considers implicit user profiling: user profiles are obtained by implicitly observing user behavior, for example, the web sites visited, the music files played etc., and a user is represented by his or her preferences towards all the items. More formally, we treat a user (profile) as a vector over the entire item space, which is denoted as a bold letter $\mathbf{l} := (l^1, \dots, l^{m'}, \dots, l^M)$, where $l^{m'}$ denotes an item frequency count, e.g., number of times a user played or visited item $i_{m'}$. Note that we deliberately used the item index m' for the items in the user profile, as opposed to the target item index m . For each user u_k , the user profile vector is instantiated (denoted as \mathbf{l}_k) by assigning *this* user's item frequency counts to it: $l^{m'} = c_k^{m'}$, where $c_k^{m'} \in \{0, 1, 2, \dots\}$ denotes number of times the user u_k played or visited item $i_{m'}$. Changing the user presentation from Eq. 3.5, we have the following:

$$o_{u_k}(i_m) = \ln \frac{P(\mathbf{l}_k|i_m, r)}{P(\mathbf{l}_k|i_m, \bar{r})} + \ln \frac{P(r|i_m)}{P(\bar{r}|i_m)} = \sum_{\forall m'} \ln \frac{P(l^{m'} = c_k^{m'}|i_m, r)}{P(l^{m'} = c_k^{m'}|i_m, \bar{r})} + \ln \frac{P(r|i_m)}{P(\bar{r}|i_m)} \quad (3.6)$$

where we have assumed frequency counts of items in the target user profile are conditionally independent. Although this conditional independence assumption

does not hold in many real situations, it has been empirically shown to be a competitive approach (e.g., in text classification [27]). It is worthwhile noticing that we only ignore the item dependency in the profile of the *target* user, while for all *other* users, we do consider their dependence. In fact, how to utilise the correlations between items is crucial to the item-based approach.

For the sake of computational convenience, we intend to focus on the items ($i_{m'}$, where $m' \in \{1, M\}$) that are present in the target user profile ($c_k^{m'} > 0$). By splitting items in the user profile into two groups, i.e. presence and absence, we have:

$$\begin{aligned} o_{u_k}(i_m) &= \sum_{\forall m': c_k^{m'} > 0} \ln \frac{P(l^{m'} = c_k^{m'} | i_m, r)}{P(l^{m'} = c_k^{m'} | i_m, \bar{r})} + \sum_{\forall m': c_k^{m'} = 0} \ln \frac{P(l^{m'} = 0 | i_m, r)}{P(l^{m'} = 0 | i_m, \bar{r})} + \ln \frac{P(r | i_m)}{P(\bar{r} | i_m)} \end{aligned} \quad (3.7)$$

Both subtracting

$$\sum_{\forall m': c_k^{m'} > 0} \ln \frac{P(l^{m'} = 0 | i_m, r)}{P(l^{m'} = 0 | i_m, \bar{r})}, \quad (3.8)$$

to the first term and adding it from the second (where $\ln x - \ln y = \ln \frac{x}{y}$) gives

$$\begin{aligned} o_{u_k}(i_m) &= \left(\sum_{\forall m': c_k^{m'} > 0} \ln \frac{P(l^{m'} = c_k^{m'} | i_m, r) P(l^{m'} = 0 | i_m, \bar{r})}{P(l^{m'} = c_k^{m'} | i_m, \bar{r}) P(l^{m'} = 0 | i_m, r)} \right) \\ &\quad + \left(\sum_{\forall m'} \ln \frac{P(l^{m'} = 0 | i_m, r)}{P(l^{m'} = 0 | i_m, \bar{r})} \right) + \ln \frac{P(r | i_m)}{P(\bar{r} | i_m)} \end{aligned} \quad (3.9)$$

where the first term only deals with those items that are present in the user profile. $P(l^{m'} = c_k^{m'} | i_m, r)$ is the probability that item $i_{m'}$ occurs $c_k^{m'}$ times in a profile of a user who likes item i_m (i.e. item i_m is relevant to this user). In other words, it means, given the evidence that a user who likes item i_m , what is the probability that this user plays item $i_{m'}$ $c_k^{m'}$ times.

In summary, we have the following ranking formula:

$$o_{u_k}(i_m) = W_{u_k, i_m} + X_{i_m} + Y_{i_m} \quad (3.10)$$

where

$$W_{u_k, i_m} = \left(\sum_{\forall m': c_k^{m'} > 0} \ln \frac{P(l^{m'} = c_k^{m'} | i_m, r) P(l^{m'} = 0 | i_m, \bar{r})}{P(l^{m'} = c_k^{m'} | i_m, \bar{r}) P(l^{m'} = 0 | i_m, r)} \right) \quad (3.11)$$

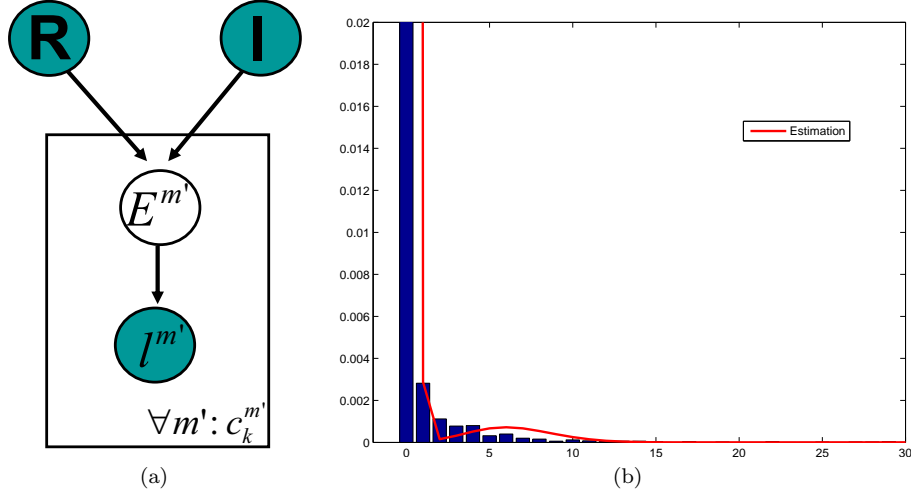


Figure 3.1: A Poisson mixture model for modelling the item occurrences in user profiles. (a) A graphical model of the Poisson mixtures. (b) An estimation of the Poisson mixtures for the Last.FM data set in the relevance case ($\lambda_0 = 0.0028$, $\lambda_1 = 6.4691$ and $p = 0.0046$).

$$X_{i_m} = \sum_{\forall m'} \ln \frac{P(l^{m'} = 0 | i_m, r)}{P(l^{m'} = 0 | i_m, \bar{r})} \quad (3.12)$$

$$Y_{i_m} = \ln \frac{P(r | i_m)}{P(\bar{r} | i_m)} \quad (3.13)$$

From the final ranking score, we observe that the relevance ranking of a target item in the item-based model is a combination between the evidence that is dependent on the target user profile (W_{u_k, i_m}) and that of the target item itself ($X_{i_m} + Y_{i_m}$). However, we shall see in Section 3.2 that, due to the asymmetry between users and items, the final ranking of the user-based model (Eq. 3.27) only requires the “user profile”-dependent evidence.

3.3.1.1 Probability Estimation

Let us look at the weighting function W_{u_k, i_m} (Eq. 3.11) first. Item occurrences within user profiles (either $P(l^{m'} = c_k^{m'} | i_m, r)$ or $P(l^{m'} = c_k^{m'} | i_m, \bar{r})$) can be modeled by a Poisson distribution. Yet, an item occurring in a user profile does not necessarily mean that this user likes this item: randomness is another explanation, particularly when the item occurs few times only. Thus, a better model would be a mixture of two Poisson models, i.e. a linear combination between a Poisson model coping with items that are “truly” liked by the user

and a Poisson model dealing with some background noise. To achieve this, we introduce a hidden random variable $E^{m'} \in \{e, \bar{e}\}$ for each of the items in the user profile, describing whether the presence of the item in a user profile is due to the fact that the user truly liked it ($E^{m'} = e$), or because the user accidentally selected it ($E^{m'} = \bar{e}$). A graphical model describing the probabilistic relationships among the random variables is illustrated in Fig. 3.1 (a). More formally, for the relevance case, we have

$$\begin{aligned} P(l^{m'} = c_k^{m'} | i_m, r) &= P(l^{m'} = c_k^{m'} | e)P(e | i_m, r) + P(l^{m'} = c_k^{m'} | \bar{e})P(\bar{e} | i_m, r) \\ &= \frac{\lambda_1^{(c_k^{m'})} \exp(-\lambda_1)}{(c_k^{m'})!} p + \frac{\lambda_0^{(c_k^{m'})} \exp(-\lambda_0)}{(c_k^{m'})!} (1 - p) \end{aligned} \quad (3.14)$$

where λ_1 and λ_0 are the two Poisson means, which can be regarded as the expected item frequency counts in the two different cases (e and \bar{e}) respectively. $p \equiv P(e | i_m, r)$ denotes the probability that the user indeed likes item i'_m , given the condition that he or she liked another item i_m . A straight-forward method to obtain the parameters of the Poisson mixtures is to apply the Expectation-Maximization (EM) algorithm [22]. To illustrate this, Fig. 3.1 (b) plots the histogram of the item frequency distribution in the Last.FM data set as well as its estimated Poisson mixtures by applying the EM algorithm.

The same can be applied to the non-relevance case. Incorporating the Poisson mixtures for the both cases into Eq. 3.11 gives

$$\begin{aligned} W_{u_k, i_m} &= \sum_{\forall m': c_k^{m'} > 0} W_{i'_m, i_m} \\ &= \sum_{\forall m': c_k^{m'} > 0} \ln \frac{(p + (\lambda_0/\lambda_1)^{(c_k^{m'})} \exp(\lambda_1 - \lambda_0)(1 - p)) (\exp(\lambda_0 - \lambda_1)q + (1 - q))}{(q + (\lambda_0/\lambda_1)^{(c_k^{m'})} \exp(\lambda_1 - \lambda_0)(1 - q)) (\exp(\lambda_0 - \lambda_1)p + (1 - p))} \end{aligned} \quad (3.15)$$

where, similarly, $q \equiv P(e | i_m, \bar{r})$ denotes the probability of the true preference of an item in the non-relevance case, while $W_{i'_m, i_m}$ denotes the ranking score obtained from the target item and the item in the user profile.

For each of the item pairs (i'_m, i_m) , we need to estimate four parameters (p, q, λ_0 and λ_1), making the model difficult to apply in practice. Furthermore, it should be emphasised that the component distributions estimated by the EM algorithm may not necessarily correspond to the two reasons that we mentioned for the presence of an item in a user profile, even if the estimated mixture distribution may fit the data well.

In this regard, this paper takes an alternative approach, approximating the ranking function by a much simpler function. In text retrieval, a similar two-

Poisson model has been proposed for modeling within-document term frequencies [35]. To make it applicable also, [87] introduced an approximation method, resulting in the widely-used BM25 weighting function for query terms. Following the same way of thinking, we can see that the weighting function for each of the items in the target user profile $W_{i'_m, i_m}$ (Eq. 3.15) has the following characteristics: 1) Function $W_{i'_m, i_m}$ increases monotonically with respect to the item frequency count $c_k^{m'}$, and 2) it reaches its upper-bound, governed by $\log(p(1-q)/q(1-p))$, when $c_k^{m'}$ becomes infinity ∞ [103, 104]. Roughly speaking, as demonstrated in Fig. 3.2, the parameters λ_0 and λ_1 can adjust the rate of the increase (see Fig. 3.2(a)), while the parameters p and q mainly control the upper bound (see Fig. 3.2(b)).

Therefore, it is intuitively desirable to approximate these two characteristics separately. Following the discussion in [87], we choose the function $c_k^{m'}/(k_3 + c_k^{m'})$ (where k_3 is a free parameter), which increases from zero to an asymptotic maximum, to model the monotonic increase with respect to the item frequency counts. Since the probabilities q and p cannot be directly estimated, a simple alternative is to use the probabilities of the presence of the item, i.e. $P(l^{m'} > 0|i_m, r)$ and $P(l^{m'} > 0|i_m, \bar{r})$ to approximate them respectively. In summary, we have the following ranking function:

$$W_{u_k, i_m} \approx \left(\sum_{\forall m': c_k^{m'} > 0} \frac{c_k^{m'}}{k_3 + c_k^{m'}} \ln \frac{P(l^{m'} > 0|i_m, r)P(l^{m'} = 0|i_m, \bar{r})}{P(l^{m'} > 0|i_m, \bar{r})P(l^{m'} = 0|i_m, r)} \right) \quad (3.16)$$

where the free parameter k_3 is equivalent to the normalization parameter of within-query frequencies in the BM25 formula [87] (also see Appendix 3.A), if we treat a user profile as a query. $P(l^{m'} > 0|i_m, r)$ (or $P(l^{m'} > 0|i_m, \bar{r})$) is the probability that item m' occurs in a profile of a user who is relevant (or non-relevant) to item i_m . Eq. 3.16 essentially decouples frequency counts $c_k^{m'}$ and presence (absence) probabilities (e.g. $P(l^{m'} > 0|i_m, r)$), thus largely simplifying the computation in practice.

Next, we consider the probability estimations of presence (absence) of items in user profiles. To handle data sparseness, different from the Robertson-Sparck Jones probabilistic retrieval (RSJ) model [86], we propose to use Bayesian inference [29] to estimate the presence (absence) probabilities. Since we have two events, either an item is present ($l^{m'} > 0$) or absent ($l^{m'} = 0$), we assume that they follow the *Bernoulli* distribution. That is, we define $\theta_{m', m} \equiv P(l^{m'} > 0|i_m, r)$, where $\theta_{m', m}$ is regarded as the parameter of a Bernoulli distribution. For simplicity, we treat the parameter as a random variable and estimate its value by maximizing an *a posteriori* probability. Formally we have

$$\hat{\theta}_{m', m} = \arg \max_{\theta_{m', m}} p(\theta_{m', m} | r_{m', m}, R_m; \alpha_r, \beta_r) \quad (3.17)$$

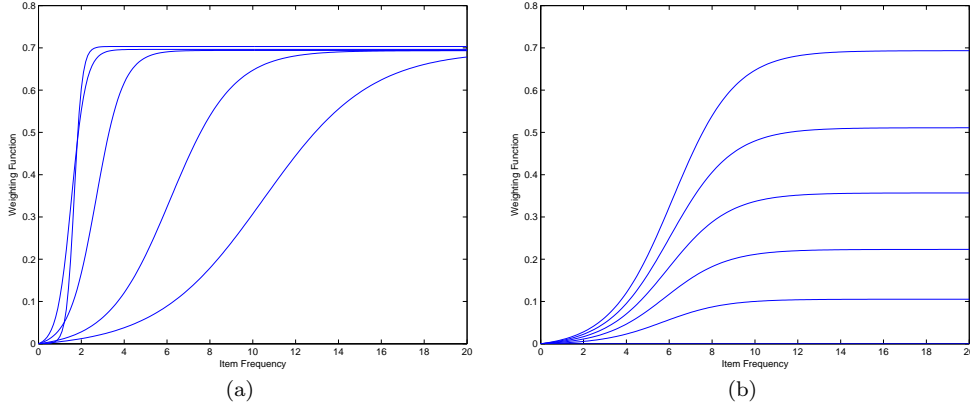


Figure 3.2: The relationship between weighting function $W_{i'_m, i_m}$ and its four parameters λ_0 , λ_1 , p and q . We plot ranking score $W_{i'_m, i_m}$ against various item frequency counts $c_k^{m'}$ from 0 to 20. (a) We fix $\lambda_0 = 0.02$, $p = 0.02$ and $q = 0.010$, and vary $\lambda_1 \in \{0.03, 0.04, 0.1, 0.4, 5\}$. (b) We fix $\lambda_0 = 0.02$, $\lambda_1 = 0.04$ and $p = 0.02$, and vary $q \in \{0.020, 0.018, 0.016, 0.014, 0.012, 0.010\}$.

where R_m denotes the number of user profiles that are relevant to an item i_m , and among these user profiles, $r_{m',m}$ denotes the number of the user profiles where an item $i_{m'}$ is present. This establishes a contingency table for each item pair (shown in Table 3.1). In addition, we choose the Beta distribution as the prior (since it is the conjugate prior for the Bernoulli distribution), which is denoted as $Beta(\alpha_r, \beta_r)$. Using the conjugate prior, the posterior probability after observing some data turns to the Beta distribution again with updated parameters.

$$p(\theta_{m',m} | r_{m',m}, R_m; \alpha_r, \beta_r) \propto \theta_{m',m}^{r_{m',m} + \alpha_r - 1} (1 - \theta_{m',m})^{R_m - r_{m',m} + \beta_r - 1} \quad (3.18)$$

Maximizing an a posteriori probability in Eq. 3.18 (i.e. taking the mode) gives the estimation of the parameter [29]

$$\hat{\theta}_{m',m} = \frac{r_{m',m} + \alpha_r - 1}{R_m + \alpha_r + \beta_r - 2} \quad (3.19)$$

Following the same reasoning, we obtain the probability of item occurrences in the non-relevance case.

$$P(l^{i_{m'}} > 0 | i_m, \bar{r}) \equiv \hat{\gamma}_i = \frac{n_{m'} - r_{m',m} + \alpha_{\bar{r}} - 1}{K - R_m + \alpha_{\bar{r}} + \beta_{\bar{r}} - 2} \quad (3.20)$$

where we used $\hat{\gamma}_i$ to denote $P(l^{i_{m'}} > 0 | i_m, \bar{r})$. $\alpha_{\bar{r}}$ and $\beta_{\bar{r}}$ are again the parameters of the conjugate prior ($Beta(\alpha_{\bar{r}}, \beta_{\bar{r}})$), while $n_{m'}$ denotes the number of

Table 3.1: Contingency table of relevance v.s. occurrence: Item Model

| | Item i_m is Relevant | Item i_m is NOT Relevant | |
|-----------------------------------|------------------------|-----------------------------------|--------------|
| Item $i_{m'}$ Contained in UP | $r_{m',m}$ | $n_{m'} - r_{m',m}$ | $n_{m'}$ |
| Item $i_{m'}$ NOT Contained in UP | $R_m - r_{m',m}$ | $(N - R_m) - (n_{m'} - r_{m',m})$ | $K - n_{m'}$ |
| | R_m | $K - R_m$ | K |

Table 3.2: Contingency table of relevance v.s. occurrence: User Model

| | User u_k is Relevant | User u_k is NOT Relevant | |
|-----------------------------------|------------------------|-----------------------------------|--------------|
| User $u_{k'}$ Contained in UP | $r_{k',k}$ | $n_{k'} - r_{k',k}$ | $n_{k'}$ |
| User $u_{k'}$ NOT Contained in UP | $R_k - r_{k',k}$ | $(M - R_k) - (n_{k'} - r_{k',k})$ | $M - n_{k'}$ |
| | R_k | $M - R_k$ | M |

times that item $i_{m'}$ is present in a user profile (See Table 1). Replacing Eq. 3.19 and Eq. 3.20 into Eq. 3.16, we have

$$\begin{aligned}
W_{u_k, i_m} &\approx \sum_{\forall m': c_k^{m'}} \frac{c_k^{m'}}{k_3 + c_k^{m'}} \ln \frac{\hat{\theta}_i(1 - \hat{\gamma}_i)}{\hat{\gamma}_i(1 - \hat{\theta}_i)} \\
&= \sum_{\forall m': c_k^{m'}} \frac{c_k^{m'}}{k_3 + c_k^{m'}} \ln \frac{(r_{m',m} + \alpha_r - 1)((K - R_m) - (n_{m'} - r_{m',m}) + \beta_{\bar{r}} - 1)}{(n_{m'} - r_{m',m} + \alpha_{\bar{r}} - 1)(R_m - r_{m',m} + \beta_r - 1)}
\end{aligned} \tag{3.21}$$

The four hyper-parameters $(\alpha_r, \alpha_{\bar{r}}, \beta_r, \beta_{\bar{r}})$ can be treated as pseudo frequency counts. Varying choices for them leads to different estimators [125]. In the information retrieval domain [86, 87], adding an extra 0.5 count for each probability estimation has been widely used to avoid zero probabilities. This choice corresponds to set tiny constant values $\alpha_r = \alpha_{\bar{r}} = \beta_r = \beta_{\bar{r}} = 1.5$. We shall see that in the experiments collaborative filtering needs relatively bigger pseudo counts for the non-relevance and/or absence estimation $(\alpha_{\bar{r}}, \beta_r \text{ and } \beta_{\bar{r}})$. This can be explained because using absence to model non-relevance is noisy, so more smoothing is needed. If we define a free parameter v and set it to be equal to $\alpha_r - 1$, we have the generalized *Laplace smoothing* estimator. Alternatively, the prior can be fit on a distribution of the given collection [126].

Applying the Bayesian inference similarly, we obtain X_{i_m} as follows:

$$\begin{aligned}
X_{i_m} &= \sum_{i_{m'}} \ln \frac{P(l_{i_{m'}} = 0 | i_m, r)}{P(l_{i_{m'}} = 0 | i_m, \bar{r})} \\
&= \sum_{i_{m'}} \ln \frac{(K - R_m + \alpha_r + \beta_r - 2)(R_m - r_{m',m} + \beta_r - 1)}{(R_m + \alpha_r + \beta_r - 2)(K - R_m - (n_{m'} - r_{m',m}) + \beta_{\bar{r}} - 1)}
\end{aligned} \tag{3.22}$$

For the last term, the popularity ranking Y_{i_m} , we have

$$Y_{i_m} = \ln \frac{P(r|i_m)}{P(\bar{r}|i_m)} = \ln \frac{R_m}{K - R_m} \quad (3.23)$$

Notice that in the initial stage, we do not have any relevance observation of item i_m . We may assume that if a user played the item frequently (say played more than t times), we treat this item being relevant to this user's interest. By doing this, we can also construct the contingency table to be able to estimate the probabilities.

3.3.2 User-Based Relevance Model

Applying Bayes' rule differently results in the following formula from Eq. 3.4:

$$o_{u_k}(i_m) = \ln \frac{P(i_m|u_k, r)}{P(i_m|u_k, \bar{r})} + \ln \frac{P(r|u_k)}{P(\bar{r}|u_k)} \quad (3.24)$$

Similarly, using frequency counts over a set of users ($l^1, \dots, l^{k'}, \dots, l^K$) to represent the target item i_m , we get

$$\begin{aligned} S_{u_k}(i_m) = & \sum_{\forall k': c_{k'}^m > 0} \ln \frac{P(l^{k'} = c_{k'}^m | u_k, r) P(l^{k'} = 0 | u_k, \bar{r})}{P(l^{k'} = c_{k'}^m | u_k, \bar{r}) P(l^{k'} = 0 | u_k, r)} \\ & + \sum_{\forall k'} \ln \frac{P(l^{k'} = 0 | u_k, r)}{P(l^{k'} = 0 | u_k, \bar{r})} + \ln \frac{P(r|u_k)}{P(\bar{r}|u_k)} \end{aligned} \quad (3.25)$$

where the last two terms in the formula are independent of target items, they can be discarded. Thus we have

$$S_{u_k}(i_m) \propto_{u_k} \sum_{\forall k': c_{k'}^m > 0} \ln \frac{P(l^{k'} = c_{k'}^m | u_k, r) P(l^{k'} = 0 | u_k, \bar{r})}{P(l^{k'} = c_{k'}^m | u_k, \bar{r}) P(l^{k'} = 0 | u_k, r)} \quad (3.26)$$

where \propto_{u_k} denotes same rank order with respect to u_k .

Following the same steps (the approximation to two-Poisson distribution and the MAP probability estimation) as discussed in the previous section gives

$$\begin{aligned} S_{u_k}(i_m) \propto_{u_k} & \sum_{\forall k': c_{k'}^m > 0} \frac{c_{k'}^m}{\mathcal{K} + c_{k'}^m} \ln \frac{P(l^{k'} > 0 | u_k, r) P(l^{k'} = 0 | u_k, \bar{r})}{P(l^{k'} > 0 | u_k, \bar{r}) P(l^{k'} = 0 | u_k, r)} \\ = & \sum_{\forall k': c_{k'}^m > 0} \frac{c_{k'}^m}{\mathcal{K} + c_{k'}^m} \ln \frac{(r_{k',k} + \alpha_r - 1)(M - n_{k'} - R_k + r_{k',k} + \beta_{\bar{r}} - 1)}{(n_{k'} - r_{k',k} + \alpha_r \bar{r} - 1)(R_k - r_{k',k} + \beta_r - 1)} \end{aligned} \quad (3.27)$$

Table 3.3: Characteristics of the test data sets.

| | Last.FM | Del.icio.us |
|---------------------------|---------|-------------|
| Num. of Users | 2408 | 1731 |
| Num. of Items | 1399 | 3370 |
| Zero Occurrences in UP(%) | 96.8% | 96.7% |

where $\mathcal{K} = k_1((1-b)+bL_m)$. k_1 is the normalization parameter of the frequency counts for the target item, L_m is the normalized item popularity (how many times the item i_m has been “used”) (i.e. the popularity of this item divided by the average popularity in the collection), and $b \in [0, 1]$ denotes the mixture weight. Notice that if we treat an item as a document, the parameter k_1 is equivalent to the normalization parameter of within-document frequencies in the BM25 formula (see Appendix 3.A). Table 3.2 shows the contingency table of user pairs.

3.3.3 Discussions

Previous studies on collaborative filtering, particularly memory-based approaches, make a distinction between user-based [9, 37, 81] and item-based approaches [23, 92]. Our probabilistic relevance models were derived with an information retrieval view on collaborative filtering. They demonstrated that the user-based (relevance) and item-based (relevance) models are equivalent from a probabilistic point of view, since they have actually been derived from the same generative relevance model. The only difference corresponds to the choice of independence assumptions in the derivations, leading to the two different factorizations. But statistically they are inequivalent because the different factorizations lead to the different probability estimations; In the item-based relevance model, the item-to-item relevancy is estimated while in the user-based one, the user-to-user relevancy is required instead. We shall see shortly in our experiments that the probability estimation is one of the important factors influencing recommendation performance.

3.4 Experiments

3.4.1 Data Sets

The standard data sets used in the evaluation of collaborative filtering algorithms (i.e. MovieLens and Netflix) are rating-based, which are not suitable for testing our method using implicit user profiles. This paper adopts two implicit

user profile data sets.

The first data set comes from a well known social music web site: **Last.FM**. It was collected from the play-lists of the users in the community by using a plug-in in the users' media players (for instance, Winamp, iTunes, XMMS etc). Plug-ins send the title (song name and artist name) of every song users play to the Last.FM server, which updates the user's musical profile with the new song. For our experiments, the triple {userID, artistID, Freq} is used.

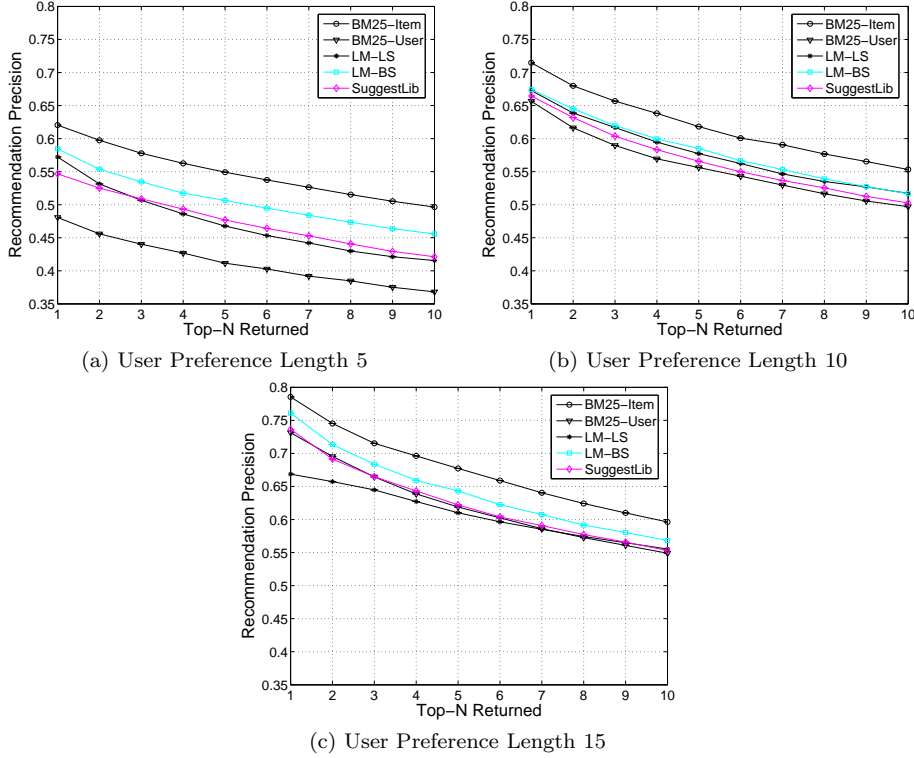
The second data set was collected from one well-known collaborative tagging Web site, **del.icio.us**. Unlike other studies focusing on directly recommending contents (Web sites), here we intend to find relevance tags on the basis of user profiles as this is a crucial step in such systems. For instance, the tag suggestion is needed in helping users assigning tags to new contents, and it is also useful when constructing a personalized "tag cloud" for the purpose of exploratory search [121] (see also Chapter 4). The Web site has been crawled between May and October 2006. We collected a number of the most popular tags, found which users were using these tags, and then downloaded the whole profiles of these users. We extracted the triples {userID, tagID, Freq} from each of the user profiles. User IDs are randomly generated to keep the users anonymous. Table 3.3 summarizes the basic characteristics of the data sets¹.

3.4.2 Experiment Protocols

For 5-fold cross-validation, we randomly divided this data set into a training set (80% of the users) and a test set (20% of the users). Results are obtained by averaging 5 different runs (sampling of training/test set). The training set was used to estimate the model. The test set was used for evaluating the accuracy of the recommendations on the new users, whose user profiles are not in the training set. For each test user, 5, 10, or 15 items of a test user were put into the user profile list. The remaining items were used to test the recommendations.

In information retrieval, the effectiveness of the document ranking is commonly measured by *precision* and *recall* [2]. Precision measures the proportion of retrieved documents that are indeed relevant to the user's information need, while recall measures the fraction of all relevant documents that are successfully retrieved. In the case of collaborative filtering, we are, however, only interested in examining the accuracy of the top-N recommended items, while paying less attention to finding *all* the relevant items. Thus, our experiments here only consider the *recommendation precision*, which measures the proportion of recommended items that are ground truth items. Note that the items

¹The two data sets can be downloaded from <http://ict.ewi.tudelft.nl/~jun/CollaborativeFiltering.html>.

Figure 3.3: Precision of different methods in the `Last.FM` Data Set.

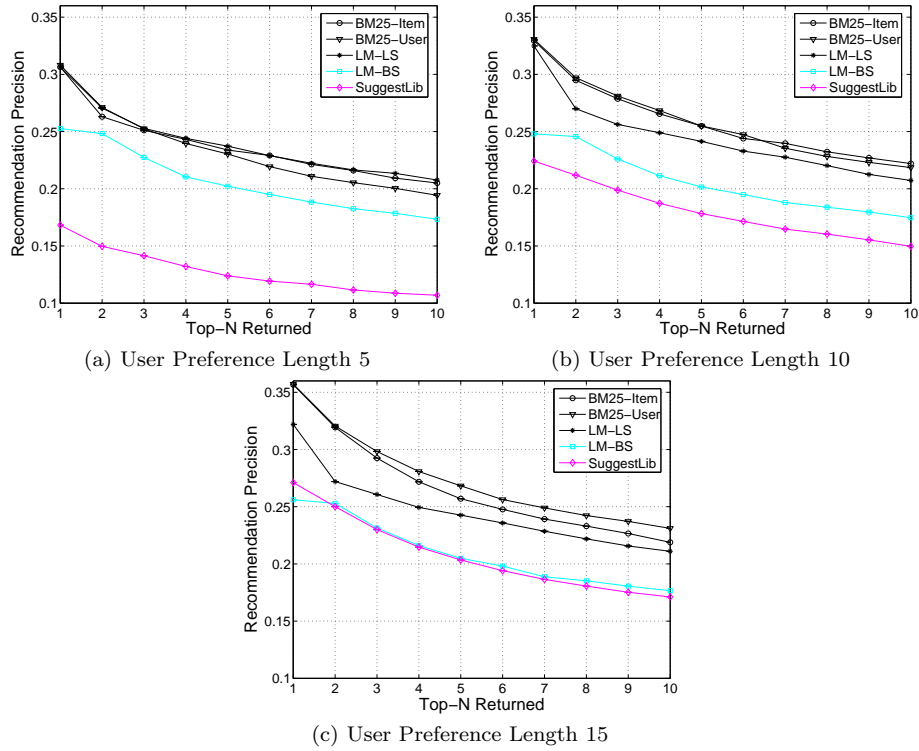
in the profiles of the test user represent only a fraction of the items that the user *truly* liked. Therefore, the measured precision *underestimates* the true precision.

3.4.3 Performance

We choose the state-of-the-art item ranking algorithms that have been discussed in Section 3.2.2 as our baselines. For the method proposed in [23], we adopt their implementation, the top-N suggest recommendation library², which is denoted as `SuggestLib`. We also implement the language modelling approach of collaborative filtering in [114] and denote this approach as `LM-LS` while its variant using the Bayes' smoothing (i.e., a Dirichlet prior) is denoted as `LM-BS`. To make a comparison, the parameters of the algorithms are set to the optimal ones.

We set the parameters of our two models to the optimal ones and compare

²<http://glaros.dtc.umn.edu/gkhome/suggest/overview>

Figure 3.4: Precision of different methods in the `Del.icio.us` Data Set.

them with these strong baselines. The item-based relevance model is denoted as `BM25-Item` while the user-based relevance model is denoted as `BM25-User`. Results are shown in Fig. 3.3 and 3.4 over different returned items. Let us first compare the performance of the `BM25-Item` and `BM25-User` models. For the `Last.FM` data set (Fig. 3.3), the item-based relevance model consistently performs better than the user-based relevance model. This confirms a previous observation that item-to-item similarity (relevancy) in general is more robust than user-to-user similarity [92]. However, if we look at the `del.icio.us` data (Fig. 3.4), the performance gain from the item-based relevance model is not clear any more - we obtain a mixture result and the user-based one even outperforms the item-based one when the number of items in user preferences is set to 15 (see Fig. 3.4 (c)). We think this is because the characteristics of data set play an important role for the probability estimations in the models. In the `Last.FM` data set, the number of users is larger than the number of items (see Table 3.3). It basically means that we have more observations from the user side about the item-to-item relevancy while having less observations from the item side about user-to-user relevancy. Thus, in the `Last.FM` data set, the probability estimation for the item based relevance model is more reliable than

Table 3.4: Comparison with the other approaches. Precision is reported in the Last.FM data set. The best results are in bold type. A Wilcoxon signed-rank test is conducted and the significant ones (P-value < 0.05) over the second best are marked as *.

| | Top-1 | Top-3 | Top-10 |
|------------------|---------------|---------------|---------------|
| BM25-Item | 0.620* | 0.578* | 0.497* |
| LM-LS | 0.572 | 0.507 | 0.416 |
| LM-BS | 0.585 | 0.535 | 0.456 |
| SuggestLib | 0.547 | 0.509 | 0.421 |

(a) User Profile Length 5

| | Top-1 | Top-3 | Top-10 |
|------------------|---------------|---------------|---------------|
| BM25-Item | 0.715* | 0.657* | 0.553* |
| LM-LS | 0.673 | 0.617 | 0.517 |
| LM-BS | 0.674 | 0.620 | 0.517 |
| SuggestLib | 0.664 | 0.604 | 0.503 |

(b) User Profile Length 10

| | Top-1 | Top-3 | Top-10 |
|------------------|---------------|---------------|---------------|
| BM25-Item | 0.785* | 0.715* | 0.596* |
| LM-LS | 0.669 | 0.645 | 0.555 |
| LM-BS | 0.761 | 0.684 | 0.568 |
| SuggestLib | 0.736 | 0.665 | 0.553 |

(c) User Profile Length 15

Table 3.5: Comparison with the other approaches. Precision is reported in the Del.icio.us data set. The best results are in bold type. A Wilcoxon signed-rank test is conducted and the significant ones (P-value < 0.05) over the second best are marked as *.

| | Top-1 | Top-3 | Top-10 |
|------------------|--------------|--------------|--------------|
| BM25-Item | 0.306 | 0.251 | 0.205 |
| LM-LS | 0.306 | 0.253 | 0.208 |
| LM-BS | 0.253 | 0.227 | 0.173 |
| SuggestLib | 0.168 | 0.141 | 0.107 |

(a) User Profile Length 5

| | Top-1 | Top-3 | Top-10 |
|------------------|--------------|---------------|---------------|
| BM25-Item | 0.329 | 0.279* | 0.222* |
| LM-LS | 0.325 | 0.256 | 0.207 |
| LM-BS | 0.248 | 0.226 | 0.175 |
| SuggestLib | 0.224 | 0.199 | 0.150 |

(b) User Profile Length 10

| | Top-1 | Top-3 | Top-10 |
|------------------|---------------|---------------|---------------|
| BM25-Item | 0.357* | 0.292* | 0.219* |
| LM-LS | 0.322 | 0.261 | 0.211 |
| LM-BS | 0.256 | 0.231 | 0.177 |
| SuggestLib | 0.271 | 0.230 | 0.171 |

(c) User Profile Length 15

that of the user-based relevance model. But in the del.icio.us data set (see Table 3.3), the number of items is larger than the number of users. Thus we

have more observations about user-to-user relevancy from the item side, causing a significant improvement for the user-based relevance model.

Since the item-based relevance model in general outperforms the user-based relevance model, we next compare the item-based relevance model with other methods (shown in Table 3.4 and 3.5). From the tables, we can see that the item-based relevance model performs consistently better than the **SuggestLib** method over all the configurations. A Wilcoxon signed-rank test [45] is done to verify the significance. We also observe that in most of the configurations our item-based model significantly outperforms the language modelling approaches, both the linear smoothing and the Bayesian smoothing variants. We believe that the effectiveness of our model is due to the fact that the model naturally integrates frequency counts and probability estimation of non-relevance into the ranking formula, apart from other alternatives.

3.4.4 Parameter Estimation

This section tests the sensitivity of the parameters, using the `del.icio.us` data set. Recall that for both the item-based relevance model (shown in Eq. 3.10) and the user-based relevance model (shown in Eq. 3.27), we have frequency smoothing parameter k_1 (and b) or k_3 , and co-occurrence smoothing parameters α and β . We first test the sensitivity of the frequency smoothing parameters. Fig. 3.5 shows recommendation precision against the parameters k_1 and b of the user-based relevance model while Fig. 3.6 shows recommendation precision varying the parameter k_3 of the item relevance model. The optimal values in the figures demonstrate that using both the frequency smoothing parameters (k_1 and k_3) and the length normalization parameter b , inspired by the BM25 formula, indeed improves the recommendation performance. We also observe that these parameters are relatively insensitive to different data sets and their different sparsity setups.

Next we fix the frequency smoothing parameters to the optimal ones and test the co-occurrence smoothing parameters for both models. Fig. 3.7 and Fig. 3.8 plot the smoothing parameters against the recommendation precision. More precisely, Fig. 3.7 (a) and Fig. 3.8 (a) plot the smoothing parameter for the relevance part $v_1 = \alpha_r - 1$ while Fig. 3.7 (b) and Fig. 3.8 (b) plot that of the non-relevance or absence parts; all of them are set to be equal ($v_2 = \alpha_{\bar{r}} - 1 = \beta_r - 1 = \beta_{\bar{r}} - 1$) in order to minimize the number of parameters while still retaining comparable performance. From the figures, we can see that the optimal smoothing parameters (pseudo counts) of the relevance part v_1 are relatively small, compared to those of the non-relevance part. For the user-based relevance model, the pseudo counts of the non-relevance estimations are in the range between 10 and 15 (Fig. 3.7(b)) while for the item-based relevance

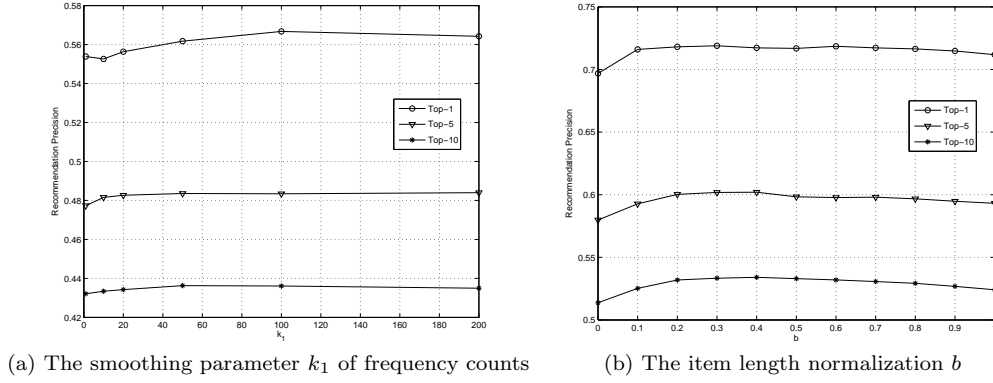
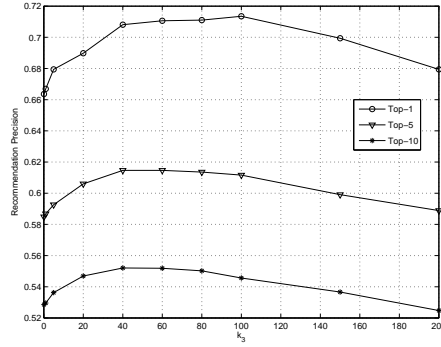


Figure 3.5: Parameters in the user-based relevance model.

Figure 3.6: The smoothing parameter of frequency counts k_3 in the item-based relevance model.

model, they are in the range of $[50, 100]$ (Fig. 3.8(b)). It is due to the fact that the non-relevance estimation is not as reliable as the relevance estimation and thus more smoothing is required.

3.5 Conclusions

This paper proposed a probabilistic item ranking framework for collaborative filtering, which is inspired by the classic probabilistic relevance model of text retrieval and its variants [86, 87, 103, 104]. We have derived two different models in the relevance framework in order to generate top-N item recommendations. We conclude from the experimental results that the proposed models are indeed effective, and significantly improve the performance of the top-N

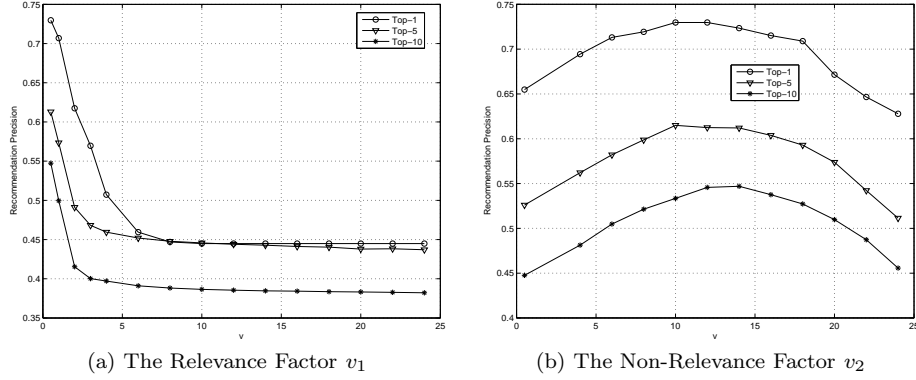


Figure 3.7: The relevance/non-Relevance smoothing parameters in the user-based relevance model.

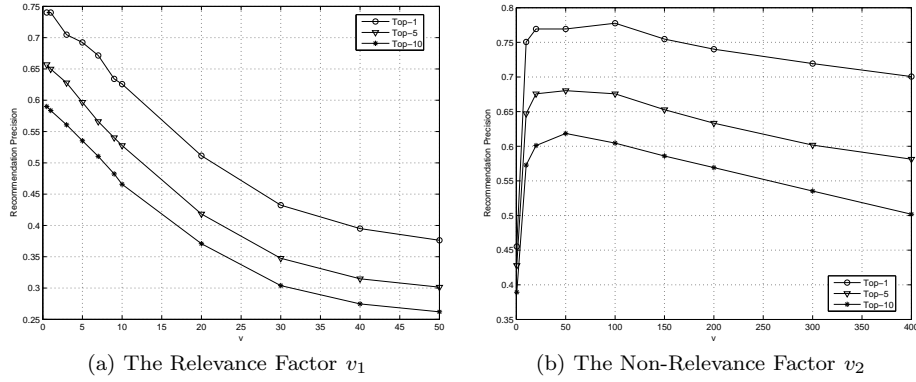


Figure 3.8: The relevance/non-relevance smoothing parameters in the item-based relevance model.

item recommendations.

In current settings, we fix a threshold when considering frequency counts as relevance observations. In the future, we may also consider graded relevance with respect to the number of times a user played an item. To do this, we may weight (sampling) the importance of the user profiles according to the number of times the user played/reviewed an item when we construct the contingency table. In current models, the hyperparameters are obtained by using cross-validation. In the future, it is worthwhile investigating the evidence approximation framework [6] by which the hyperparameters can be estimated from the whole collection; or we can take a full Bayesian approach that integrates over the hyperparameters and the model parameters by adopting variational methods [51].

It has been seen in this paper that relevance is a good concept to explain the

correspondence between user interest and information items. We have set up a close relationship between the probabilistic models of text retrieval and these of collaborative filtering. It facilitates a flexible framework to tryout more of the techniques that have been used in text retrieval to the related problem of collaborative filtering. For instance, relevance observations can be easily incorporated in the framework once we have relevance feedback from users. An interesting observation is that, different from text retrieval, relevance feedback for a given user in collaborative filtering is not dependent of this user's "query" (a user profile) only. It instead has a rather global impact, and affects the representation of the whole collection; Relevance feedback from one user could influence the ranking order of the other users. It is also worthwhile investigating query expansion by including more relevant items as query items or re-calculating (re-constructing) the contingency table according to the relevance information.

Finally, a combination of the two relevance models is of interest [115, 116]. This has some analogies with the "unified model" idea in information retrieval [85]. However, there are also some differences: in information retrieval, based on explicit features of items and explicit queries, simple user relevance feedback relates to the current query only, and a unified model is *required* to achieve the global impact which we have already identified in the present (non-unified) models for collaborative filtering. These subtle differences make the exploration of the unified model ideas particularly attractive.

3.A The Okapi BM25 Document Ranking Score

To make the paper self-contained and facilitate the comparison between the proposed model and the BM25 model of text retrieval [87, 104], here we summarise the Okapi BM25 document ranking formula. The commonly-used ranking function $S_q(d)$ of a document d given a query q is expressed as follows:

$$S_q(d) = \sum_{\forall t: c_q^t > 0} \log \frac{(k3 + 1)c_q^t}{k3 + c_q^t} \frac{(k1 + 1)c_d^t}{\mathcal{K} + c_d^t} \log \frac{(r_t + 0.5)(N - n_t - R + r_t + 0.5)}{(n_t - r_t + 0.5)(R - r_t + 0.5)} \quad (3.28)$$

where

- c_q^t denotes the within query frequency of a term t at query q , while c_d^t denotes the with document frequency of a term t at document d .
- $k1$ and $k3$ are constants. The factors $k3 + 1$ and $k1 + 1$ are unnecessary here, but help scale the weights. For instance, the first component is 1 when $c_q^t = 1$.
- $\mathcal{K} \equiv k1((1 - b) + bL_d)$. L_d is the normalised document length (i.e. the length of this document d divided by the average length of documents in the collection). $b \in [0, 1]$ is constant.
- n_t is the number of documents in the collection indexed by this term t .
- N is the total number of documents in the collection.
- r_t is the number of relevant documents indexed by this term t .
- R is the total number of relevant documents.

For detailed information about the model and its relationship with the Robertson-Sparck Jones probabilistic retrieval (RSJ) model [86], we refer to [87, 104].

Commentary on Chapter 3

Relevance and Beyond

Chapters 2 and 3 focused on item relevance ranking. Our probabilistic models of item relevance are based on the probability ranking principle (PRP) of information retrieval, requiring an assumption that “the relevance of an information item to a request is independent of other information items in the collection” [83, 112]. Under this assumption, the PRP is optimal in terms of precision and recall metrics [83]. However, it is worth noting that relevance may not necessarily be the only concern when generating a top- N item list. For example, in music recommendation scenarios, we expect to recommend music items from the artists that we like, but we might also require that these artists are new to us – we need a certain unexpectedness and novelty for the recommendation [128]. Moreover, user interests may have multiple aspects (subtopics). It may be difficult to decide which one is the suitable one. One of the solutions, as we shall introduce in Chapter 4, is to ask users to specify their current interest at hand and rank items accordingly. Another solution is to develop measurements for these qualities and model them separately from the relevance [11, 128, 130]; the system developed in [127] separates the consideration of novelty and redundancy and re-ranks the retrieval results so that the top list includes some documents for each subtopic. It is also interesting to develop a method that can directly integrate diversity or novelty into the probabilistic relevance modelling. For example, Chen and Karger [12] argue that, instead of using the PRP, a retrieval system should rank documents in order to maximize the probability of finding a relevant document among the top N . A greedy heuristic they developed shows that, for each new entry in the top- N document list, one should select documents according to the probability of relevance conditioned on an assumption that all the past selected documents in the list were non-relevant. This intuitively provides a natural diversity requirement.

Our study focuses on developing relevance models for collaborative filtering. To evaluate the effectiveness of the proposed recommendation algorithms, we limit our experiments to widely used metrics such as precision (Chapters 2, 3, and

4) and the mean absolute error (MAE) (Chapters 5 and 6). We are aware that our research is user-centered, and user satisfaction with recommender systems has various aspects. How to develop evaluation metrics with respect to these aspects is of great interest, but the answer to the question is, however, beyond the scope of this thesis. We refer to [38] for the detailed discussions and studies.

Item Correlations

The approximation method given in Eq. 3.16 indicates that the weight function $W_{i'_m, i_m}$ increases monotonically with respect to the item frequency count c_k^m in the target user profile. This might be a preferable property for text retrieval, because in most cases, query terms are chosen to be positively associated with relevance. In recommender systems, however, correlations between an item in the user profile and a target item may be negative. In such a situation, we expect that the weighting function should decrease monotonically with respect to the item frequency count. The memory-based methods that adopt Pearson's correlation [9] naturally take the negative factor into account. This is of great interest in investigating the negative influence in the two-Poisson model (Eq. 3.15) and the relation of the two-Poisson model to these heuristic methods in the future.

Chapter 4

Personalized Collaborative Tagging

As a result of the popularity of collaborative tagging systems, we have witnessed a substantial increase in *user-generated content* and *user-generated metadata* (tags). These systems capture user preferences while their users share content and metadata. This paper investigates how user preferences can be exploited for the personalization tasks in collaborative tagging systems. We consider two generative processes that generate tagging data and create probabilistic models for collaborative indexing, collaborative browsing and collaborative search, the three main steps in tagging systems. The individual user's tagging history is integrated in the recommendation of tags and items to make the suggestions more specific and focused on the user's task. From the proposed models, we demonstrate that, regardless of the task, the underlying personalized ranking should consider both the personal behavior and other users' behavior together. Experiments on two real data sets show that our personalized models significantly outperform the non-personalized ones for all the three tasks.

This work is under a journal submission. Authors are J. Wang, J. Yang, M. Clements, A. P. de Vries and M. J. T. Reinders. See also [121].

4.1 Introduction

Recently, as a new web content production circle, user-generated content has enjoyed an enormous growth. We have seen a shift among web content publishers from creating on-line content themselves to providing the facilities and playgrounds for end users to publish their self-produced content, such as bookmarks (del.icio.us), photographs (flickr.com), research papers (CiteULike.org) and video clips (YouTube.com).

Collaborative tagging systems have emerged to facilitate the procedures of tagging (annotating), sharing and exploring content over the established social networks. Fig. 4.1 illustrates the workflow of a common collaborative tagging system. In general, there are two phases during the process, namely the *indexing* phase and the *exploratory search* phase. In the indexing phase, users tag content that they are interested in. The tagged content could be injected by the users themselves, for instance the photos in *Flickr* and videos in *YouTube*, or could come from other sources, for instance the web URLs in *del.icio.us* and the scientific papers in *CiteULike*. Aggregating from millions of users creates a large amount of user-generated content and their associated tags. In the exploratory search phase, these systems allow users to search and explore relevant content using these associated tags.

The amount of user-generated content is increasing far more quickly than our capability to digest it. Compared to professionally produced content and meta data, collaborative tagging systems face the challenge that end-users assign tags in an uncontrolled manner, resulting in unsystematic and inconsistent meta-data. This calls for extensive support for suggesting tags in the indexing phase, and steering users towards their personal interests in the exploratory search phase. We identify three opportunities for personalization in collaborative tagging systems:

In the indexing phase:

- 1. Collaborative Indexing:** personalizing the tagging process when a user assigns tags to label (index) certain content. Tags act as an indication of “aboutness” towards items. But, most users are not professional to describe content by tags precisely, and are insufficiently aware of tags in use by others. For instance, users might tag the same content using “computer game”, “computer-game” or “computer_games”. Ideally, the system should suggest tags from the common vocabulary that fit the user’s intention or taste while remaining consistent with other users (shown in Fig. 4.1). As a result, users discover suitable tagging keywords more easily, and, more importantly, inconsistent tagging behavior is reduced. This way, every user benefits from and builds upon the information contributed by others; it has even been claimed that this support for suggesting tags when a user is asked to label a certain

item would lead to a true “folksonomy” (e.g. coherent categorization schemes) [32, 67]. We shall see shortly how our tag suggestion model (Section 4.3.1.1) reinforces tags that have been frequently used by the target user as well as other users.

In the exploratory search phase:

2. Collaborative Browsing: personalizing tag exploration when a user starts to browse for relevant content. Navigation through tags provides an effective way to explore and discover relevant content. To initiate the navigation, current collaborative tagging systems make use of “tag clouds”, a visual representation of the set of most popular tags [28, 70]. Popularity-based exploration is however limited, as it does not necessarily fit an individual user’s need: different users may have very different preferences. Personalizing tag exploration could alleviate the search cost and improve the retrieval performance. The formulation of the proposed collaborative browsing model is allocated in Section 4.3.2.1.

3. Collaborative Item Search: personalizing item search when a user chooses a tag (as a query). After identifying interesting tags, users click or issue these tags to further explore the related content (items). To rank items, most of the existing collaborative tagging systems rely on merely its association with the query tag, where usually a combination of the item’s popularity and “freshness” are employed as ranking criteria. However, it is well-known in text retrieval that, due to its ambiguity, a term (tag) alone is not semantically and contextually expressive enough to represent the needs for a particular user. For example, the term “apple” can be referring to a type of fruit, a computer brand or even a city. In this regard, an optimal relevance ranking should utilize as much extra information as possible to clarify the user’s needs. It is worthwhile investigating the usage of user preference in order to facilitate the personalized item search. The formulation of the personalized item search model can be found in Section 4.3.2.2.

Considering two generative processes in the tagging data, this paper proposes three types of task-focussed ranking models to solve the three aforementioned personalization problems in a unified framework. We show how the underlying personalized ranking scores for a given candidate (an item or a tag depending on the task) consist of the popularity of the candidate and its likelihood towards the user preference. For probability estimation, we consider different types of generative processes in the tagging data, where the smoothing methods are naturally integrated. We then choose an optimal candidate model for each of the three problems introduced above, and estimate the probability of the user preferences being generated from that candidate model. Our experiments on two real data sets demonstrate the effectiveness of the methods, showing that all the three personalized models perform significantly better than the non-personalized ones, while the collaborative browsing model outperforms the ranking-based collaborative filtering approaches when we have more user pref-

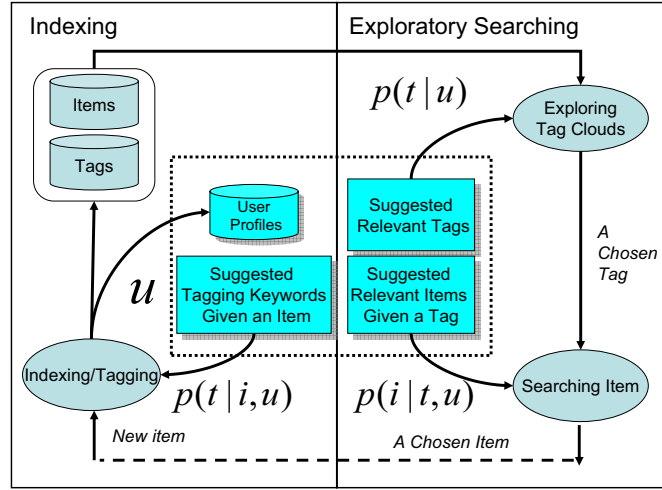


Figure 4.1: Personalized Collaborative Tagging.

ferences data.

The remainder of the paper is organized as follows. We first summarize related work. We then introduce the generative processes of the tagging data and the derived ranking models for the three suggestion problems of indexing, browsing and search. We provide an empirical evaluation of the performance of our models for the three ranking tasks and the impact of parameters, and finally conclude our work.

4.2 Related Work

Collaborative tagging systems have recently emerged as tools that assist to find structure in online database and user-generated content. As an example, Golder and Huberman conducted an investigation of *del.icio.us*, a web bookmarking system [32]. Their results have been confirmed on measurements on the online photo album *Flickr* in [67]. These works have also investigated the incentives for users to collaborate in a social tagging system, and although users mostly tag their items for their personal use, these tags can still be a great contribution to social exploratory search. Halpin et al studied the dynamics of the collaborative tagging system, showing that tagging distributions tend to stabilize into power law distributions [33]. We think that providing a tagging suggestion that carefully examining both the personal behavior and other users' behavior could accelerate the stabilization process, leading to a coherence categorization

scheme in collaborative tagging systems (the task 1 in our paper).

In order to improve social navigation results in collaborative tagging systems, researchers tried to detect semantically related tag clusters [4]. We believe that a personalized tag cloud (the task 2) can contribute more to the user experience than a tag clustering, although a clustering system can easily be integrated into our computation of personalized tag clouds.

So far, academic research into the retrieval and prediction models for collaborative tagging systems has been limited. Collaborative filtering provides methods to predicts a user's interests by looking at other but similar users (user-based collaborative filtering, e.g., [9, 37, 129]) or other but similar items to the target item (e.g., item-based collaborative filtering [23]). A drawback of these collaborative filtering approaches is that they are usually based on explicit user ratings, that are hard to obtain in practice [15]. Alternatively, user preferences may be inferred from implicit observations of user interactions with the system (for instance, tagging) and consequently building a ranking model for user interests. Examples include the item-based Top-N collaborative filtering approaches [23, 114] and Amazon's operational item-based collaborative filtering [61]. However, almost all collaborative filtering research has ignored the tag structure, relying on user-item interactions only. The predictions made about user preferences are conditional on the full user profile, and therefore independent of the user's task. Without other data or inputs, collaborative filtering cannot accurately model the important aspects of users or items. Although some hidden aspect models have been proposed to compute recommendations [41, 50, 99], the interpretation of the hidden aspects in terms of their meaning remains usually unclear. User-input in the form of tags (the opportunity 3) could however provide an effective channel to infer and learn the aspects of user interests and contents, resulting in more specific and task-focussed recommendations.

4.3 Personalization Models

This section introduces generative probabilistic models to address the three personalization problems in collaborative tagging systems posed in the introduction, using the following notation. Let u be a discrete random variable over the sample space of users $\Phi_U = \{1, \dots, M\}$, let i be a random variable over the sample space of items (content) $\Phi_I = \{1, \dots, K\}$, and let t be a random variable over the sample space of tags $\Phi_T = \{1, \dots, L\}$ (where M is the number of users, K the number of items, and L the number of tags in the collection).

Consider the following two processes that generate tagging data. The first model views a tag as the output of a generative process associated with each

user. A particular user's decision to choose a tag is the result of *choosing* a generative model for that particular user, and then generating the tag using that model (illustrated in Fig. 4.2). More formally, for each user $u \in \Phi_U$, we choose a *tag-generative* model Θ_u^T :

$$\Theta_u^T = (\theta_u^1, \dots, \theta_u^t, \dots, \theta_u^L), \text{ with } \theta_u^t \in [0, 1], \sum_u \theta_u^t = 1, \quad (4.1)$$

where θ_u^t indicates the probability of generating a tag t from the distribution belonging to the generative model of a user u . Later on, we assume a multinomial distribution over the vocabulary of tags, but the model itself does not depend on a specific choice of distribution.

The second model assumes that items are the output of a generative process associated with each tag, Θ_T^I . This process is assumed independent from the user variable, to keep the sparsity in the data manageable. This independence assumption keeps our model relatively simple (less parameters) while still having a close representation of the underlying patterns.

Research on the language modelling approach for text retrieval has identified various methods to estimate the term probabilities in document models, using smoothing to handle the sparsity in the term document matrix [126]. We follow these ideas for the estimation of the probabilities in our generative models. Table 4.1 summarizes the two generative models and their probability estimations. For readability, we leave the detailed description about the probability estimations in Appendix 4.A, and continue to formulate the three personalization problems in terms of conditional probabilities:

1. **Collaborative Indexing:** The tag to assign to an item can be suggested based on the probability of a candidate tag t being used to label a given item i from the user u , i.e., $p(t|i, u)$.
2. **Collaborative Browsing:** When browsing a 'tag cloud', the most relevant tags can be identified using the probability of a candidate tag t given a user u , i.e. $p(t|u)$.
3. **Collaborative Search:** After user u selects tag t , the most relevant items have the highest probability $p(i|t, u)$.

These probabilities provide principled ranking scores for a personalized ordering of tags and items in collaborative tagging systems. Our suggestion models combine the user preferences for items with the observed user actions involving tags (e.g., selecting a tag to explore items, or tagging a particular item). The role of tags distinguishes this approach from the suggestions provided by existing collaborative filtering approaches, where items are ranked based on user

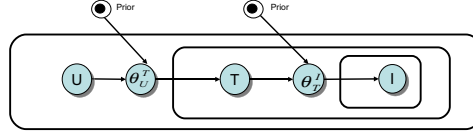


Figure 4.2: A Generative Model of Tagging Data.

Table 4.1: Probability estimation.

| | ML | Laplace smooth. | Bayes smooth. | Jelinek-Mercer Smooth. |
|-------------------------|--------------------------------|--|---|--|
| $p(t \hat{\Theta}_u^T)$ | $\frac{n(u,t)}{\sum_t n(u,t)}$ | $\frac{n(u,t)+\nu}{\sum_t n(u,t)+\nu L}$ | $\frac{n(u,t)+\mu(\sum_u n(u,t)/\sum_{i,t} n(i,t))}{\sum_t n(u,t)+\mu}$ | $\lambda \frac{n(u,t)}{\sum_t n(u,t)} + (1-\lambda) \frac{\sum_u n(u,t)}{\sum_{u,t} n(u,t)}$ |
| | $\alpha_t = 1$ | $\alpha_t = \nu + 1$ | $\alpha_t = \mu \frac{\sum_{i,t} n(i,t)}{\sum_{i,t} n(i,t)} + 1$ | - |
| $p(i \hat{\Theta}_t^I)$ | $\frac{n(i,t)}{\sum_i n(i,t)}$ | $\frac{n(i,t)+\nu}{\sum_i n(i,t)+\nu K}$ | $\frac{n(i,t)+\mu(\sum_t n(i,t)/\sum_{i,t} n(i,t))}{\sum_i n(i,t)+\mu}$ | $\lambda \frac{n(i,t)}{\sum_i n(i,t)} + (1-\lambda) \frac{\sum_t n(i,t)}{\sum_{i,t} n(i,t)}$ |
| | $\alpha_i = 1$ | $\alpha_i = \nu + 1$ | $\alpha_i = \mu \frac{\sum_{i,t} n(i,t)}{\sum_{i,t} n(i,t)} + 1$ | - |

preferences alone, i.e., using $p(i|u)$. Of course, this probability can be derived from our model by marginalizing out the tags, $p(i|u) = \sum_t p(i|t, u)p(t|u)$. In other words, the usage of tags makes the proposed suggestion models more context-aware than traditional collaborative filtering approaches.

4.3.1 Indexing Phase

This section describes personalization for the indexing phase. We aim at suggesting a tag for a given item from a pool of tags that have been employed by other users.

4.3.1.1 Collaborative Indexing Model

Formally, personalizing collaborative indexing refers to the suggestion of candidate tags with high $p(t|i, u)$, for a given user u who labels item i . Using Bayes' rule gives the following ranking formula:

$$p(t|u, i) = \frac{p(t|u)p(i|t, u)}{p(i|u)} \quad (4.2)$$

Assuming that tags generate items independent from u (Fig. 4.2), i.e., $p(i|t, u) = p(i|t)$, and ignore $p(i|u)$ as it is independent from t and therefore does not influence the ranking of tags, we get

$$p(t|u, i) \propto_t \log p(t|u) + \log p(i|t) \quad (4.3)$$

where α_t denotes same rank order with respect to t .

To estimate the two conditional probabilities, we consider the two generative processes that have been discussed (illustrated in Fig. 4.2). In Bayesian inference [29], the generative process can be expressed as an integration over all the model parameters to take the uncertainty about the right model into account. In the case of $p(t|u)$, we have:

$$p(t|u) = \int_{\Theta_u^T} p(t|\Theta_u^T, u) p(\Theta_u^T|u) d\Theta_u^T \quad (4.4)$$

where $p(\Theta_u^T|u)$ (given as $p(\Theta_u^T|\{n(u, t)\}_{t=1}^L, \mathbf{a}_u^T)$ in Appendix 4.A) is the posterior probability of model parameter Θ_u^T when we have observed some tags (denoted as $\{n(u, t)\}_{t=1}^L$) associated with this user u , and $p(t|\Theta_u^T, u)$ describes the generative process from the estimated model to a tag.

In practice, it is common to approximate the full Bayesian integration over the model by estimating the “optimal” model parameters $\hat{\Theta}_u^T$ (e.g. by Maximizing their *A Posterior* probability (MAP)) and then setting $p(\Theta_u^T|u) \approx \delta(\Theta_u^T, \hat{\Theta}_u^T)$ [46]:

$$p(t|u) \approx \int_{\Theta_u^T} p(t|\Theta_u^T, u) \delta(\Theta_u^T, \hat{\Theta}_u^T) d\Theta_u^T = p(t|\hat{\Theta}_u^T) \quad (4.5)$$

This paper takes the approximation approach, leaving the full Bayesian approach for future work. Substitute the optimal model of Eq. 4.5 for each generative process in Eq. 4.3:

$$p(t|u, i) \propto_t \log p(t|\hat{\Theta}_u^T) + \log p(i|\hat{\Theta}_i^I) \quad (4.6)$$

The ranking score of each tagging keyword is composed of two generative processes. The first process calculates how probable the candidate keyword is to be generated from the user model (a completely personal suggestion), while the other one computes from the candidate tag (keyword) model how probable the query item would be generated (a completely popularity-based suggestion). Thus, a tag that has been frequently used in the past by the target user *and* by other users for the target item will have a high ranking score. It naturally copies with the “rich get richer” phenomenon [33].

We can incorporate different types of smoothing techniques into the model estimations in Eq. 4.6 to count data sparsity. (refer to Table 4.1). The smoothing parameter also plays a central role in balancing the personal suggestion with the popularity-based suggestion (see Eq. 4.18). Varying the smoothing parameter μ shows that an optimum can be found between a completely personal and completely popularity-based tag suggestion.

4.3.2 Exploratory Search Phase

This section introduces personalization for the exploratory search phase. Different from the case of personalizing collaborative indexing, we need to predict “new” items or tags, i.e., those that do not exist in the given user preference. Using a model $\hat{\Theta}_u^T$ per user u is suitable for the indexing problem, but inappropriate to make predictions on new tags or items – since we have no observations to compute the user model for the candidate tags (or items). To address this problem, we invert the Bayesian inference, to infer the user’s tags rather than deploy them. In this regard, we represent user preferences explicitly, such that they can be linked to the preferences of other users. Formally, \mathbf{q}_u denotes the preferences of user u , either based on items or on tags. In the former case, user preferences correspond to the set of items that this user has tagged or preferred, i.e., $\mathbf{q}_u = \{i | n(u, i) > 0\}$, where $n(u, i)$ denotes the number of times a user u has tagged an item i (normally these are binary, as users tend to tag an item only once). In the latter case, user preferences are represented by the set of tags that this user has used, $\mathbf{q}_u = \{t | n(u, t) > 0\}$.

4.3.2.1 Collaborative Browsing Model

In collaborative browsing, *interesting* tags are suggested to create a personalized “tag cloud” using the following ranking formula:

$$\begin{aligned} p(t|\mathbf{q}_u) &= \frac{p(\mathbf{q}_u|t)p(t)}{p(\mathbf{q}_u)} \\ &\propto_t \log p(\mathbf{q}_u|t) + \log p(t) - \log p(\mathbf{q}_u) \\ &\propto_t \log p(\mathbf{q}_u|t) + \log p(t) \end{aligned} \quad (4.7)$$

where $\log p(\mathbf{q}_u)$ can be removed since it is independent of the target t . The tag ranking has two parts: its likelihood towards the user preference $p(\mathbf{q}_u|t)$ and its popularity $p(t)$. The probability $p(t)$ can be easily estimated by counting the frequency from the collection.

To estimate the likelihood $p(\mathbf{q}_u|t)$, we follow the argument of the previous section. That is we first choose an optimal tag model $\hat{\Theta}_t^I$ (an item-generation model) for each candidate tag t and then estimate the probability of the user preference (as query) being generated by the candidate tag model:

$$p(t|\mathbf{q}_u) \propto_t \log p(\mathbf{q}_u|\hat{\Theta}_t^I) + \log p(t) \quad (4.8)$$

The estimation of the likelihood $p(\mathbf{q}_u|\hat{\Theta}_t^I)$ depends again on the representation of the user preferences (using items or tags). If we use the representation as a

set of items and assume that each item in the user preference is independently generated, we have

$$p(t|\mathbf{q}_u) \propto_t \sum_{i' \in \mathbf{q}_u} \log p(i'|\hat{\Theta}_t^I) + \log p(t) \quad (4.9)$$

Taking the alternative representation of user preferences by a set of tags and assuming that each tag in the user preference is independently generated result in the following ranking score from Eq. 4.7:

$$\begin{aligned} & p(t|\mathbf{q}_u) \\ & \propto_t \log p(\mathbf{q}_u|\hat{\Theta}_t^I) + \log p(t) \\ & = \sum_{t' \in \mathbf{q}_u} n(t', u) \log p(t'|\hat{\Theta}_t^I) + \log p(t) \\ & = \sum_{t' \in \mathbf{q}_u} n(t', u) \log \left(\sum_{i'} p_{\text{ML}}(t'|i') p(i'|\hat{\Theta}_t^I) \right) + \log p(t) \end{aligned} \quad (4.10)$$

where $p_{\text{ML}}(t|i) = \frac{n(i,t)}{\sum_t n(i,t)}$. The ranking corresponds to the sum (in logarithm domain) of a personalized suggestion and the popularity suggestion. When we know little about the user, we have less observations on the generation from the target to the user preference and thus the prediction comes mainly from the popularity part. The smoothing parameters balance the two suggestions. For instance, in the Jelinek-Mercer smoothing, when the λ is zero, the first term becomes constant for all the candidate tags and the prediction relies solely on popularity.

4.3.2.2 Collaborative Item Search Model

Following the discussion in section 4.3.2.1 and considering the tag-generation model of an item, we can derive the relevant item ranking model similarly. We omit the derivations and give the final ranking score as follows:

$$\begin{aligned} p(i|\mathbf{q}_u, t) & \propto_i \log p(\mathbf{q}_u|\hat{\Theta}_i^T) + \log p(t|\hat{\Theta}_i^T) + \log p(i) \\ & \propto_i \left(\sum_{t' \in \mathbf{q}_u} n(t', u) \log p(t'|\hat{\Theta}_i^T) \right) \\ & \quad + \log p(t|\hat{\Theta}_i^T) + \log p(i) \end{aligned} \quad (4.11)$$

where the user preference is represented by a set of tags (the model using the item-based user preference can be obtained similarly). We can see that the ranking for a given item is a combination of the its popularity ($p(i)$), its probability of generating the query tag ($p(t|\hat{\Theta}_i^T)$), and its probability of generating the user preference ($p(\mathbf{q}_u|\hat{\Theta}_i^T)$).

Table 4.2: Characteristics of the test data sets.

| | del.icio.us | CiteULike |
|-------------------------------|-------------|-----------|
| Num. of Users | 1731 | 741 |
| Num. of Items | 3370 | 2179 |
| Num. of Tags | 1097 | 960 |
| Num. of User-Item-Tag Triples | 772087 | 20703 |
| Avg. Num. of Tags per User | 109 | 12 |
| Avg. Num. of Items per Tag | 135 | 14 |
| Avg. Num. of Tags per Item | 44 | 6 |

4.3.3 Discussions

Our approach to modelling tagging data resembles the generative models depicted in language modelling of text retrieval [55, 56], with the difference that tagging data has two different generative processes. For each specific task and its model, their comparison with other common topics in information retrieval is summarized as follows:

1. Collaborative Indexing: This indexing task is similar to document categorization or classification, as both aim at labeling (classifying) the documents (items). However, the indexing in our model is “collaborative”. The final ranking balances the popularity ranking against the personalized one. Therefore, a good tag candidate should be consistent with other users’ labeling behavior while still focusing on the user’s own vocabulary (familiar tags in the user preference).

2. Collaborative Browsing: This model for ranking tags looks somewhat similar to the case of query expansion using relevance feedback in text retrieval, where terms are ranked against a set of judged documents from a given user. Yet, the two problems are quite different. In our tag ranking, we want to suggest tags that are not used previously by this user. Thus, we have to use the information from other users such that the terms (tags), which are not judged in the relevant document list (user preferences), can still be suggested. We achieve this by looking at how similar the tag is to the items that the target user preferred (item-based user preference) or to the tags that the target user used (tag-based user preference).

3. Collaborative Item Search: If we treat tags as one-word queries, our model intuitively provides a unified framework to combine the language modelling of ranking queries to the collaborative filtering of ranking user preferences. This is similar to the work of collaborative web search in [102] with the difference that our model gives a more theoretical foundation.

Table 4.3: Comparison of the Collaborative Indexing (Tagging) Model with non-personalized ones. A Wilcoxon signed-rank test is conducted and the significant ones (P-value < 0.05) are marked as *.

| Precision: | | | | | | |
|-------------------------|---------------|---------------|---------------|---------------|---------------|---------------|
| User Prof. Length: | 20% | | | 40% | | |
| Top-N Returned: | 1 | 3 | 5 | 1 | 3 | 5 |
| <i>Tagging-BS</i> | 0.832* | 0.733* | 0.641* | 0.852* | 0.749* | 0.656* |
| <i>Non-Personalized</i> | 0.799 | 0.684 | 0.613 | 0.801 | 0.692 | 0.615 |
| User Prof. Length: | 60% | | | 80% | | |
| Top-N Returned: | 1 | 3 | 5 | 1 | 3 | 5 |
| <i>Tagging-BS</i> | 0.867* | 0.768* | 0.673* | 0.866* | 0.780* | 0.683* |
| <i>Non-Personalized</i> | 0.807 | 0.702 | 0.626 | 0.806 | 0.698 | 0.631 |

| Recall: | | | | | | |
|-------------------------|---------------|---------------|---------------|---------------|---------------|---------------|
| User Prof. Length: | 20% | | | 40% | | |
| Top-N Returned: | 1 | 3 | 5 | 1 | 3 | 5 |
| <i>Tagging-BS</i> | 0.135 | 0.353* | 0.511* | 0.137* | 0.358* | 0.517* |
| <i>Non-Personalized</i> | 0.130 | 0.329 | 0.487 | 0.128 | 0.329 | 0.482 |
| User Prof. Length: | 60% | | | 80% | | |
| Top-N Returned: | 1 | 3 | 5 | 1 | 3 | 5 |
| <i>Tagging-BS</i> | 0.138* | 0.361* | 0.522* | 0.135* | 0.360* | 0.520* |
| <i>Non-Personalized</i> | 0.128 | 0.329 | 0.484 | 0.126 | 0.322 | 0.480 |

(a) in the del.icio.us Data Set.

| Precision: | | | | | | |
|-------------------------|---------------|---------------|---------------|---------------|---------------|---------------|
| User Prof. Length: | 20% | | | 40% | | |
| Top-N Returned: | 1 | 3 | 5 | 1 | 3 | 5 |
| <i>Tagging-BS</i> | 0.615* | 0.432* | 0.313* | 0.661* | 0.452* | 0.322* |
| <i>Non-Personalized</i> | 0.501 | 0.382 | 0.292 | 0.515 | 0.393 | 0.298 |
| User Prof. Length: | 40% | | | 80% | | |
| Top-N Returned: | 1 | 3 | 5 | 1 | 3 | 5 |
| <i>Tagging-BS</i> | 0.671* | 0.479* | 0.336* | 0.662* | 0.462* | 0.324* |
| <i>Non-Personalized</i> | 0.522 | 0.405 | 0.306 | 0.503 | 0.373 | 0.288 |

| Recall: | | | | | | |
|-------------------------|---------------|---------------|---------------|---------------|---------------|---------------|
| User Prof. Length: | 20% | | | 40% | | |
| Top-N Returned: | 1 | 3 | 5 | 1 | 3 | 5 |
| <i>Tagging-BS</i> | 0.178* | 0.369* | 0.439* | 0.190* | 0.384* | 0.451* |
| <i>Non-Personalized</i> | 0.144 | 0.325 | 0.413 | 0.147 | 0.333 | 0.419 |
| User Prof. Length: | 60% | | | 80% | | |
| Top-N Returned: | 1 | 3 | 5 | 1 | 3 | 5 |
| <i>Tagging-BS</i> | 0.191* | 0.401* | 0.462* | 0.190* | 0.391* | 0.451* |
| <i>Non-Personalized</i> | 0.147 | 0.337 | 0.424 | 0.142 | 0.314 | 0.404 |

(b) in the CiteULike Data Set.

4.4 Experiments

4.4.1 Data Set Preparation

Despite the popularity of collaborative tagging, there are no standard data sets available for research evaluation. We collected data from two well-known col-

Table 4.4: Comparison of the Collaborative Item Search Model with non-personalized ones. A Wilcoxon signed-rank test is conducted and the significant ones (P-value < 0.05) are marked as *. Precision is measured.

| User Prof. Length: | 20% | | | 40% | | |
|-------------------------|---------------|---------------|---------------|---------------|---------------|---------------|
| Top-N Returned: | 1 | 5 | 10 | 1 | 5 | 10 |
| <i>ItemUP-BS</i> | 0.284* | 0.192* | 0.160* | 0.280* | 0.191* | 0.154* |
| <i>Non-Personalized</i> | 0.263 | 0.182 | 0.150 | 0.257 | 0.171 | 0.139 |
| User Prof. Length: | 60% | | | 80% | | |
| Top-N Returned: | 1 | 5 | 10 | 1 | 5 | 10 |
| <i>ItemUP-BS</i> | 0.249* | 0.175* | 0.139* | 0.240* | 0.138* | 0.113* |
| <i>Non-Personalized</i> | 0.228 | 0.144 | 0.119 | 0.186 | 0.112 | 0.094 |

(a) in the del.icio.us Data Set.

| User Prof. Length: | 20% | | | 40% | | |
|-------------------------|---------------|---------------|---------------|---------------|---------------|---------------|
| Top-N Returned: | 1 | 5 | 10 | 1 | 5 | 10 |
| <i>Tag UP-BS</i> | 0.190* | 0.094* | 0.063* | 0.193* | 0.095* | 0.065* |
| <i>Non-Personalized</i> | 0.146 | 0.077 | 0.055 | 0.141 | 0.075 | 0.057 |
| User Prof. Length: | 60% | | | 80% | | |
| Top-N Returned: | 1 | 5 | 10 | 1 | 5 | 10 |
| <i>Tag UP-BS</i> | 0.174* | 0.081* | 0.052* | 0.183* | 0.078* | 0.048* |
| <i>Non-Personalized</i> | 0.118 | 0.062 | 0.044 | 0.111 | 0.052 | 0.040 |

(b) in the CiteULike Data Set.

laborative tagging web sites, del.icio.us and CiteULike. The corpus has been crawled between May and October 2006. We collected a number of the most popular tags, found which users were using these tags, and then downloaded the whole profiles of these users, and applied standard term tokenization techniques from text retrieval followed by stopword removal. Finally, we extracted the user-item-tag triples from each of the user profiles. User IDs are randomly generated to keep the users anonymous. Table 4.2 summarizes the basic characteristics of the data sets.

4.4.2 Evaluation Protocols

Evaluation Methodology: Since the three tasks, either ranking items or tags, are essentially prediction tasks, we can evaluate their performance by holding a certain amount of data as ground-truth and building the prediction models from the remaining data to predict the ground-truth. The effectiveness of these predictions can be therefore evaluated by comparing the predicted ones with the ground-truth. For this, we randomly divided the data set into a training set (80% of the users) and a test set (20% of the users). The training set was used to estimate the model. The test set was used for evaluating the accuracy of the suggestions on the new users, whose user profiles (represented by items or tags) are not in the training set. For each test user, part of the items and their associated tags of that test user, for instance, 20% and 40% etc., was put into

Table 4.5: Comparison of the Collaborative Browsing Model with other alternatives in the *del.icio.us* Data Set. A Wilcoxon signed-rank test is conducted and the significant ones (P-value < 0.05) over the second best are marked as *. Precision is measured.

| User Prof.: | 20% | | | 40% | | |
|--------------------|---------------|---------------|---------------|---------------|---------------|---------------|
| Top-N: | 1 | 5 | 10 | 1 | 5 | 10 |
| <i>Item UP-BS</i> | 0.848* | 0.795* | 0.743* | 0.835* | 0.753* | 0.688* |
| <i>Non-Person.</i> | 0.746 | 0.746 | 0.690 | 0.705 | 0.690 | 0.623 |
| User Prof.: | 60% | | | 80% | | |
| Top-N: | 1 | 5 | 10 | 1 | 5 | 10 |
| <i>Item UP-BS</i> | 0.776* | 0.666* | 0.588* | 0.645* | 0.495* | 0.404* |
| <i>Non-Person.</i> | 0.631 | 0.591 | 0.507 | 0.504 | 0.413 | 0.328 |

(a) Comparison with the popularity-based one.

| User Prof.: | 20% | | | 40% | | |
|-------------------|---------------|--------------|---------------|---------------|---------------|---------------|
| Top-N: | 1 | 5 | 10 | 1 | 5 | 10 |
| <i>Item UP-BS</i> | 0.849 | 0.795 | 0.744 | 0.836* | 0.754* | 0.688 |
| <i>ItemProb</i> | 0.860 | 0.786 | 0.741 | 0.803 | 0.729 | 0.672 |
| <i>ItemCos</i> | 0.833 | 0.799 | 0.752 | 0.815 | 0.740 | 0.683 |
| <i>UserCos</i> | 0.822 | 0.779 | 0.728 | 0.793 | 0.732 | 0.674 |
| User Prof.: | 60% | | | 80% | | |
| Top-N: | 1 | 5 | 10 | 1 | 5 | 10 |
| <i>Item UP-BS</i> | 0.776* | 0.667 | 0.588* | 0.645* | 0.495* | 0.404* |
| <i>ItemProb</i> | 0.738 | 0.636 | 0.561 | 0.580 | 0.455 | 0.379 |
| <i>ItemCos</i> | 0.748 | 0.656 | 0.576 | 0.597 | 0.470 | 0.385 |
| <i>UserCos</i> | 0.733 | 0.647 | 0.571 | 0.583 | 0.471 | 0.386 |

(b) Comparison with the ranking-based collaborative filtering.

the user profile list. The remaining items and their associated tags were used to test the prediction (suggestion) performance. By doing so, we model different sparsity of user profiles. The overall scheme is illustrated in Fig. 4.3. For cross-validation, results are averaged over 5 different runs (sampling of training/test set).

Evaluation Metrics: We have three types of suggestion models. To evaluate their effectiveness, we choose different evaluation metrics with respect to their purposes. For the collaborative browsing ($p(t|u)$) and search models ($p(i|u, t)$), their effectiveness can be measured using the *precision* since they are used to help a user to find relevant tags (or items). It basically measures the proportion of suggested tags (the collaborative browsing model) or items (the collaborative search model) that are ground truth tags or items (only partially known).

For the collaborative tagging model ($p(t|u, i)$), users need to get some specific keywords to label the given item. Thus *recall* is also an appropriate measurement. It measures the proportion of the ground truth tags that are indeed suggested. For this model, we use both precision and recall.

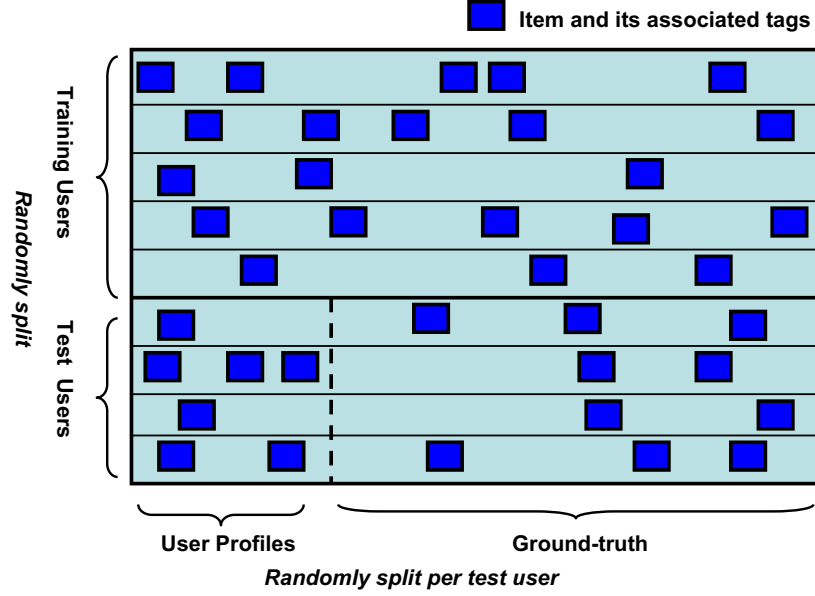


Figure 4.3: The split of Data Sets.

4.4.3 Performance of Personalization Models

This section evaluates the performance of the proposed personalization models. The first experiments investigate the performance of the collaborative indexing models and the collaborative item search. The main advantage of these two models is to integrate “collaborative” user behavior into the ranking scores. To evaluate the effectiveness of these models, we compare the personalized results to those obtained with a non-personalized ranking (i.e., applying the standard language modelling approach for text retrieval [39]: a generative model from candidate item to the query tag or vice versa). The results of collaborative indexing search are given in Table 4.3 and those for collaborative item search in Tables 4.4. Both of them use the Bayes smoothing (see Table 4.1) and the parameters are chosen to be optimal in a validate set. A Wilcoxon signed-rank test [45] is done to verify the significance. These results indicate that the personalized collaborative models significantly outperform the non-personalized approach, regardless of the sparsity of user preferences and the different data sets investigated. More specifically, comparing the different user profile length, we notice that the more observations about user preferences we have, the more our models improve over the non-personalized performance.

Next, let us look at collaborative browsing task. Table 4.5 (a) shows that

our model with Bayes smoothing (denoted as *ItemUP-BS*) also significantly outperform the popularity-based ranking in all configurations. Recall that collaborative browsing aims at ranking (suggesting) tags with respect to a certain user profile (user interest). For this task, some existing collaborative filtering techniques can be applied if we treat tags as items. Collaborative filtering techniques can be roughly classified as rating prediction and item ranking. Rating-prediction-based collaborative filtering is designed for rating data and its purpose is to predict user ratings of items. This makes it inapplicable to implicit ratings like click data (or, in our case, tagging data). But for item-ranking-based collaborative filtering, its purpose is to rank items and thus it can be employed for the tag suggestion task. Therefore we choose one of the state-of-the-art ranking-based collaborative filtering techniques: the item-based top-N recommendation method [23]. To make a fair comparison, we directly use their optimized *Top-N-suggest* recommendation engine¹ [52] and compared them with the results of the our collaborative browsing model. Particularly, we compared our own results to the item-based TF×IDF-like version (denoted as ItemProb) and to the user-based cosine similarity method [37] (denoted as UserCos), setting the parameters to the optimal ones according to the user manual. We report cosine similarity results for item-based approaches [92] as well (denoted as ItemCos). Results in Table 4.5 (b) show that, in most cases, our model outperforms the ranking-based collaborative filtering approaches. Particularly, the improvement becomes significant when we have more data in user profiles. This is because we model two generative processes in tagging data while common collaborative filtering techniques only model one generative process, in this case, the user-to-tag generative process.

4.4.4 Representation of User Profiles

As explained in Section 3, user profiles can be represented by a set of items or tags. This section compares the effectiveness of the two representations (e.g., *ItemUP*, *TagUP*).

Starting with the collaborative browsing model, we report results for user preference lengths 5, 10, and 20, and the corresponding precisions at top-1, top-5 and the top-10. Table 4.6 (a) shows that, in general, the item-based user preference representation outperforms the tag-based representation. This may due to the fact that in a tagging system like *del.icio.us*, a user only considers a new tag when the old tags are insufficiently expressive for the newly added items. As a consequence, the correlation between two tags in one user profile is less obvious than that between two items. Thus, using the “old” tags to predict (rank) new tags is not as reliable as using the “old” items.

¹<http://www-users.cs.umn.edu/~karypis/suggest/>

Table 4.6: Comparison of the two representation methods for user profiles. Precision is measured.

| User Prof.: | 20% | | | 40% | | |
|--------------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Top-N: | 1 | 5 | 10 | 1 | 5 | 10 |
| <i>Item UP-BS</i> | 0.848 | 0.795 | 0.743 | 0.835 | 0.753 | 0.688 |
| <i>Item UP-JMS</i> | 0.772 | 0.764 | 0.714 | 0.763 | 0.723 | 0.664 |
| <i>Tag UP-BS</i> | 0.810 | 0.764 | 0.708 | 0.797 | 0.719 | 0.646 |
| <i>Tag UP-JMS</i> | 0.753 | 0.752 | 0.692 | 0.716 | 0.695 | 0.625 |
| User Prof.: | 60% | | | 80% | | |
| Top-N: | 1 | 5 | 10 | 1 | 5 | 10 |
| <i>Item UP-BS</i> | 0.776 | 0.666 | 0.588 | 0.645 | 0.495 | 0.404 |
| <i>Item UP-JMS</i> | 0.709 | 0.632 | 0.564 | 0.595 | 0.473 | 0.392 |
| <i>Tag UP-BS</i> | 0.715 | 0.623 | 0.542 | 0.578 | 0.447 | 0.358 |
| <i>Tag UP-JMS</i> | 0.656 | 0.595 | 0.515 | 0.530 | 0.422 | 0.345 |

(a) the Collaborative Browsing Model in the `del.icio.us` Data Set.

| User Prof. Length: | 20% | | | 40% | | |
|--------------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Top-N Returned: | 1 | 5 | 10 | 1 | 5 | 10 |
| <i>ItemUP-BS</i> | 0.284 | 0.192 | 0.160 | 0.280 | 0.191 | 0.154 |
| <i>TagUP-BS</i> | 0.282 | 0.194 | 0.161 | 0.274 | 0.187 | 0.149 |
| User Prof. Length: | 60% | | | 80% | | |
| Top-N Returned: | 1 | 5 | 10 | 1 | 5 | 10 |
| <i>ItemUP-BS</i> | 0.249 | 0.175 | 0.139 | 0.240 | 0.138 | 0.113 |
| <i>TagUP-BS</i> | 0.248 | 0.164 | 0.132 | 0.213 | 0.132 | 0.107 |

(b) the Collaborative Item Search Model in the `del.icio.us` Data Set.

| User Prof. Length: | 20% | | | 40% | | |
|--------------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Top-N Returned: | 1 | 5 | 10 | 1 | 5 | 10 |
| <i>Item UP-BS</i> | 0.153 | 0.079 | 0.055 | 0.173 | 0.080 | 0.054 |
| <i>Tag UP-BS</i> | 0.190 | 0.094 | 0.063 | 0.193 | 0.095 | 0.065 |
| User Prof. Length: | 60% | | | 80% | | |
| Top-N Returned: | 1 | 5 | 10 | 1 | 5 | 10 |
| <i>Item UP-BS</i> | 0.134 | 0.065 | 0.043 | 0.140 | 0.062 | 0.040 |
| <i>Tag UP-BS</i> | 0.174 | 0.081 | 0.052 | 0.183 | 0.078 | 0.048 |

(c) the Collaborative Item Search Model in the `CiteULike` Data Set.

But for the collaborative item search model, the two user profile representations may perform differently on different data sets. Notice from Table 4.6 (b) and (c) that the tag-based user preference representation (i.e. TagUP-BS), in general, outperforms the item-based one in the `CiteULike` data set, but it behaves differently in `del.icio.us`. This is because the tags for scientific papers are more specific than those for URLs. As a consequence, tags in `CiteULike` can represent users' interests more precisely than in `del.icio.us`.

Besides, we also observe from table 4.6 (a) that the Bayes smoothing works better than the Jelinek-Mercer smoothing, regardless of the different representation of user profiles. The possible explanation in our case is that, like document length in text retrieval, our candidate tags have different number of items associated with them. Bayes smoothing adapts to the "tag length" (see

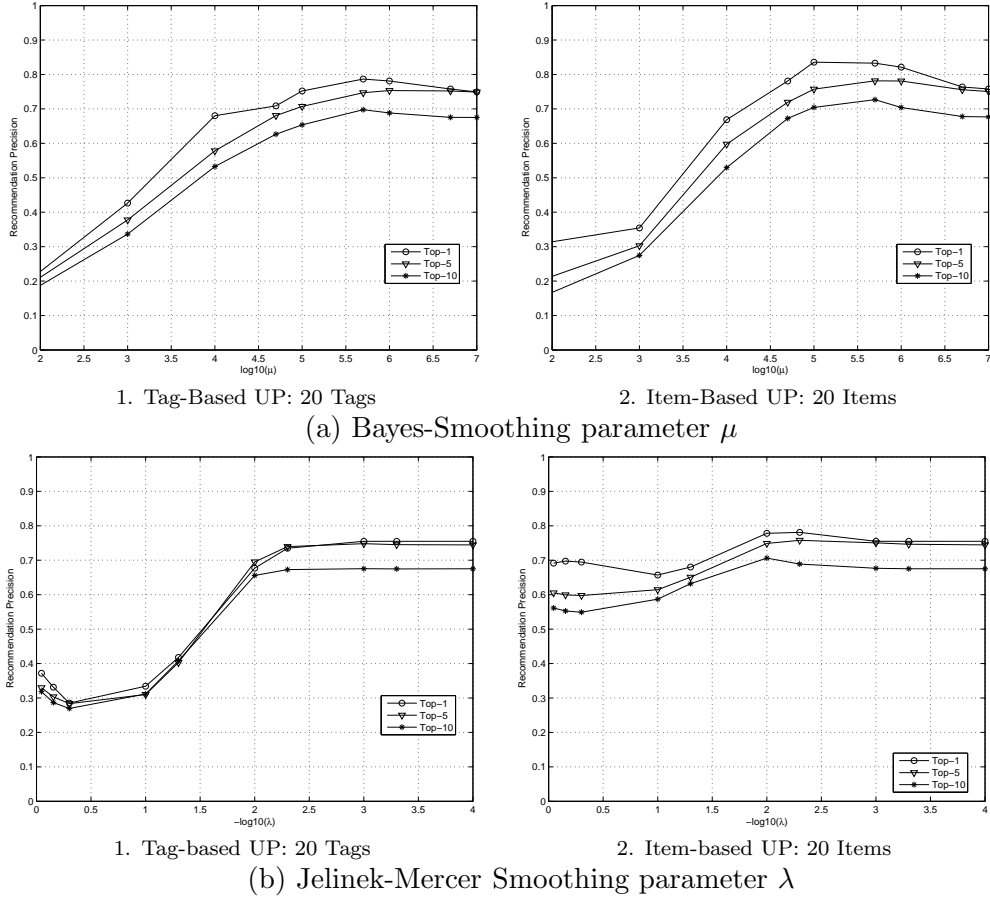


Figure 4.4: Impact of the parameters in the Collaborative Browsing Model. In the `del.icio.us` Data Set.

explanation in Eq. 4.19) and thus leads to a better performance.

4.4.5 Impact of Parameters

This section evaluates the smoothing parameters in our models in the `del.icio.us` data set. Our objective is to study the sensitivity and impact of the parameters, and, for a thorough study of the different smoothing techniques, we refer to [126].

The first experiments address the collaborative exploratory search phase. Fig. 4.4 (a) and (b) plot precision against parameter μ in Bayes smoothing (BS) and parameter λ of Jelinek-Mercer smoothing (JMS), respectively, using a logarithmic scale. Notice that we plot $-\log_{10}(\lambda)$ such that in both Figures the

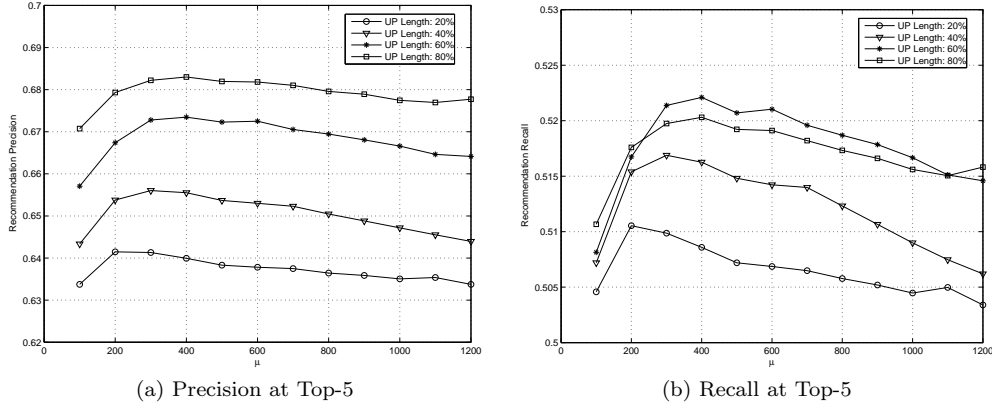


Figure 4.5: Impact of the parameter μ in the Collaborative Indexing Model. In the `del.icio.us` Data Set.

amount of smoothing increases along the axis. We observe from both figures that the optimal precision emphasizes either a higher value of μ or a low value of λ , indicating that parameter estimation needs a large amount of smoothing from the background collection model (a popularity-based model). The large amount of smoothing is attributed to data sparsity. The optimal results of Bayes smoothing are relatively stable in a wide range between 10^5 to 10^6 and those of Jelinek-Mercer smoothing are in the range of 10^{-4} and 10^{-2} , independent of representation and length of user preferences. In addition, we observe that the precision obtained with Jelinek-Mercer smoothing is more sensitive to λ for the tag-based user preferences than for the item-based user preferences. The tag-based user preferences need even more smoothing (almost corresponding to coordination level matching [126]). Additional experiments (not reported here) show that smoothing in collaborative item search exhibits similar behavior.

Regarding collaborative indexing, recall how Eq. 4.6 is based on two generative models, the user's tag-generation model and tag's item-generation model. Since the goal is to create a vocabulary shared by all users, we do not want to suggest tags that no other user assigned to the item, so we choose the maximum-likelihood estimator for the tag's item-generation model. Bayes smoothing is applied in the user's tag-generation model. Fig. 4.5 plots the precision and recall against parameter μ , showing that the optimal μ is insensitive to the length of user profiles, and has a relatively small value compared to the one in the collaborative browsing model. This means that the indexing model needs less smoothing from the background collection model, which can be explained because users tend to select tags from their own user preference list to label an item.

4.5 Conclusions

User-generated content and meta data open new avenues for many interesting applications and challenges. This paper investigated three ways to personalize collaborative tagging systems: during indexing (assigning tags to items), browsing (finding interesting tags to explore) and search (finding items representative of an interesting tag). We conclude from the experimental results that the proposed personalized methods are effective, and could improve the user's indexing and retrieval experiences in collaborative tagging systems.

In the future, like other personalized tasks [14, 24, 98], we will also need do a user study to further evaluate our models in operational tagging systems. In such an environment, a setup that could incorporate relevance feedback is of particular interest. In addition, we plan to extend the estimation process to a full Bayesian treatment (Eq. 4.4 and [125]).

4.A Probability Estimation

For simplicity, we only describe the parameter estimation for the user's tag-generation model, which is used for the collaborative tagging model. The probability estimations of other generative models, used for other two models, can be derived analogously.

As we described in Section 4.3, the user's tag-generation model views a tag as the output of a generative process associated with each user. For a given user u , we treat the parameters of the tag-generation model Θ_u^T , defined in Eq. 4.1, as random variables (Fig. 4.2) and estimate their value by maximizing their *a posteriori* probability [46]:

$$\hat{\Theta}_u^T = \arg \max_{\Theta_u^T} p(\Theta_u^T | \{n(u, t)\}_{t=1}^L, \mathbf{a}_u^T) \quad (4.12)$$

where $n(u, t)$ denotes the observation of the number of times that a tag t has been used by the user u and \mathbf{a}_u^T denotes the parameters of the prior distribution (often referred to as *hyper-parameters*). The posterior probability is proportional to the product of the likelihood and the prior probability:

$$p(\Theta_u^T | \{n(u, t)\}_{t=1}^L, \mathbf{a}_u^T) \propto p(\{n(u, t)\}_{t=1}^L | \Theta_u^T) p(\Theta_u^T | \mathbf{a}_u^T) \quad (4.13)$$

The likelihood $p(\{n(u, t)\}_{t=1}^L | \Theta_u^T) \propto \prod_t (\theta_u^t)^{n(u, t)}$ captures the knowledge of the model parameters coming from the observed data $(\{n(u, t)\}_{t=1}^L)$. Data sparsity results in only small amounts of data to achieve an "accurate" estimation of these parameters. A solution is to deploy the prior $p(\Theta_u^T | \mathbf{a}_u^T)$ to incorporate prior knowledge of the model parameters. In practice, the multinomial's conjugate distribution (the Dirichlet) is chosen as prior to simplify estimation [29]:

$$p(\Theta_u^T | \mathbf{a}_u^T) \propto \prod_t (\theta_u^t)^{a_t - 1} \quad (4.14)$$

where $\mathbf{a}_u^T = (a_1, \dots, a_L)$ are the parameters of the Dirichlet distribution. Because of using the conjugate, the posterior probability after observing some data corresponds to a Dirichlet with updated parameters:

$$\begin{aligned} p(\Theta_u^T | \{n(u, t)\}_{t=1}^L, \mathbf{a}_u^T) &\propto \prod_t (\theta_u^t)^{n(u, t)} \prod_t (\theta_u^t)^{a_t - 1} \\ &= \prod_t (\theta_u^t)^{n(u, t) + a_t - 1} \end{aligned} \quad (4.15)$$

Maximizing the posterior probability in Eq. 4.15 (taking the mode [29]) gives the estimation of the probabilities in the tag-generation model.

$$p(t | \hat{\Theta}_u^T) = \hat{\theta}_u^t = \frac{n(u, t) + a_t - 1}{(\sum_t n(u, t) + (\sum_t a_t) - L)} \quad (4.16)$$

Varying choices for hyper-parameter a_t lead to different estimators [125]. For instance, a constant value $a_t = 1$ gives the *maximum-likelihood* estimator. Setting $a_t = \nu + 1$, where ν is a free parameter, results in the generalized *Laplace smoothing* estimator. Alternatively, the prior can be fit on the distribution of the tags in a given collection:

$$a_t = \mu \cdot p_{\text{ML}}(t) + 1, \text{ where } p_{\text{ML}}(t) = \frac{\sum_u n(u, t)}{\sum_{u, t} n(u, t)} \quad (4.17)$$

where p_{ML} is the maximum-likelihood estimator. Substituting Eq. 4.17 into Eq. 4.16 results in the *Bayes-smoothing* estimator [125]

$$p(t|\hat{\Theta}_u^T) = \hat{\theta}_u^t = \frac{n(u, t) + \mu \cdot p_{\text{ML}}(t)}{(\sum_t n(u, t)) + \mu} \quad (4.18)$$

Eq. 4.18 is equivalent to (details in [126])

$$p(t|\hat{\Theta}_u^T) = \hat{\theta}_u^t = \lambda_u p_{\text{ML}}(t|u) + (1 - \lambda_u) p_{\text{ML}}(t), \quad (4.19)$$

where

$$\lambda_u = \left(\frac{\sum_t n(u, t)}{\mu + \sum_t n(u, t)} \right), \quad p_{\text{ML}}(t|u) = \frac{n(u, t)}{\sum_t n(u, t)} \quad (4.20)$$

The result can be viewed as an adaptive version of linear interpolation smoothing with $p(t)$, the term probability estimated from a background model. One can also fix the influence from the background model with a constant $\lambda_u = \lambda$, giving the commonly used *Jelinek-Mercer* smoothing [126].

Commentary on Chapter 4

A drawback of *pure* collaborative filtering algorithms (including those introduced in Chapters 1 and 2) is that the relevance prediction of a user information need (interest) relies entirely on the profile of the user, regardless of the user's task at hand. As a result, the recommendations are less task-focused. This is why there are as yet no available generic recommendation engines: current recommendation systems are designed specifically for one particular type of information item: for example, the recommender system of *Netflix* (netflix.com) handles only DVD movies, while that of *Last.FM* (Last.FM) deals solely with music items. Chapter 4 suggests that, in recommender systems, asking users to provide queries could be effective in inferring and learning the aspects of user interests, resulting in more specific and task-focused recommendations.

On the other hand, to address common search tasks such as “find a favorite restaurant to have dinner tonight, a blog to read, or the best review about LCD TV”, we need an information retrieval system not only to handle one-time, short-term queries (e.g. information about “restaurant” or “blog” or “LCD TV”) but also, importantly, to make suggestions on the basis of repetitive, long-term

user preferences. Thus, it would be of great interest in the future to break the barriers between information retrieval and (social) information filtering and to create a new retrieval paradigm that automatically determines the relevance of

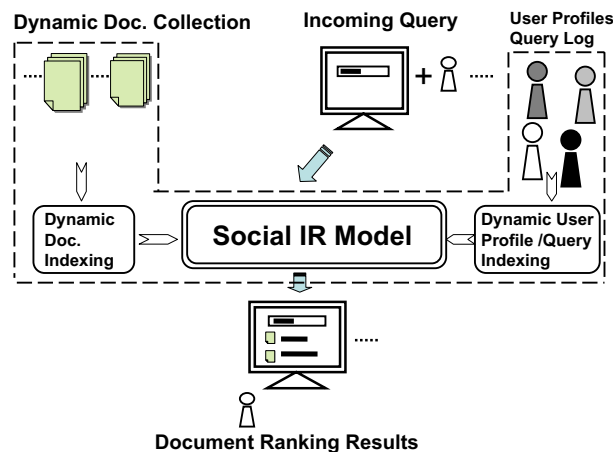


Figure 4.6: A Social Information Retrieval System.

information items by both considering the interactions or contributions of users (**social-based**) and analyzing the content of information items and queries (**content-based**). Our vision is illustrated in Fig. 4.6. We not only deal with the document and query collections but, importantly, also consider information retrieval as a social activity, including other users' knowledge and experience in the framework.

Part II

Unified Models

Chapter 5

Similarity Fusion

Memory-based methods for collaborative filtering predict new ratings by averaging (weighted) ratings between, respectively, pairs of similar users *or* items. In practice, a large number of ratings from similar users or similar items are not available, due to the sparsity inherent to rating data. Consequently, prediction quality can be poor. This paper reformulates the memory-based collaborative filtering problem in a generative probabilistic framework, treating individual user-item ratings as predictors of missing ratings. The final rating is estimated by fusing predictions from three sources: predictions based on ratings of the same item by other users, predictions based on different item ratings made by the same user, and, third, ratings predicted based on data from other but similar users rating other but similar items. Existing user-based and item-based approaches correspond to the two simple cases of our framework. The complete model is however more robust to data sparsity, because the different types of ratings are used in concert, while additional ratings from similar users towards similar items are employed as a background model to smooth the predictions. Experiments demonstrate that the proposed methods are indeed more robust against data sparsity and give better recommendations.

This work has been published as “Unifying user-based and item-based collaborative filtering approaches by similarity fusion”, by J. Wang, A. P. de Vries, and M. J. T. Reinders, in SIGIR06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pages 501–508, New York, NY, 2006. So also [116].

5.1 Introduction

Collaborative filtering aims at predicting the user interest for a given item based on a collection of user profiles. Commonly, these profiles either result from asking users explicitly to rate items or are inferred from log-archives ([41]). Research started with memory-based approaches to collaborative filtering, that can be divided in user-based approaches like [9, 37, 48, 81] and item-based approaches like [23, 92]. The former approaches form a heuristic implementation of the “Word of Mouth” phenomenon. Memory-based approaches are widely used in practice, e.g., [37, 61].

Given an unknown test rating (of a test item by a test user) to be estimated, memory-based collaborative filtering first measures similarities between test user and other users (user-based), or, between test item and other items (item-based). Then, the unknown rating is predicted by averaging the (weighted) known ratings of the test item by similar users (user-based), or the (weighted) known ratings of similar items by the test user (item-based).

In both cases, only partial information from the data embedded in the user-item matrix is employed to predict unknown ratings (using either correlation between user data or correlation between item data). Because of the sparsity of user profile data however, many related ratings will not be available for the prediction. Therefore, it seems intuitively desirable to fuse the ratings from both similar users and similar items, to reduce the dependency on often missing data. Also, methods known previously ignore the information that can be obtained from ratings made by other but similar users to the test user on other but similar items. Not using such ratings causes the *data sparsity problem* of memory-based approaches to collaborative filtering: for many users and items, no reliable recommendation can be made because of a lack of similar ratings.

This paper sets up a generative probabilistic framework to exploit more of the data available in the *user-item matrix*, by fusing all ratings with predictive value for a recommendation to be made. Each individual rating in the user-item matrix is treated as a separate prediction for the unknown test rating (of a test item from a test user). The confidence of each individual prediction can be estimated by considering both its similarity towards the test user and that towards the test item. The overall prediction is made by averaging the individual ratings weighted by their confidence. The more similar a rating towards the test rating, the higher the weight assigned to that rating to make the prediction. Under this framework, the item-based and user-based approaches are two special cases, and these can be systematically combined. By doing this, our approach allows us to take advantage of user correlations *and* item correlations embedded in the user-item matrix. Besides, smoothing from a background model (estimated from known ratings of similar items by

similar users) is naturally integrated into our framework to improve probability estimation and counter the problem of data sparsity.

The remainder of the paper is organized as follows. We first summarize related work, introduce notation, and present additional background information for the two main memory-based approaches, i.e., user-based and item-based collaborative filtering. We then introduce our similarity fusion method to unify user-based and item-based approaches. We provide an empirical evaluation of the relationship between data sparsity and the different models resulting from our framework, and finally conclude our work.

5.2 Related Work

Collaborative filtering approaches are often classified as memory-based or model-based. In the memory-based approach, all rating examples are stored *as-is* into memory (in contrast to learning an abstraction). In the prediction phase, similar users or items are sorted based on the memorized ratings. Based on the ratings of these similar users or items, a recommendation for the test user can be generated.

Examples of memory-based collaborative filtering include user-based methods [9, 37, 48, 81] and item-based methods [23, 92]. The advantage of the memory-based methods over their model-based alternatives is that less parameters have to be tuned; however, the data sparsity problem is not handled in a principled manner.

In the model-based approach, training examples are used to generate a model that is able to predict the ratings for items that a test user has not rated before. Examples include decision trees [9], aspect models [41, 99] and latent factor models [10]. The resulting ‘compact’ models solve the data sparsity problem to a certain extent. However, the need to tune an often significant number of parameters has prevented these methods from practical usage. Lately, researchers have introduced dimensionality reduction techniques to address data sparsity [31, 80, 93]. However, as pointed out in [44, 123], some useful information may be discarded during the reduction.

Recently, [44] has explored a graph-based method to deal with data sparsity, using transitive associations between user and items in the bipartite user item graph. [114] has extended the probabilistic relevance model in text retrieval ([40]) to the problem of collaborative filtering and a linear interpolation smoothing has been adopted. These approaches are however limited to binary rating data. Another recent direction in collaborative filtering research combines memory-based and model-based approaches [77, 123]. For example, [123] clus-

ters the user data and applies intra-cluster smoothing to reduce sparsity. The framework proposed in our paper extends this idea to include item-based recommendations into the final prediction, and does not require to cluster the data set a priori.

5.3 Background

This section introduces briefly the user- and item-based approaches to collaborative filtering [37, 92]. For M items and K users, the user profiles are represented in a $K \times M$ user-item matrix \mathbf{X} (Fig. 5.1(a)). Each element $x_{k,m} = r$ indicates that user k rated item m by r , where $r \in \{1, \dots, |r|\}$ if the item has been rated, and $x_{k,m} = \emptyset$ means that the rating is unknown.

The user-item matrix can be decomposed into row vectors:

$$\mathbf{X} = [\mathbf{u}_1, \dots, \mathbf{u}_K]^T, \mathbf{u}_k = [x_{k,1}, \dots, x_{k,M}]^T, k = 1, \dots, K$$

where T denotes transpose. Each row vector \mathbf{u}_k^T corresponds to a user profile and represents a particular user's item ratings. As discussed below, this decomposition leads to user-based collaborative filtering.

Alternatively, the matrix can also be represented by its column vectors:

$$\mathbf{X} = [\mathbf{i}_1, \dots, \mathbf{i}_M], \mathbf{i}_m = [x_{1,m}, \dots, x_{K,m}]^T, m = 1, \dots, M$$

where each column vector \mathbf{i}_m corresponds to a specific item's ratings by all K users. This representation results in item-based recommendation algorithms.

5.3.1 User-based Collaborative Filtering

User-based collaborative filtering predicts a test user's interest in a test item based on rating information from similar user profiles [9, 37, 81]. As illustrated in Fig. 5.1(b), each user profile (row vector) is sorted by its dis-similarity towards the test user's profile. Ratings by more similar users contribute more to predicting the test item rating. The set of similar users can be identified by employing a threshold or selecting top- N . In the top- N case, a set of top- N similar users $\mathcal{S}_{\mathbf{u}}(\mathbf{u}_k)$ towards user k can be generated according to:

$$\mathcal{S}_{\mathbf{u}}(\mathbf{u}_k) = \{\mathbf{u}_a | \text{rank } s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a) \leq N, x_{a,m} \neq \emptyset\} \quad (5.1)$$

where $|\mathcal{S}_{\mathbf{u}}(\mathbf{u}_k)| = N$. $s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)$ is the similarity between users k and a . Cosine similarity and Pearson's correlation are popular similarity measures in collaborative filtering, see e.g. [9, 37]. The similarity could also be learnt from training

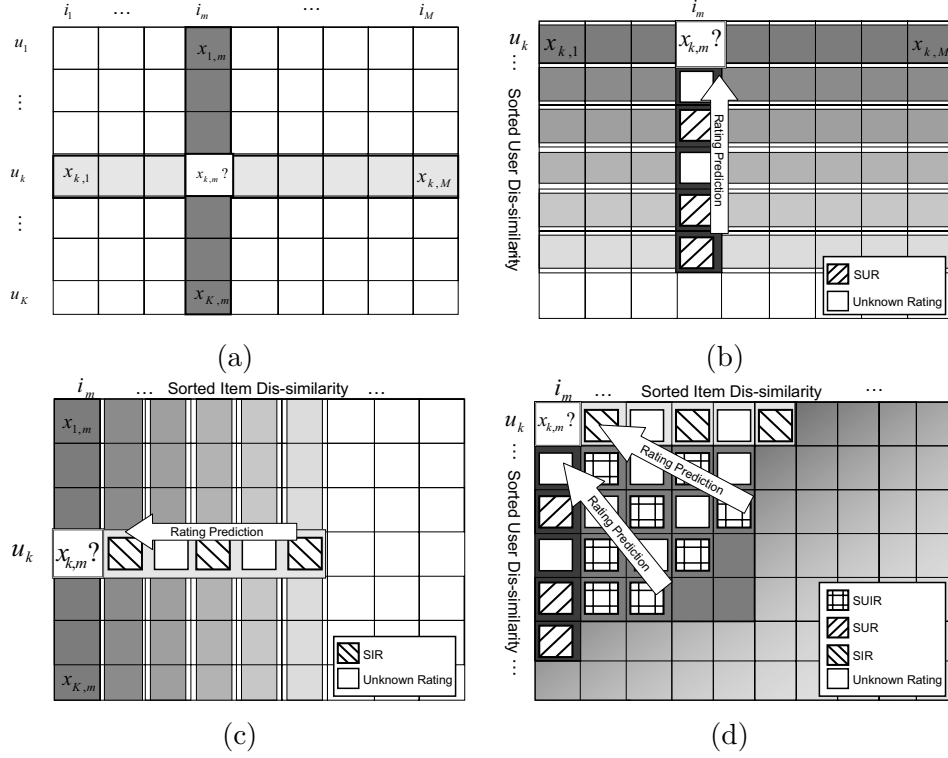


Figure 5.1: (a) The user-item matrix (b) Rating prediction based on user similarity (c) Rating prediction based on item similarity (d) Rating prediction based on rating similarity.

data [48]. This paper adopts the cosine similarity measure, comparing two user profiles by the cosine of the angle between the corresponding row vectors.

Consequently, the predicted rating $\hat{x}_{k,m}$ of test item m by test user k is computed as (see also [9, 37])

$$\hat{x}_{k,m} = \bar{u}_k + \frac{\sum_{\mathbf{u}_a \in \mathcal{S}_{\mathbf{u}}(\mathbf{u}_k)} s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)(x_{a,m} - \bar{u}_a)}{\sum_{\mathbf{u}_a \in \mathcal{S}_{\mathbf{u}}(\mathbf{u}_k)} s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)} \quad (5.2)$$

where \bar{u}_k and \bar{u}_a denote the average rating made by users k and a , respectively.

Existing methods differ in their treatment of unknown ratings from similar users ($x_{a,m} = \emptyset$). Missing ratings can be replaced by a 0 score, which lowers the prediction, or the average rating of that similar user could be used [9, 37]. Alternatively, [123] replaces missing ratings by an interpolation of the user's average rating and the average rating of his or her cluster.

Before we discuss its dual method, notice in Eq. 5.2 and the illustration in Fig. 5.1(b) how user-based collaborative filtering takes only a small proportion of the user-item matrix into consideration for recommendation. Only the *known test item ratings by similar users* are used. We refer to these ratings as the set of ‘similar user ratings’ (the blocks with upward diagonal pattern in Fig. 5.1(b)): $SUR_{k,m} = \{x_{a,m} | \mathbf{u}_a \in \mathcal{S}_u(\mathbf{u}_k)\}$. For simplicity, we drop the subscript k, m of $SUR_{k,m}$ in the remainder of the paper.

5.3.2 Item-based Collaborative Filtering

Item-based approaches such as [23, 61, 92] apply the same idea, but use similarity between items instead of users. As illustrated in Fig. 5.1(c), the unknown rating of a test item by a test user can be predicted by averaging the ratings of other similar items rated by this test user [92]. Again, each item (column vector) is sorted and re-indexed according to its dis-similarity towards the test item in the user-item matrix, and, ratings from more similar items are weighted stronger. Formally (see also [92]),

$$\hat{x}_{k,m} = \frac{\sum_{\mathbf{i}_b \in \mathcal{S}_i(\mathbf{i}_m)} s_i(\mathbf{i}_m, \mathbf{i}_b)(x_{k,b})}{\sum_{\mathbf{i}_b \in \mathcal{S}_i(\mathbf{i}_m)} s_i(\mathbf{i}_m, \mathbf{i}_b)} \quad (5.3)$$

Where item similarity $s_i(\mathbf{i}_m, \mathbf{i}_b)$ can be approximated by the cosine measure or Pearson correlation [61, 92]. To remove the difference in rating scale between users when computing the similarity, [92] has proposed to adjust the cosine similarity by subtracting the user’s average rating from each co-rated pair beforehand. We adopt this similarity measure in this paper. Like the top- N similar users, a set of top- N similar items towards item m , denoted as $\mathcal{S}_i(\mathbf{i}_m)$, can be generated according to:

$$\mathcal{S}_i(\mathbf{i}_m) = \{\mathbf{i}_b | \text{rank } s_i(\mathbf{i}_m, \mathbf{i}_b) \leq N, x_{k,b} \neq \emptyset\} \quad (5.4)$$

Fig. 5.1(c) illustrates how Eq. 5.3 takes only the *known similar item ratings by the test user* into account for prediction. We refer to these ratings as the set of ‘similar item ratings’ (the blocks with downward diagonal pattern in Fig. 5.1(c)): $SIR_{k,m} = \{x_{k,b} | \mathbf{i}_b \in \mathcal{S}_i(\mathbf{i}_m)\}$. Again, for simplicity, we drop the subscript k, m of $SIR_{k,m}$ in the remainder of the paper.

5.4 Similarity Fusion

Relying on *SUR* or *SIR* data only is undesirable, especially when the ratings from these two sources are quite often not available. Consequently, predictions are often made by averaging ratings from ‘not-so-similar’ users or items. We propose to improve the accuracy of prediction by fusing the *SUR* and *SIR* data, to complement each other under the missing data problem.

Additionally, we point out that the user-item matrix contains useful data beyond the previously used *SUR* and *SIR* ratings. As illustrated in Fig. 5.1 (d), the *similar item ratings made by similar users* may provide an extra source for prediction. They are obtained by sorting and re-indexing rows and columns according to their dis-similarities towards the test user and the test item respectively. In the remainder, this part of the matrix is referred to as ‘similar user item ratings’ (the grid blocks in Fig. 5.1(d)): $SUIR_{k,m} = \{x_{a,b} | \mathbf{u}_a \in \mathcal{S}_{\mathbf{u}}(\mathbf{u}_k), \mathbf{i}_b \in \mathcal{S}_{\mathbf{i}}(\mathbf{i}_m), a \neq k, b \neq m\}$. The subscript k, m of $SUIR_{k,m}$ is dropped.

Combining these three types of ratings in a single collaborative filtering method is non-trivial. We propose to treat each element of the user-item matrix as a separate predictor. Its reliability or *confidence* is then estimated based upon its similarity towards the test rating. We then predict the test rating by averaging the individual predictions weighted by their confidence. The remainder of the section gives a probabilistic formulation for the proposed method.

5.4.1 Individual Predictors

Users rate items differently. Some users have a preference for the extreme values of the rating scale, while others rarely deviate from the median. Likewise, items may be rated by different types of users. Some items get higher ratings than their ‘true’ value, simply because they have been rated by a positive audience. Addressing the differences in rating behavior, we first normalize the user-item matrix before making predictions.

Removing the mean ratings per user and item gives individual predictions as

$$p_{k,m}(x_{a,b}) = x_{a,b} - (\bar{x}_a - \bar{x}_k) - (\bar{x}_b - \bar{x}_m) \quad (5.5)$$

where $p_{k,m}(x_{a,b})$ is the prediction function for the test item k rating made by test user m , \bar{x}_a and \bar{x}_k are the average ratings by user a and k , and \bar{x}_b and \bar{x}_m are the average ratings of item b and m . Appendix 5.A derives that normalizing the matrix by independently subtracting the row and column means gives the same result.

5.4.2 Probabilistic Fusion Framework

Let us first define the sample space of ratings as $\Phi_r = \{\emptyset, 1, \dots, |r|\}$ (like before, \emptyset denotes the unknown rating). Let $x_{a,b}$ be a random variable over the sample space Φ_r , captured in the user-item matrix, $a \in \{1, \dots, K\}$ and $b \in \{1, \dots, M\}$. Collaborative filtering then corresponds to estimating conditional probability $P(x_{k,m}|\mathcal{P}_{k,m})$, for an unknown test rating $x_{k,m}$, given a pool of individual predictors

$$\mathcal{P}_{k,m} = \{p_{k,m}(x_{a,b})|x_{a,b} \neq \emptyset\}.$$

Consider first a pool that consists of *SUR* and *SIR* ratings only (i.e., $x_{a,b} \in (SUR \cup SIR)$).

$$P(x_{k,m}|SUR, SIR) \equiv P(x_{k,m}|\{p_{k,m}(x_{a,b})|x_{a,b} \in SUR \cup SIR\}) \quad (5.6)$$

We write $P(x_{k,m}|SUR, SIR)$ for the conditional probability depending on the predictors originating from *SUR* and *SIR*. Likewise, $P(x_{k,m}|SUR)$ and $P(x_{k,m}|SIR)$ specify a pool consisting of *SUR* or *SIR* predictors only.

Now introduce a binary variable I_1 , that corresponds to the relative importance of *SUR* and *SIR*. This hidden variable plays the same role as the prior introduced in [40] to capture the importance of a query term in information retrieval. $I_1 = 1$ states that $x_{k,m}$ depends completely upon ratings from *SUR*, while $I_1 = 0$ corresponds to full dependency on *SIR*. Under these assumptions, the conditional probability can be obtained by marginalization of variable I_1 :

$$\begin{aligned} & P(x_{k,m}|SUR, SIR) \\ &= \sum_{I_1} P(x_{k,m}|SUR, SIR, I_1)P(I_1|SUR, SIR) \\ &= P(x_{k,m}|SUR, SIR, I_1 = 1)P(I_1 = 1|SUR, SIR) + \\ & \quad P(x_{k,m}|SUR, SIR, I_1 = 0)P(I_1 = 0|SUR, SIR) \end{aligned} \quad (5.7)$$

By definition, $x_{k,m}$ is independent from *SUR* when $I_1 = 1$, so $P(x_{k,m}|SUR, SIR, I_1 = 1) = P(x_{k,m}|SUR)$. Similarly, $P(x_{k,m}|SUR, SIR, I_1 = 0) = P(x_{k,m}|SIR)$. If we provide a parameter λ as shorthand for $P(I_1 = 1|SUR, SIR)$, we have

$$P(x_{k,m}|SUR, SIR) = P(x_{k,m}|SUR)\lambda + P(x_{k,m}|SIR)(1 - \lambda) \quad (5.8)$$

Next, we extend the model to take into account the *SUIR* ratings:

$$P(x_{k,m}|SUR, SIR, SUIR) \equiv P(x_{k,m}|\{p_{k,m}(x_{a,b})|x_{a,b} \in SUR \cup SIR \cup SUIR\}) \quad (5.9)$$

We introduce a second binary random variable I_2 , that corresponds to the relative importance of the *SUIR* predictors. $I_2 = 1$ specifies that the unknown rating depends on ratings from *SUIR* only and $I_2 = 0$ that it depends on the ratings from *SIR* and *SUR* instead. Marginalization on variable I_2 gives:

$$\begin{aligned}
& P(x_{k,m}|SUR, SIR, SUIR) \\
&= \sum_{I_2} P(x_{k,m}|SUR, SIR, SUIR, I_2)P(I_2|SUR, SIR, SUIR) \\
&= P(x_{k,m}|SUR, SIR, SUIR, I_2 = 1) \cdot P(I_2 = 1|SUR, SIR, SUIR) + \\
&\quad P(x_{k,m}|SUR, SIR, SUIR, I_2 = 0) \cdot (1 - P(I_2 = 1|SUR, SIR, SUIR))
\end{aligned} \tag{5.10}$$

Following the argument from above and providing a parameter δ as shorthand for $P(I_2 = 1|SUR, SIR, SUIR)$, we have

$$\begin{aligned}
& P(x_{k,m}|SUR, SIR, SUIR) \\
&= P(x_{k,m}|SUR, SIR)(1 - \delta) + P(x_{k,m}|SUIR)\delta
\end{aligned} \tag{5.11}$$

Substitution of Eq. 5.8 then gives:

$$\begin{aligned}
& P(x_{k,m}|SUR, SIR, SUIR) \\
&= \left(P(x_{k,m}|SUR)\lambda + P(x_{k,m}|SIR)(1 - \lambda) \right) (1 - \delta) + \\
&\quad P(x_{k,m}|SUIR)\delta
\end{aligned} \tag{5.12}$$

Finally, the following equation gives the expected value of the unknown test rating:

$$\begin{aligned}
\hat{x}_{k,m} &= \sum_{r=1}^{|r|} r P(x_{k,m} = r|SUR, SIR, SUIR) \\
&= \left(\sum_{r=1}^{|r|} r P(x_{k,m} = r|SUIR)\delta \right) + \\
&\quad \left(\sum_{r=1}^{|r|} r P(x_{k,m} = r|SUR)\lambda(1 - \delta) \right) + \\
&\quad \left(\sum_{r=1}^{|r|} r P(x_{k,m} = r|SIR)(1 - \lambda)(1 - \delta) \right)
\end{aligned} \tag{5.13}$$

The resulting model can be viewed as using importance sampling of the neighborhood ratings as predictors. λ and δ control the selection (sampling) of data from the three different sources.

5.4.3 Probability Estimation

The next step is to estimate the probabilities in the fusion framework expressed in Eq. 5.13.

λ and δ are determined experimentally by using the cross-validation, for example following the methodology of Section 5.5.3. The three remaining probabilities can be viewed as estimates of the likelihood of a rating $x_{a,b}$ from *SIR*, *SUR*, or *SUIR*, to be similar to the test rating $x_{k,m}$. We assume that the probability estimates for *SUR* and *SIR* are proportional to the similarity between row vectors $s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)$ (Section 5.3.1) and column vectors $s_{\mathbf{i}}(\mathbf{i}_m, \mathbf{i}_b)$ (Section 5.3.2), respectively. For *SUIR* ratings, we assume the probability estimate to be proportional to the combination of $s_{\mathbf{u}}$ and $s_{\mathbf{i}}$. To combine them, we use a Euclidean dis-similarity space such that the resulting combined similarity is lower than either of them.

$$s_{\mathbf{ui}}(x_{k,m}, x_{a,b}) = \frac{1}{\sqrt{(1/s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a))^2 + (1/s_{\mathbf{i}}(\mathbf{i}_m, \mathbf{i}_b))^2}} \quad (5.14)$$

This results in the following conditional probability estimates:

$$\begin{aligned} P(x_{k,m} = r | \text{SUR}) &= \frac{\sum_{\forall x_{a,b}: (x_{a,b} \in \text{SUR}) \wedge (p_{k,m}(x_{a,b})=r)} s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)}{\sum_{\forall x_{a,b}: x_{a,b} \in \text{SUR}} s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)} \\ P(x_{k,m} = r | \text{SIR}) &= \frac{\sum_{\forall x_{a,b}: (x_{a,b} \in \text{SIR}) \wedge (p_{k,m}(x_{a,b})=r)} s_{\mathbf{i}}(\mathbf{i}_m, \mathbf{i}_b)}{\sum_{\forall x_{a,b}: x_{a,b} \in \text{SIR}} s_{\mathbf{i}}(\mathbf{i}_m, \mathbf{i}_b)} \\ P(x_{k,m} = r | \text{SUIR}) &= \frac{\sum_{\forall x_{a,b}: (x_{a,b} \in \text{SUIR}) \wedge (p_{k,m}(x_{a,b})=r)} s_{\mathbf{ui}}(x_{k,m}, x_{a,b})}{\sum_{\forall x_{a,b}: x_{a,b} \in \text{SUIR}} s_{\mathbf{ui}}(x_{k,m}, x_{a,b})} \end{aligned} \quad (5.15)$$

After substitution from Eq. 5.15 (for readability, we put the detailed derivations in Appendix 5.B), Eq. 5.13 results in:

$$\hat{x}_{k,m} = \sum_{x_{a,b}} p_{k,m}(x_{a,b}) W_{k,m}^{a,b} \quad (5.16)$$

where

$$W_{k,m}^{a,b} = \begin{cases} \frac{\sum_{x_{a,b} \in SUR} s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)}{\sum_{x_{a,b} \in SUR} s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)} \lambda (1 - \delta) & x_{a,b} \in SUR \\ \frac{\sum_{x_{a,b} \in SIR} s_{\mathbf{i}}(\mathbf{i}_m, \mathbf{i}_b)}{\sum_{x_{a,b} \in SIR} s_{\mathbf{i}}(\mathbf{i}_m, \mathbf{i}_b)} (1 - \lambda)(1 - \delta) & x_{a,b} \in SIR \\ \frac{\sum_{x_{a,b} \in SUIR} s_{\mathbf{ui}}(x_{k,m}, x_{a,b})}{\sum_{x_{a,b} \in SUIR} s_{\mathbf{ui}}(x_{k,m}, x_{a,b})} \delta & x_{a,b} \in SUIR \\ 0 & otherwise \end{cases} \quad (5.17)$$

It is easy to prove that $\sum_{x_{a,b}} W_{k,m}^{a,b} = 1$. $W_{k,m}^{a,b}$ acts as a unified weight matrix to combine the predictors from the three different sources.

5.4.4 Discussions

Sum as Combination Rule λ and δ control the importance of the different rating sources. Their introduction results in a sum rule for fusing the individual predictors (Eq. 5.12 and 5.16.). Using the independence assumption on the three types of ratings and the Bayes' rule, one can easily derive a product combination from the conditional probability ([53]). However, the high sensitivity to estimation errors makes this approach less attractive in practice. We refer to [53] for a more detailed discussion of using a sum rule vs. the product rule for combining classifiers.

Unified Weights The unified weights in Eq. 5.17 provide a generative framework for memory-based collaborative filtering.

Eq. 5.17 shows how our scheme can be considered as two subsequent steps of linear interpolation. First, predictions from *SUR* ratings are interpolated with *SIR* ratings, controlled by λ . Next, the intermediate prediction is interpolated with predictions from the *SUIR* data, controlled by δ . Viewing the *SUIR* ratings as a background model, the second interpolation corresponds to smoothing the *SIR* and *SUR* predictions from the background model.

A bigger λ emphasizes user correlations, while smaller λ emphasizes item correlations. When λ equals one, our algorithm corresponds to a user-based approach, while λ equal to zero results in an item-based approach.

Tuning parameter δ controls the impact of smoothing from the background model (i.e. *SUIR*). When δ approaches zero, the fusion framework becomes the mere combination of user-based and item-based approaches without smoothing from the background model.

Table 5.1: Percentage of the ratings that are available ($\neq \emptyset$).

| | test item | 1st most sim | 2nd most sim | 3rd most sim | 4th most sim |
|--------------------|-----------|--------------|--------------|--------------|--------------|
| test user | - | 0.58 | 0.56 | 0.55 | 0.54 |
| 1st most sim. user | 0.54 | 0.58 | 0.58 | 0.58 | 0.57 |
| 2nd most sim. user | 0.51 | 0.56 | 0.56 | 0.56 | 0.56 |
| 3rd most sim. user | 0.51 | 0.57 | 0.57 | 0.57 | 0.56 |
| 4th most sim. user | 0.49 | 0.55 | 0.55 | 0.56 | 0.55 |

Table 5.2: Mean Absolute Err (MAE) of individual predictions.

| | test item | 1st most sim | 2nd most sim | 3rd most sim | 4th most sim |
|--------------------|-----------|--------------|--------------|--------------|--------------|
| test user | - | 0.824 | 0.840 | 0.866 | 0.871 |
| 1st most sim. user | 0.914 | 0.925 | 0.927 | 0.942 | 0.933 |
| 2nd most sim. user | 0.917 | 0.921 | 0.931 | 0.935 | 0.927 |
| 3rd most sim. user | 0.927 | 0.947 | 0.952 | 0.953 | 0.945 |
| 4th most sim. user | 0.928 | 0.929 | 0.939 | 0.946 | 0.932 |

5.5 Empirical Evaluation

5.5.1 Experimental Setup

We experimented with the MovieLens¹, EachMovie², and book-crossing³ data sets. While we report only the MovieLens results (out of space considerations), the model behaves consistently across the three data sets.

The MovieLens data set contains 100,000 ratings (1-5 scales) from 943 users on 1682 movies (items), where each user has rated at least 20 items. To test on different number of training users, we selected the users in the data set at random into a training user set (100, 200, 300 training users, respectively) and the remaining users into a test user set. Users in the training set are only used for making predictions, while test users are the basis for measuring prediction accuracy. Each test user's ratings have been split into a set of *observed items* and one of *held-out items*. The ratings of observed items are input for predicting the ratings of held-out items.

We are specifically interested in the relationship between the density of the user-item matrix and the collaborative filtering performance. Consequently, we set up the following configurations:

- **Test User Sparsity** Vary the number of items rated by test users in the observed set, e.g., 5, 10, or 20 ratings per user.
- **Test Item Sparsity** Vary the number of users who have rated test items in the held-out set; less than 5, 10, or 20 (denoted as '< 5', '< 10', or '<

¹<http://www.grouplens.org/>

²<http://research.compaq.com/SRC/eachmovie/>

³<http://www.informatik.uni-freiburg.de/~chiegler/BX/>

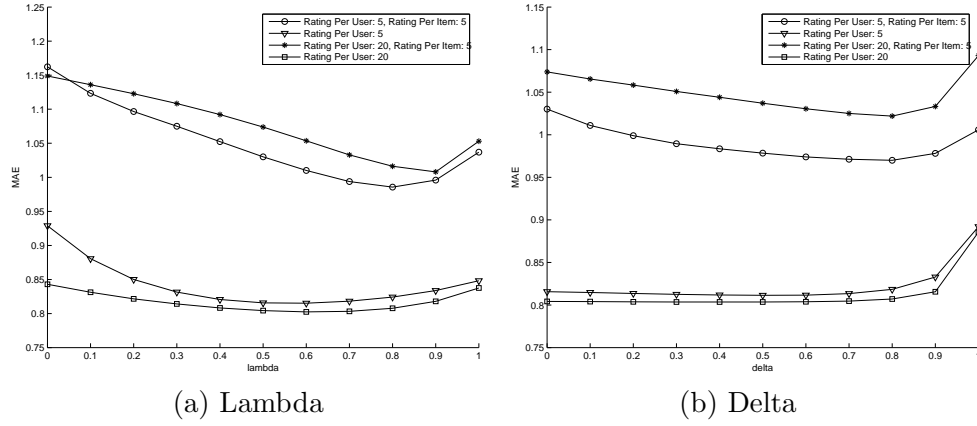


Figure 5.2: Impact of the two parameters.

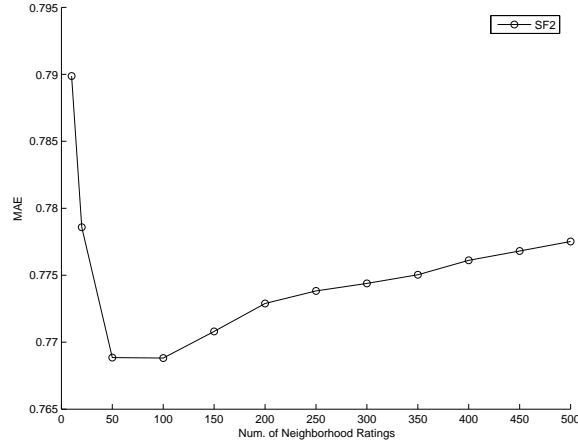


Figure 5.3: Size of neighborhood.

20'), or, unconstrained (denoted as 'No constraint').

- **Overall Training User Sparsity** Select a part of the rating data at random, e.g., 20%, 40%, 60% of the data set.

For consistency with experiments reported in the literature, e.g., [48, 92, 123]), we report the mean absolute error (MAE) evaluation metric. MAE corresponds to the average absolute deviation of predictions to the ground truth data, for all test item ratings and test users:

$$MAE = \frac{\sum_{k,m} |x_{k,m} - \hat{x}_{k,m}|}{L}, \quad (5.18)$$

where L denotes the number of tested ratings. A smaller value indicates a better performance.

5.5.2 Individual Predictors

We first report some properties of the three types of individual predictions used in our approach. Table 5.1 illustrates the availability of the top-4 neighborhood ratings in the MovieLens data set. The first column contains the top-4 *SUR* ratings, the first row the top-4 *SIR* ratings; the remaining cells correspond to the top-4x4 *SUIR* ratings. We observe that only about half of these ratings are given. Table 5.2 summarizes recommendation MAE of individual predictors (applying Eq. 5.5) using leave-one-out cross-validation. Clearly, more similar ratings provide more accurate predictions. While *SUIR*s ratings are in general less accurate than *SUR*s and *SIR*s, these may indeed complement missing or unreliable *SIR* and *SUR* ratings.

5.5.3 Impact of Parameters

Recall the two parameters in Eq. 5.17: λ balances the predictions between *SUR* and *SIR*, and δ smoothes the fused results by interpolation with a pool of *SUIR* ratings.

We first test the sensitivity of λ , setting δ to zero. This scheme, called *SF1*, combines user-based and item-based approaches, but does not use additional background information. Fig. 5.2(a) shows recommendation MAE against varying λ from zero (a pure item-based approach) to one (a pure user-based approach). The graph plots test user sparsity 5 and 20, and test item sparsity settings ‘< 5’ and unconstrained. The value of the optimal λ demonstrates that interpolation between user-based and item-based approaches (*SF1*) improves the recommendation performance. More specifically, the best results are obtained with λ between 0.6 and 0.9. This optimal value emphasizing the *SUR* ratings may be somewhat surprising, as Table 5.2 indicated that the *SIR* ratings should be more reliable for prediction. However, in the data sets considered, the number of users is smaller than the number of items, causing the user weights $s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)$ to be generally smaller than the item weights $s_{\mathbf{i}}(\mathbf{i}_m, \mathbf{i}_b)$. When removing the constraint on test item sparsity, the optimal λ shifts down from about 0.9 for the two upper curves (‘< 5’) to 0.6 for the two lower curves (unconstrained). A lower λ confirms the expectation that *SIR* ratings gain value when more items have been rated.

Fig. 5.2 (b) shows the sensitivity of δ after fixing λ to 0.7. The graph plots the MAE for the same four configurations when parameter δ is varied from

Table 5.3: Comparison with other memory-based approaches. A smaller value means a better performance.

| $ L_{i_m} $ | < 5 | | | < 10 | | | < 20 | | | No constrain | | |
|-------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| $ L_{u_k} $ | 5 | 10 | 20 | 5 | 10 | 20 | 5 | 10 | 20 | 5 | 10 | 20 |
| SF2 | 1.054 | 0.966 | 1.070 | 0.995 | 0.917 | 0.997 | 0.945 | 0.879 | 0.923 | 0.825 | 0.794 | 0.805 |
| SF1 | 1.086 | 1.007 | 1.097 | 1.035 | 0.942 | 1.024 | 0.976 | 0.898 | 0.936 | 0.836 | 0.796 | 0.809 |
| UBVS | 1.129 | 1.034 | 1.117 | 1.052 | 0.972 | 1.054 | 0.996 | 0.913 | 0.969 | 0.891 | 0.809 | 0.836 |
| IBVS | 1.190 | 1.055 | 1.131 | 1.108 | 0.992 | 1.068 | 1.066 | 0.954 | 0.977 | 0.938 | 0.842 | 0.842 |

(a) Number of Training Users: 100

| $ L_{i_m} $ | < 5 | | | < 10 | | | < 20 | | | No constrain | | |
|-------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| $ L_{u_k} $ | 5 | 10 | 20 | 5 | 10 | 20 | 5 | 10 | 20 | 5 | 10 | 20 |
| SF2 | 0.960 | 0.945 | 0.948 | 0.915 | 0.875 | 0.885 | 0.826 | 0.802 | 0.828 | 0.806 | 0.786 | 0.803 |
| SF1 | 0.976 | 0.960 | 0.963 | 0.927 | 0.883 | 0.895 | 0.832 | 0.804 | 0.831 | 0.808 | 0.786 | 0.804 |
| UBVS | 1.108 | 1.028 | 1.024 | 1.070 | 0.962 | 0.972 | 0.914 | 0.842 | 0.885 | 0.879 | 0.811 | 0.848 |
| IBVS | 1.187 | 1.071 | 1.034 | 1.122 | 1.006 | 0.976 | 0.974 | 0.875 | 0.886 | 0.921 | 0.840 | 0.847 |

(b) Number of Training Users: 200

| $ L_{i_m} $ | < 5 | | | < 10 | | | < 20 | | | No constrain | | |
|-------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| $ L_{u_k} $ | 5 | 10 | 20 | 5 | 10 | 20 | 5 | 10 | 20 | 5 | 10 | 20 |
| SF2 | 0.956 | 0.908 | 0.941 | 0.911 | 0.885 | 0.912 | 0.842 | 0.828 | 0.859 | 0.798 | 0.782 | 0.805 |
| SF1 | 1.013 | 0.968 | 0.977 | 0.928 | 0.908 | 0.938 | 0.847 | 0.834 | 0.867 | 0.802 | 0.783 | 0.807 |
| UBVS | 1.024 | 0.971 | 1.044 | 0.966 | 0.919 | 0.980 | 0.921 | 0.877 | 0.936 | 0.886 | 0.808 | 0.852 |
| IBVS | 1.117 | 1.043 | 1.024 | 1.044 | 0.990 | 1.004 | 0.962 | 0.910 | 0.932 | 0.914 | 0.837 | 0.850 |

(c) Number of Training Users: 300

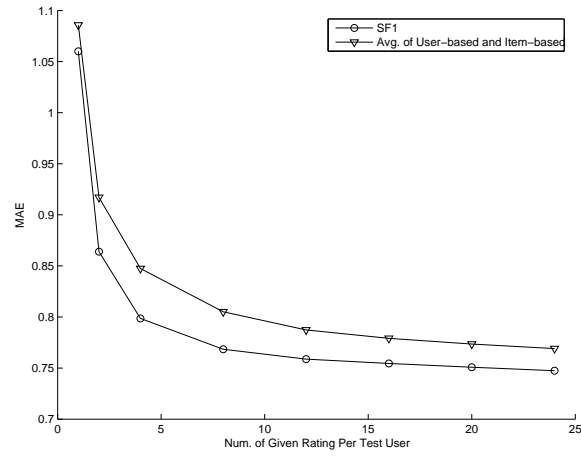
zero (without smoothing) to one (rely solely on the background model: *SUIR* ratings). When δ is non-zero, the *SF1* results are smoothed by a pool of *SUIR* ratings, which we called fusion scheme *SF2*. We observe that δ reaches its optimal in 0.8 when the rating data is sparse in the neighborhood ratings from the item and user aspects (upper two curves). In other words, smoothing from a pool of *SUIR* ratings improves the performance for sparse data. However, when the test item sparsity is not constrained, its optimum spreads a wide range of values, and the improvement over MAE without smoothing ($\delta = 0$) is not clear.

Additional experiments (not reported here) verified that there is little dependency between the choice of λ and the optimal value of δ . The optimal parameters can be identified by using the cross validation from the training data.

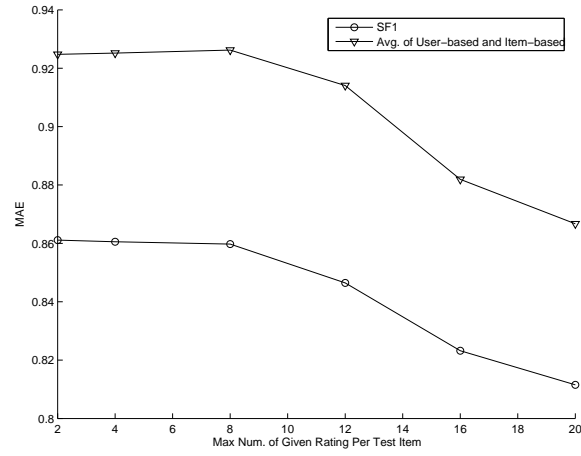
Like pure user-based and item-based approaches, the size of neighborhood N also influences the performance of our fusion methods. Fig. 5.3 shows MAE of *SF2* when the number of neighborhood ratings is varied. The optimal results are obtained with the neighborhood size between 50 and 100. We select 50 as our optimal choice.

5.5.4 Data Sparsity

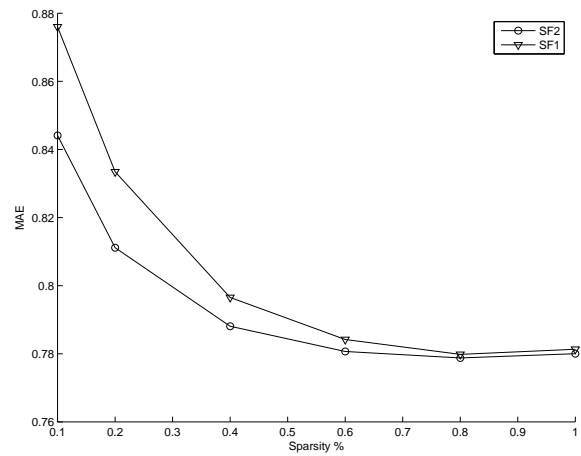
The next experiments investigate the effect of data sparsity on the performance of collaborative filtering in more detail. Fig. 5.4(a) and (b) compare the behavior of scheme *SF1* to that obtained by simply averaging user-based and item-based approaches, when varying test user sparsity (Fig. 5.4(a)) and test item sparsity (5.4(b)). The results indicate that combining user-based and



(a) Test User Sparsity



(b) Test Item Sparsity



(c) Overall Training User Sparsity

Figure 5.4: Performance under different sparsity.

Table 5.4: Comparison with the result reported in [123]. A smaller value means a better performance.

| Num. of Training Users: Ratings Given (Test User): | 100 | | | 200 | | | 300 | | |
|---|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| | 5 | 10 | 20 | 5 | 10 | 20 | 5 | 10 | 20 |
| SF2 | 0.847 | 0.774 | 0.792 | 0.827 | 0.773 | 0.783 | 0.804 | 0.761 | 0.769 |
| SCBPCC | 0.848 | 0.819 | 0.789 | 0.831 | 0.813 | 0.784 | 0.822 | 0.810 | 0.778 |
| AM | 0.963 | 0.922 | 0.887 | 0.849 | 0.837 | 0.815 | 0.820 | 0.822 | 0.796 |
| PD | 0.849 | 0.817 | 0.808 | 0.836 | 0.815 | 0.792 | 0.827 | 0.815 | 0.789 |
| PCC | 0.874 | 0.836 | 0.818 | 0.859 | 0.829 | 0.813 | 0.849 | 0.841 | 0.820 |

item-based approaches (*SF1*) consistently improves the recommendation performance regardless neighborhood sparsity of test users or items.

Next, Fig. 5.4(c) plots the gain of *SF2* over *SF1* when varying overall training user sparsity. The figure shows that *SF2* improves *SF1* more and more when the rating data becomes more sparse. This can be explained as follows. When the user-item matrix is less dense, it contains insufficient test item ratings by similar users (for user-based recommendation), and insufficient similar item ratings by the test user (for item-based recommendation) as well. Therefore, smoothing using ratings by similar items made by similar users improves predictions.

We conclude from these experiments that the proposed fusion framework is effective at improving the quality of recommendations, even when only sparse data are available.

5.5.5 Comparison to Other Methods

We continue with a comparison to results obtained with other methods, setting λ to 0.7 and δ to 0 for *SF1* and using $\lambda = 0.7$ and $\delta = 0.7$ for *SF2*. We first compare our results to the standard user-based vector similarity (UBVS) approach of [9] and the item-based adjusted cosine similarity (IBVS) of [92]. We report results for test user sparsity 5, 10, or 20, and test item sparsity ‘< 5’, ‘< 10’, ‘< 20’ or ‘No constrain’. Table 5.3 summarizes the results, showing how *SF1* and *SF2* outperform the other methods in all twelve resulting configurations.

Next, we adopt the subset of MovieLens (see [48, 123]), which consists of 500 users and 1000 items. We followed the exact evaluation procedure described in [123] to compare the performance of our *SF2* scheme with the state-of-art results listed in [123]. Table 5.4 presents our experimental results, as well as the four best methods according to their experiments, i.e., cluster-based Pearson Correlation Coefficient (SCBPCC) [123], the Aspect Model (AM) ([41]), ‘Personality Diagnosis’ (PD) ([77]) and the user-based Pearson Correlation Coefficient (PCC) ([9]). Our method outperforms these methods in all configurations.

5.6 Conclusions

We proposed a novel algorithm to unify the user-based and item-based collaborative filtering approaches to overcome limitations specific to either of them. We showed that user-based and item-based approaches are only two special cases in our probabilistic fusion framework. Furthermore, by using a linear interpolation smoothing, other ratings by similar users towards similar items can be treated as a background model to smooth the rating predictions. The experiments showed that our new fusion framework is effective in improving the prediction accuracy of collaborative filtering and dealing with the data sparsity problem. In the future, we plan to conduct better formal analyses of the fusion model and more complete comparisons with previous methods.

5.A Normalization

We first normalize the matrix by subtracting the average item ratings:

$$n(x_{a,b})_I = x_{a,b} - \frac{1}{K} \sum_i x_{i,b} = x_{a,b} - \bar{x}_b$$

where $n(x_{a,b})_I$ normalizes ratings by subtracting the mean item rating. \bar{x}_b is the average rating of item b .

We normalize again by the average user rating:

$$\begin{aligned} n(x_{a,b})_{I,U} &= n(x_{a,b})_I - \frac{1}{M} \sum_j n(x_{a,j})_I \\ &= x_{a,b} - \frac{1}{K} \sum_i x_{i,b} - \frac{1}{M} \sum_j \left(x_{a,j} - \frac{1}{K} \sum_i x_{i,j} \right) \\ &= x_{a,b} - \frac{1}{K} \sum_i x_{i,b} - \frac{1}{M} \sum_j x_{a,j} + \frac{1}{MK} \sum_{i,j} x_{i,j} \\ &= x_{a,b} - \bar{x}_b - \bar{x}_a + \bar{x} \end{aligned}$$

where $n(x_{a,b})_{I,U}$ is the normalization of both item and user aspects. \bar{x}_a is the average rating from user a . \bar{x} is the average of all the ratings. From here, we see that the result does not depend on the order of normalization (whether to normalize first by user or by item).

Treating each normalized individual rating as individual predictor results in:

$$\begin{aligned} \hat{x}_{k,m} - \bar{x}_m - \bar{x}_k + \bar{x} &= x_{a,b} - \bar{x}_b - \bar{x}_a + \bar{x} \\ \therefore p_{k,m}(x_{a,b}) = \hat{x}_{k,m} &= x_{a,b} - (\bar{x}_a - \bar{x}_k) - (\bar{x}_b - \bar{x}_m) \end{aligned}$$

5.B A Unified Weighting Function

More specifically, replacing three conditional probabilities with Eq. 5.15, the following can be derived from Eq. 5.13:

$$\begin{aligned}
\hat{x}_{k,m} &= \sum_{r=1}^{|r|} r \left(\sum_{p_{k,m}(x_{a,b})=r} \left(\sum_{x_{a,b} \in SUR} A + \sum_{x_{a,b} \in SIR} B + \sum_{x_{a,b} \in SUIR} C \right) \right) \\
&= \sum_{\forall x_{a,b}: x_{a,b} \in SUR} p_{k,m}(x_{a,b}) A + \sum_{\forall x_{a,b}: x_{a,b} \in SIR} p_{k,m}(x_{a,b}) B + \\
&\quad \sum_{\forall x_{a,b}: x_{a,b} \in SUIR} p_{k,m}(x_{a,b}) C
\end{aligned}$$

where

$$\begin{aligned}
A &= \frac{s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)}{\sum_{\forall x_{a,b}: x_{a,b} \in SUR} s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)} \lambda(1 - \delta) \\
B &= \frac{s_{\mathbf{i}}(\mathbf{i}_m, \mathbf{i}_b)}{\sum_{\forall x_{a,b}: x_{a,b} \in SIR} s_{\mathbf{i}}(\mathbf{i}_m, \mathbf{i}_b)} (1 - \lambda)(1 - \delta) \\
C &= \frac{s_{\mathbf{ui}}(x_{k,m}, x_{a,b})}{\sum_{\forall x_{a,b}: x_{a,b} \in SUIR} s_{\mathbf{ui}}(x_{k,m}, x_{a,b})} \delta
\end{aligned}$$

where A,B and C act as the weights to combine the predictors from three different sources. Unifying them we can obtain Eq. 5.16.

Chapter 6

Unified Relevance Models

This paper views collaborative filtering as a problem highly related to information retrieval, drawing an analogy between the concepts of users and items in recommender systems and queries and documents in text retrieval.

We present a probabilistic user-to-item relevance framework that introduces the concept of relevance into the related problem of collaborative filtering. Three different models are derived, namely, a *user-based*, an *item-based* and a *unified relevance model*, estimating their rating predictions from three sources: the user's own ratings for different items, other users' ratings for the same item, and, ratings from different but similar users for other but similar items.

To reduce the data sparsity encountered when estimating the probability density function of the relevance variable, we apply the non-parametric (data-driven) density estimation technique known as the *Parzen-window method* (or, kernel-based density estimation). Using a Gaussian window function, the similarity between users and/or items would however be based on Euclidean distance. Because collaborative filtering literature has reported improved prediction accuracy when using cosine similarity, we generalise the Parzen-window method by introducing a *projection kernel*.

Existing user-based and item-based approaches correspond to two simplified instantiations of our framework. User-based and item-based collaborative filtering represent only a partial view of the prediction problem, where the unified relevance model brings these partial views together under the same umbrella. Experimental results complement the theoretical insights with improved recommendation accuracy. The unified model is more robust to data sparsity, because

This work is to appear in ACM Trans. on Information Systems (TOIS), 2008. The authors are J. Wang, A. P. de Vries, and M. J. T. Reinders. See also [115].

the different types of ratings are used in concert.

6.1 Introduction

Collaborative filtering (CF) algorithms use a collection of user profiles to identify interesting “information” (items) for these users. These profiles result from asking users explicitly to rate items (rating-based CF), or, they are inferred from log-archives (log-based CF) [41, 114]. The profiles can be thought of as the evidence (observation) of *relevance* between users and items. Thus, the task of collaborative filtering is to predict the unknown relevance between the test user and the test item [114].

Relevance is also a very important concept in the text retrieval domain and has been heavily studied (e.g., [112, 57]). Many probabilistic approaches have been developed to model the estimation of relevance, ranging from the traditional probabilistic models [86] to the latest developments on language models of information retrieval [58, 56].

Despite the concept of relevance existing in both collaborative filtering and text retrieval, collaborative filtering has often been formulated as a self-contained problem, apart from the classic information retrieval problem (i.e. ad hoc text retrieval). Research started with memory-based approaches to collaborative filtering, that can be divided in user-based approaches like [81, 9, 37, 48] and item-based approaches like [92, 23]. Given an unknown test rating (of a test item by a test user) to be estimated, memory-based collaborative filtering first measures similarities between the test user and all other users (user-based), or, between the test item and all other items (item-based). Then, the unknown rating is predicted by averaging the (weighted) known ratings of the test item by the similar users (user-based), or the (weighted) known ratings of the similar items by the test user (item-based).

The two approaches share the same drawback as the *document-oriented* and *query-oriented* views in information retrieval (IR) [82]: neither presents a complete view on the problem. In both the user-based and item-based approaches, only partial information from the data embedded in the *user-item matrix* is employed to predict unknown ratings, i.e. using either correlation between user data or correlation between item data. Because of the sparsity of user profile data, however, many ratings will not be available. Therefore, it is desirable to unify the ratings from both similar users and similar items, to reduce the dependency on data that is often missing. Even ratings made by other but similar users on other but similar items can be used to make predictions [116]. Not using such ratings causes the *data sparsity problem* of memory-based approaches to collaborative filtering: for many users and items, no reliable recommendation

can be made because of a lack of similar ratings.

Thus it is of great interest to see how we can follow the school of thinking in IR relevance models to solve the collaborative filtering problem, once we draw an analogy between the concept of user and items in collaborative filtering and query and documents in IR. This paper applies IR relevance models at a conceptual level, setting up a unified probabilistic relevance framework to exploit more of the data available in the user-item matrix. We establish the models of relevance by considering user ratings of items as observations of the relevance between users and items. Different from any other IR relevance models, the densities of the our relevance models are estimated by applying the Parzen-window approach [25]. This approach reduces the data sparsity encountered when estimating the densities, providing a data-driven solution, i.e., without specifying the model structure a priori. We then extend the Parzen-window density estimation into a projected space, to provide a generalised distance measure which naturally includes most of commonly used similarity measures into our framework.

We derive three types of models from the framework, namely the *user-based relevance model*, the *item-based relevance model* and the *unified relevance model*. The former two models represent a partial view of the problem. In the unified relevance model, each individual rating in the user-item matrix may influence the prediction for the unknown test rating (of a test item from a test user). The overall prediction is made by averaging the individual ratings weighted by their contribution. The weighting is controlled by three factors: the shape of the Parzen window, the two (user and item) bandwidth parameters, and the distance metric to the test rating. The more a rating contributes towards the test rating, the higher the weight assigned to that rating to make the prediction. Under the framework, the item-based and user-based approaches are two special cases and they are systematically combined. By doing this, our approach allows us to take advantage of user similarity *and* item similarity embedded in the user-item matrix to improve probability estimation and counter the problem of data sparsity.

The remainder of the paper is organized as follows. We first summarise related work, introduce notation, and present additional background information for the three main memory-based approaches, i.e., user-based, item-based and the combined collaborative filtering approaches. We then introduce our unified relevance prediction models. We provide an empirical evaluation of the relationship between data sparsity and the different models resulting from our framework, and finally conclude our work.

6.2 Related Work

6.2.1 Collaborative Filtering

Collaborative filtering approaches are often classified as memory-based or model-based. In the memory-based approach, all rating examples are stored *as-is* into memory (in contrast to learning an abstraction). In the prediction phase, similar users or items are sorted based on the memorized ratings. Based on the ratings of these similar users or items, a recommendation for the test user can be generated. Different views of the correlations embedded in the user-item matrix lead to two different types of approaches: user-based methods (looking at user correlation) [81, 9, 37, 48] and item-based methods (looking at item correlation) [92, 23]. These approaches form a heuristic implementation of the “Word of Mouth” phenomenon and are widely used in practice, e.g., [37, 61]. The advantage of the memory-based methods over their model-based alternatives is that less parameters have to be tuned; however, the data sparsity problem is not handled in a principled manner. Lately, researchers have introduced dimensionality reduction techniques to address the data sparsity [93, 31, 80]. But, as pointed out in [44, 123], some useful information may be discarded during the reduction. [123] clusters the user data and applies intra-cluster smoothing to reduce sparsity. [43] extends this idea, by further adding a linear interpolation between the user-based and item-based approaches within the user cluster. The *similarity fusion* method presented in [116] can be also regarded as an effort along this direction, which implicitly clusters users and items simultaneously. Different from [43], [116] takes a rather formal view on the fusion problem and proposes a multi-layer linear smoothing model, showing that additional ratings from similar users towards similar items are also valuable to counter the data sparsity.

In the model-based approach, training examples are used to generate a model that is able to predict the ratings for items that a test user has not rated before. In this regard, many probabilistic models have been proposed. For example, to consider user correlation, [77] proposed a method called personality diagnosis (PD), treating each user as a separate cluster and assuming a Gaussian noise applied to all ratings. It computes the probability that a test user is of the same “personality type” as other users and, in turn, the probability of his or her rating to a test item can be predicted. On the other hand, to model item correlation, [9] utilizes a Bayesian Network model, in which the conditional probabilities between items are maintained. Some researchers have tried mixture models, explicitly assuming some hidden variables embedded in the rating data. Examples include the aspect model [41, 99], the cluster model [9] and the latent factor model [10]. These methods require some assumptions about the underlying data structures and the resulting ‘compact’ models solve

the data sparsity problem to a certain extent. However, the need to tune an often significant number of parameters has prevented these methods from practical usage.

In contrast, the work presented in this paper takes a data-driven approach without assuming any data structure a priori, thus getting rid of the significant number of model parameters. It extends the ideas in [116] by further exploring the usage of the probabilistic model of text retrieval. The Parzen-window method is adopted for probability estimation, causing a natural integration of user and item correlations. As a result, the proposed prediction framework is of a general nature. We shall see how some of the previously proposed methods, such as the PD algorithm [77], the Similarity Fusion method [116] and the linear combination method [43], are equivalent to one of the simplified instantiations of our framework (Table 6.3).

6.2.2 Probabilistic Models for Information Retrieval

Probabilistic (relevance) models for information (text) retrieval have been proposed and tested over decades. Rather than giving a comprehensive overview, here we mainly review the models that are related to the problem of collaborative filtering.

The two different *document-oriented* and *query-oriented* views on how to assign a probability of relevance of a document to a user need result in two different types of practical models [85, 8]. The RSJ probabilistic model of information retrieval [86] takes the query-oriented view, and estimates the log ratio of relevance versus non-relevance given the document and the query. To instantiate this model, there is no need to directly represent the documents (e.g., using terms) – the only task is to represent queries in such a way that they will match well to those documents that the user will judge as relevant. The document-oriented view has been first proposed by Maron and Kuhn in [68]. Here, documents are indexed by terms carefully chosen such that they will match well to the query to fulfill the user needs. Recently, the language modelling approach to information retrieval (e.g., [78]) builds upon the same document-oriented view. In the basic language models, a unigram model is estimated for each document and the likelihood of the document model with respect to the query is computed. Many variations and extensions have been proposed (e.g., [39, 55, 126]). The concept of relevance has been integrated into the language modelling approach to information retrieval by [58]. Several recent publications have further investigated the relationship between the RSJ models and the language modelling approach to information retrieval [56, 84].

The two views rely on fixing one variable and optimizing the other variable,

i.e., fixing the query and tuning the document or the other way around [82]. [82] has pointed out the drawback that neither view represents the problem of information retrieval completely. It seems intuitively desirable to treat both the query and the document as variables, and to optimize both. Robertson et al. [85] have illustrated this unified view to text retrieval using Fig. 6.1 (a). The authors identified three types of information that can be used for retrieval systems: 1) data describing the relations between other queries in the same class (or in other words similar queries) and this particular document, i.e. marginal information about the column; 2) data describing the relations between this particular query and other documents (similar documents), i.e. marginal information about the row; and 3) data describing the relations between other (similar) queries and other (similar) documents, i.e. the joint information about columns and rows.

The first literature proposing a unified model of information retrieval has been of a theoretical nature only [85, 82]. A first implementation of these ideas is found in [7], where learning methods have been introduced into the vector space model to incorporate relevance information. Hereto, Multi-Dimensional Scaling (MDS) has been adopted to re-position the document vectors and the query vectors such that document representations are moved closer to queries when their relevance is observed, and away from queries when their non-relevance is observed. Bodoff and Robertson [8] modeled the joint distribution of the observed documents, queries and relevances by assuming three underlying stochastic processes in the data generations: 1) the observed documents are generated by the true hidden document models, (2) the observed queries are generated by the true hidden query models, and (3) the relevances are generated by both the document and query models. A Maximum Likelihood (ML) method was applied to estimates the model parameters.

Going back to the collaborative filtering problem, Fig 6.1 (b) illustrates an analogy between the concepts of users and items in CF and the concepts of documents and queries in information retrieval. Like [85] did for documents and queries in information retrieval, [116] identified three types of information embedded in the user-item matrix (see also Section 6.3.3): 1) ratings of the same item by other users; 2) ratings of different items made by the same user, and, 3) ratings of other (but similar) items made by other (but similar) users. This paper explores deeper the usage of these three types of information in a unified probabilistic framework.

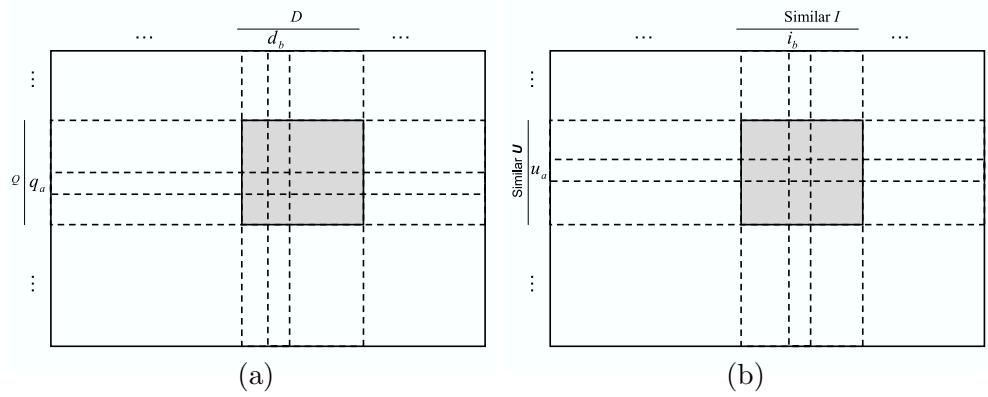


Figure 6.1: Illustration of the joint data in text retrieval and collaborative filtering. (a) Query-document joint data in text retrieval (reproduced from [85]). (b) User-item joint data in collaborative filtering.

6.3 Background

This section introduces briefly the user- and item-based approaches to collaborative filtering [37, 92]. For A users and B items, the user profiles are represented in a $A \times B$ user-item matrix \mathbf{X} (Fig. 6.2(a)). Each element $x_{a,b} = r$ indicates that user a rated item b by r , where r is an integer ≥ 0 and $|R|$ is the number of rating scales. If the item has been rated, and elements $x_{a,b} = \emptyset$ indicate that the rating is unknown.

The user-item matrix can be decomposed into row vectors,

$$\begin{aligned}\mathbf{X} &= [\mathbf{u}_1, \dots, \mathbf{u}_A]^T \\ \mathbf{u}_a &= [x_{a,1}, \dots, x_{a,B}]^T, \quad a = 1, \dots, A\end{aligned}$$

corresponding to the A user profiles \mathbf{u}_a . Each row vector represents the item ratings of a particular user. As discussed below, this decomposition leads to user-based collaborative filtering.

Alternatively, the matrix can be represented by its column vectors,

$$\begin{aligned}\mathbf{X} &= [\mathbf{i}_1, \dots, \mathbf{i}_B] \\ \mathbf{i}_b &= [x_{1,b}, \dots, x_{A,b}]^T, \quad b = 1, \dots, B\end{aligned}$$

corresponding to the ratings by all A users for a specific item b . As will be shown, this representation results in item-based recommendation algorithms.

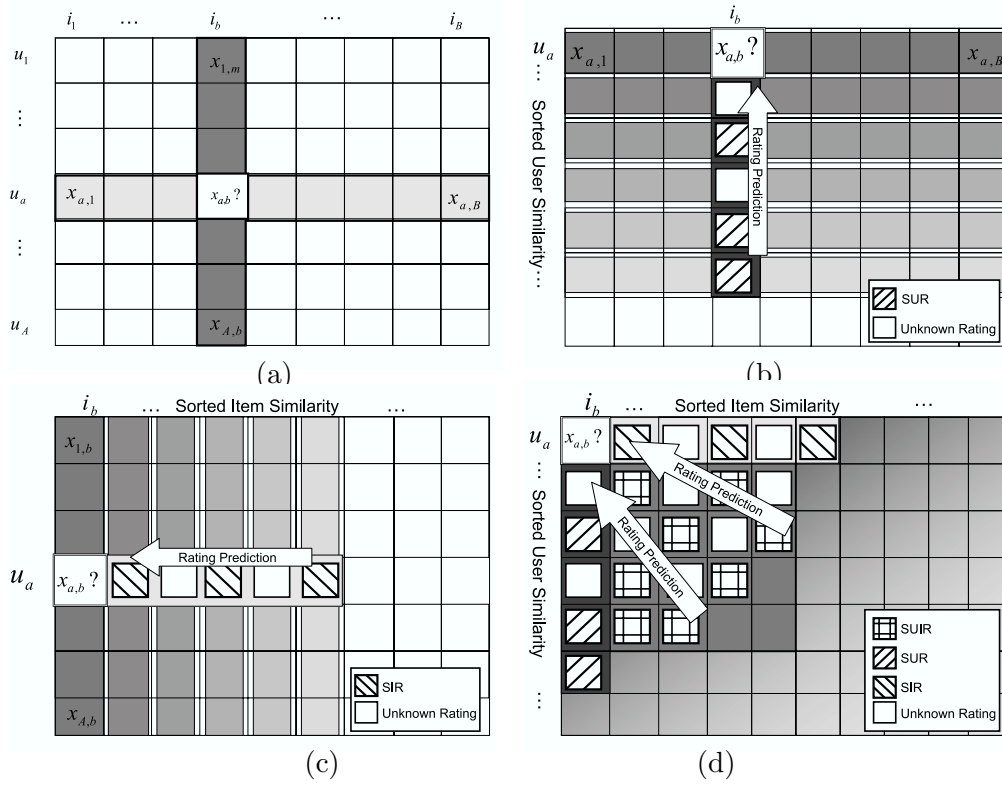


Figure 6.2: (a) The user-item matrix (b) User-based approaches (c) Item-based approaches (d) Combining user-based and item-based approaches.

6.3.1 User-based Collaborative Filtering

User-based collaborative filtering predicts a test user's interest on a test item based on the ratings of *this* test item from other similar users. Ratings by more similar users contribute more to predicting the test item rating. The set of similar users can be identified by employing a threshold on the similarity measure or just selecting the top- N most similar users. In the top- N case, a set of top- N users similar to test user a , $\mathcal{T}_{\mathbf{u}}(\mathbf{u}_a)$, can be generated by ranking users in order of their similarities:

$$\mathcal{T}_{\mathbf{u}}(\mathbf{u}_a) = \{\mathbf{u}_c | \text{rank } s_{\mathbf{u}}(\mathbf{u}_a, \mathbf{u}_c) < N\}, \quad (6.1)$$

where $s_{\mathbf{u}}(\mathbf{u}_a, \mathbf{u}_c)$ is the similarity between users a and c . Cosine similarity and Pearson Correlation are popular similarity measures in collaborative filtering, see e.g. [9]. The similarity could also be learnt from training data [13, 48]. Notice that the formulation in Eq. 6.1 assumes that $x_{c,b} = \emptyset \implies s_{\mathbf{u}}(\mathbf{u}_a, \mathbf{u}_c) = 0$, i.e., the users that did not rate the test item b are not part of the top- N similar users $\mathcal{T}_{\mathbf{u}}(\mathbf{u}_a)$.

Consequently, the predicted rating $\hat{x}_{a,b}$ of test item b by test user a is computed as

$$\hat{x}_{a,b} = \bar{x}_a + \frac{\sum_{\mathbf{u}_c \in \mathcal{T}_{\mathbf{u}}(\mathbf{u}_a)} s_{\mathbf{u}}(\mathbf{u}_a, \mathbf{u}_c) \cdot (x_{c,b} - \bar{x}_c)}{\sum_{\mathbf{u}_c \in \mathcal{T}_{\mathbf{u}}(\mathbf{u}_a)} s_{\mathbf{u}}(\mathbf{u}_a, \mathbf{u}_c)}, \quad (6.2)$$

where \bar{x}_a and \bar{x}_c denote the average rating on all the rated items, made by users a and c , respectively.

Existing methods differ in their treatment of unknown ratings from similar users (i.e. the value $x_{c,b} = \emptyset$). Missing ratings can be replaced by a 0 score, which lowers the prediction, or the average rating of that similar user could be used (mean imputation) [9, 37]. Alternatively, [123] replaces missing ratings by an interpolation of the user's average rating and the average rating of his or her cluster (k-NN imputation).

Fig. 6.2(b) illustrates the user-based approach. In the figure, each user profile (row vector) is sorted and re-indexed by its similarity towards the test user's profile. Notice in Eq. 6.2 and in Fig. 6.2(b) user-based collaborative filtering takes only a small proportion of the user-item matrix into consideration for recommendation, i.e. only the *known test item ratings by similar users* are used. We refer to these ratings as the set of 'similar user ratings' (the blocks with upward diagonal pattern in Fig. 6.2(b)): $SUR_{a,b} = \{x_{c,b} | \mathbf{u}_c \in \mathcal{T}_{\mathbf{u}}(\mathbf{u}_a)\}$. For readability, we drop the subscript a, b of $SUR_{a,b}$ in the remainder of the paper.

6.3.2 Item-based Collaborative Filtering

Item-based approaches such as [23, 61, 92] apply the same idea using the similarity between items instead of users. The unknown rating of a test item by a test user is predicted by averaging the ratings of other similar items rated by *this* test user. Ratings from more similar items are weighted stronger. Formally,

$$\hat{x}_{a,b} = \frac{\sum_{\mathbf{i}_d \in \mathcal{T}_{\mathbf{i}}(\mathbf{i}_b)} s_{\mathbf{i}}(\mathbf{i}_b, \mathbf{i}_d) \cdot x_{a,d}}{\sum_{\mathbf{i}_d \in \mathcal{T}_{\mathbf{i}}(\mathbf{i}_b)} s_{\mathbf{i}}(\mathbf{i}_b, \mathbf{i}_d)}, \quad (6.3)$$

where item similarity $s_{\mathbf{i}}(\mathbf{i}_b, \mathbf{i}_d)$ can be approximated by the cosine measure or Pearson correlation [61, 92]. To remove the difference in rating scale between users when computing the similarity, [92] has proposed to adjust the cosine

similarity by subtracting the user's average rating from each co-rated pair beforehand. Like the top- N similar users, a set of top- N similar items towards test item b , denoted as $\mathcal{T}_i(\mathbf{i}_b)$, can be generated according to:

$$\mathcal{T}_i(\mathbf{i}_b) = \{\mathbf{i}_d | \text{rank } s_i(\mathbf{i}_b, \mathbf{i}_d) < N\} \quad (6.4)$$

Fig. 6.2(c) illustrates the item-based approaches. Each item (column vector) is sorted and re-indexed according to its similarity towards the test item in the user-item matrix. Eq. 6.3 shows that only the *known similar item ratings by the test user* are taken into account for the prediction. We refer to the ratings used in the item-based approach as the set of ‘similar item ratings’ (the blocks with downward diagonal pattern in Fig. 6.2(c)): $SIR_{a,b} = \{x_{a,d} | \mathbf{i}_d \in \mathcal{T}_i(\mathbf{i}_b)\}$. Again, for simplicity, we drop the subscript a, b of $SIR_{a,b}$ in the remainder of the paper.

6.3.3 Combining User-based and Item-based Approaches

When the ratings from these two sources are quite often not available, predictions are often made by averaging ratings from ‘not-so-similar’ users or items. Therefore, relying on *SUR* or *SIR* ratings only is undesirable.

In order to improve the accuracy of prediction, [116] proposed to combine both user-based and item-based approaches. Additionally, we pointed out that the user-item matrix contains useful data beyond the previously used *SUR* and *SIR* ratings. As illustrated in Fig. 6.2 (d), the *similar item ratings made by similar users* may provide an extra source for prediction. They are obtained by sorting and re-indexing rows and columns according to their similarities towards the test user and the test item respectively. In the remainder, this part of the matrix is referred to as ‘similar user item ratings’ (the grid blocks in Fig. 6.2(d)): $SUIR_{a,b} = \{x_{c,d} | \mathbf{u}_c \in \mathcal{T}_u(\mathbf{u}_a), \mathbf{i}_d \in \mathcal{T}_i(\mathbf{i}_b), c \neq a, d \neq b\}$.

The subscript a, b of $SUIR_{a,b}$ is dropped. Their similarity towards the target rating $x_{a,b}$, denoted as $s_{ui}(x_{a,b}, x_{c,d})$, can be calculated as follows:

$$s_{ui}(x_{a,b}, x_{c,d}) = \frac{1}{\sqrt{(1/s_u(\mathbf{u}_a, \mathbf{u}_c))^2 + (1/s_i(\mathbf{i}_b, \mathbf{i}_d))^2}} \quad (6.5)$$

where a Euclidean dis-similarity space is adopted such that the resulting combined similarity is lower than either of them. Now we are ready to combine these three types of ratings in a single collaborative filtering method. We treat each element of the user-item matrix as a separate predictor. Its confidence for the prediction is then estimated based upon its similarity towards the test

rating. We then predict the test rating by averaging the individual predictions weighted by their confidence. Formally,

$$\hat{x}_{a,b} = \sum_{x_{c,d}} p_{a,b}(x_{c,d}) W_{a,b}^{c,d}, \quad (6.6)$$

where

$$p_{a,b}(x_{c,d}) = x_{c,d} - (\bar{x}_c - \bar{x}_a) - (\bar{x}_d - \bar{x}_b) \quad (6.7)$$

can be treated as a normalization function when predicting rating $x_{a,b}$ from rating $x_{c,d}$. \bar{x}_a and \bar{x}_c are the average ratings by user a and c , and \bar{x}_b and \bar{x}_d are the average ratings of item b and d . For each test rating $x_{a,b}$, $W_{a,b}^{c,d}$ acts as a unified weight matrix to combine the predictors from the three different sources:

$$W_{a,b}^{c,d} = \begin{cases} \frac{\sum_{x_{c,d} \in SUR} s_{\mathbf{u}}(\mathbf{u}_a, \mathbf{u}_c)}{s_{\mathbf{u}}(\mathbf{u}_a, \mathbf{u}_c)} \lambda (1 - \delta) & x_{c,d} \in SUR \\ \frac{\sum_{x_{c,d} \in SIR} s_{\mathbf{i}}(\mathbf{i}_b, \mathbf{i}_d)}{s_{\mathbf{i}}(\mathbf{i}_b, \mathbf{i}_d)} (1 - \lambda) (1 - \delta) & x_{c,d} \in SIR \\ \frac{\sum_{x_{c,d} \in SUIR} s_{\mathbf{ui}}(x_{a,b}, x_{c,d})}{s_{\mathbf{ui}}(x_{a,b}, x_{c,d})} \delta & x_{c,d} \in SUIR \\ 0 & otherwise \end{cases}, \quad (6.8)$$

where $\lambda \in [0, 1]$ and $\delta \in [0, 1]$ control the importance of the different rating sources. This combination of ratings can be considered as two subsequent steps of linear interpolation. First, predictions from *SUR* ratings are interpolated with *SIR* ratings, controlled by λ . Next, the intermediate prediction is interpolated with predictions from the *SUIR* data, controlled by δ . The second interpolation corresponds to smoothing the *SIR* and *SUR* predictions with *SUIR* ratings as a background model.

A bigger λ emphasizes user correlations, while smaller λ emphasizes item correlations. When λ equals one, the algorithm corresponds to a user-based approach, while λ equal to zero results in an item-based approach.

Tuning parameter δ controls the impact of smoothing from the background model (i.e. *SUIR*). When δ approaches zero, the fusion framework becomes the mere combination of user-based and item-based approaches without smoothing from the background model.

6.4 Probabilistic Relevance Prediction Models

This section re-formulates the collaborative filtering problem in a probabilistic framework. Motivated by the probabilistic relevance models proposed in text

retrieval domain [86, 56, 78], we introduce the concept of “relevance” into collaborative filtering. By analogy with the relevance models in text retrieval, the collaborative filtering problem can be solved by answering the following basic question: what is the probability that *this* item is relevant to *this* user, given his or her profile?

To answer this question, firstly, let us define a sample space of relevance: Φ_R and let R be a random variable over the relevance space Φ_R . Unlike the commonly used binary relevance in text retrieval, in our case here, the relevance is multiple-valued. Thus, let us define that Φ_R has multiple values, which are observed from the rating data: $\Phi_R = \{1, \dots, |R|\}$, where $|R|$ is the number of rating scales. From now on, each known element ($x_{a,b} \neq \emptyset$) in the user-item matrix is treated as an observation of this multiple-scaled relevance. Secondly, let U be a discrete random variable over the sample space of *user id*’s: $\Phi_U = \{u_1, \dots, u_A\}$ and let I be a random variable over the sample space of *item id*’s: $\Phi_I = \{i_1, \dots, i_B\}$, where A is the number of users and B the number of items in the collection. In other words, U refers to the user identifiers and I refers to the item identifiers.

We then denote P as a probability function on the joint sample space $\Phi_U \times \Phi_I \times \Phi_R$. A prediction model thus can be built by estimating the conditional probability of the rating $P(R|U, I)$, given the specified user and item identifiers. The expectation of the unknown rating of a given item $I = i_b$ from a given user $U = u_a$ can be formulated as follows:

$$\hat{x}_{a,b} = \sum_{r=1}^{|R|} r P(R = r | I = i_b, U = u_a) \quad (6.9)$$

For simplicity, $R = r$, $U = u_a$, and $I = i_b$ are denoted as r , u_a , and i_b , respectively.

Before proceeding, let us highlight how the current formulation using ratings differs from the probabilistic model that we have proposed before for log-based CF [114]. In the log-based model, relevance variable r is binary, observed as “the file is (not) downloaded” or the “web-site is (not) visited”. In that case, Φ_R has binary values ‘relevant’ r and ‘non-relevant’ \bar{r} . The resulting model is a *ranking model*, and does not attempt to predict the rating. Conversely, the model proposed in *this* paper is a rating prediction model, that is especially targeted to rating-based CF; i.e., the model predicts directly the number of stars that a user would assign to the test item.

6.4.1 Three Different Relevance Models

The way to estimate $P(r|i_b, u_a)$ plays an important role in our model. Three different models, namely *user-based relevance*, *item-based relevance* and *unified*

relevance models can be derived if we apply the Bayes' rule differently:

$$P(r|i_b, u_a) = \begin{cases} \frac{P(u_a|r, i_b)P(r|i_b)}{P(u_a|i_b)} & \text{User-based Relevance} \\ \frac{P(i_b|r, u_a)P(r|u_a)}{P(i_b|u_a)} & \text{Item-based Relevance} \\ \frac{P(u_a, i_b|r)P(r)}{P(i_b, u_a)} & \text{Unified Relevance} \end{cases} \quad (6.10)$$

Each of the three derivations represents a different view of the problem. We shall see that the former two models (i.e. the user-based relevance model and the item-based relevance model) represent a partial view of the collaborative filtering problem while the third model unifies these partial views.

6.4.1.1 User-based Relevance Model

Applying the first factorization in Eq. 6.9, we derive:

$$\begin{aligned} \hat{x}_{a,b} &= \sum_{r=1}^{|R|} r \frac{P(u_a|r, i_b)P(r|i_b)}{P(u_a|i_b)} \\ &= \frac{\sum_{r=1}^{|R|} r P(u_a|r, i_b)P(r|i_b)}{P(u_a|i_b)} \\ &= \frac{\sum_{r=1}^{|R|} r P(u_a|r, i_b)P(r|i_b)}{\sum_{r=1}^{|R|} P(u_a|r, i_b)P(r|i_b)}, \end{aligned} \quad (6.11)$$

where the final prediction relies on two probabilities: 1) The conditional probability $P(u_a|r, i_b)$ builds up a user space model conditioned on the target item and the rating. It measures how probable a user may rate item i_b as rating r and thus it is the *preference model* of a user. 2) The probability $P(r|i_b)$ measures how likely the target item i_b may be rated as rating r . It can be regarded as a rating *prior* or the *rating model*. Clearly, together the product of the two probabilities serves as a weight for each rating scale r . The denominator, a sum over the different rating scales $\sum_{r=1}^{|R|} P(u_a|r, i_b)P(r|i_b)$, serves as a normalization factor. To simplify the notation, we choose r_i to denote the joint event between r and i (i.e. rating for item i). Hence $P(u_a|r, i_b)$ is written as $P(u_a|r_{i_b})$.

Notice that, in the user preference model ($P(u_a|r, i_b)$) in Eq. 6.11, users are defined by their 'user identifiers' in the *user id* space. To be able to make a prediction for a new user, we need to build a feature representation and use it to relate the new user to the training users. For this, instead of putting users in the original *user id* space, we represent them by their ratings. So, $P(u_a|r_{i_b}) = P(\mathbf{u}_a|r_{i_b})$, where the vector $\mathbf{u}_a = [x_{a,1}, \dots, x_{a,B}]^T$ denotes the B item ratings from user a . Unrated items can be filled with the average rating

value, or taken as a constant value 0 instead. Substituting user identifiers by their rating vectors in Eq. 6.11 gives:

$$\hat{x}_{a,b} = \frac{\sum_{r=1}^{|R|} r P(\mathbf{u}_a | r_{i_b}) P(r | i_b)}{\sum_{r=1}^{|R|} P(\mathbf{u}_a | r_{i_b}) P(r | i_b)} \quad (6.12)$$

6.4.1.2 Item-based Relevance Model

We derive the following equation the same way, by factorizing $P(r | u_a, i_b)$ in Eq. 6.9 as $P(i_b | u_a, r) P(r | u_a) / P(i_b | u_a)$:

$$\hat{x}_{a,b} = \frac{\sum_{r=1}^{|R|} r P(i_b | r, u_a) P(r | u_a)}{\sum_{r=1}^{|R|} P(i_b | r, u_a) P(r | u_a)} \quad (6.13)$$

To simplify notation, let r_u denote the joint event between r and u (i.e., the rating from user u), and $P(i_b | r, u_a)$ be written as $P(i_b | r_{u_a})$.

Following the same line of reasoning as for the user case above, an item can be represented using each user's rating as a feature, such that $P(i_b | r_{u_a}) = P(\mathbf{i}_b | r_{u_a})$ where the vector $\mathbf{i}_b = [x_{1,b}, \dots, x_{A,b}]^T$ denotes the A user ratings for item b . Again, the missing ratings can be replaced by the average rating value or by a constant value 0. This gives

$$\hat{x}_{a,b} = \frac{\sum_{r=1}^{|R|} r P(\mathbf{i}_b | r_{u_a}) P(r | u_a)}{\sum_{r=1}^{|R|} P(\mathbf{i}_b | r_{u_a}) P(r | u_a)} \quad (6.14)$$

Probability $P(\mathbf{i}_b | r_{u_a})$ conditions the item space on the user's rating. It expresses how probable an item is rated as value r by user u_a , and can be regarded as the *preference model* of an item. The probability $P(r | u_a)$ measures how likely the target user u_a may provide a rating as value r . It can be treated as a rating *prior*, or, the target user's *rating model*. Obviously, the final weight for each rating scale is a product between these two models. The summation over different rating scales $\sum_{r=1}^{|R|} P(\mathbf{i}_b | r_{u_a}) P(r | u_a)$ in the denominator serves as a normalization factor.

6.4.1.3 Unified Relevance Model

Let us now introduce the unified relevance model. We derive from Eq. 6.9:

$$\hat{x}_{a,b} = \frac{\sum_{r=1}^{|R|} r P(u_a, i_b | r) P(r)}{\sum_{r=1}^{|R|} P(u_a, i_b | r) P(r)} \quad (6.15)$$

Table 6.1: Probabilities in the three different models.

| | Preference Model | Rating Model |
|----------------------|-----------------------------------|--------------|
| User-based Relevance | $P(\mathbf{u}_a r_{i_b})$ | $P(r i_b)$ |
| Item-based Relevance | $P(\mathbf{i}_b r_{u_a})$ | $P(r u_a)$ |
| Unified Relevance | $P(\mathbf{u}_a, \mathbf{i}_b r)$ | $P(r)$ |

Analogous to the two other derivations, we use ratings as features to represent users and items, respectively: $P(u_a, i_b|r) = P(\mathbf{u}_a, \mathbf{i}_b|r)$.

$$\hat{x}_{a,b} = \frac{\sum_{r=1}^{|R|} r P(\mathbf{u}_a, \mathbf{i}_b|r) P(r)}{\sum_{r=1}^{|R|} P(\mathbf{u}_a, \mathbf{i}_b|r) P(r)} \quad (6.16)$$

The preference model now involves the user and item together.

Table 6.1 summarises the three different models derived from Eq. 6.9 (Eq. 6.12, 6.14 and 6.16). The weights to average the rating scales equal the product of a preference model and a rating model. Since the three models are derived from the same root, they are probabilistically equivalent, but the different factorizations lead to the different probability estimations, so statistically they are inequivalent.

6.4.2 Probability Estimation

The next problem is how to estimate the probabilities listed in Table 6.1. Because the rating space is one-dimensional, the densities for the rating models can be estimated by simply counting the frequency of the co-occurrences. For the density estimations of the preference models however, the high dimensionality of the user feature space and the item feature space complicates matters. Here, we use the non-parametric Parzen-window method for density estimation (also known as kernel density estimation). This approach extrapolates the density from the sample data. The main advantage over a parametric approach is that we do not have to specify the model structure a priori, but may determine it from the data itself.

6.4.2.1 Density Estimation for Rating Models

Estimating the three rating models corresponds to counting the co-occurrence frequencies of the three joint events:

$$P(r|u_a) = \frac{\sum_{i_b} c(u_a, i_b, r)}{\sum_{i_b, r} c(u_a, i_b, r)} = \frac{|S_{r_{u_a}}|}{|S_{u_a}|} \quad (6.17a)$$

Table 6.2: Commonly used Parzen kernel functions.

| | |
|-------------------------|--|
| Rectangular | $\frac{1}{2}$ for $ x < 1$, 0 otherwise |
| Triangular | $1 - x $ for $ x < 1$, 0 otherwise |
| Biweight | $\frac{15}{16}(1 - x^2)^2$ for $ x < 1$, 0 otherwise |
| Gaussian | $\frac{1}{\sqrt{2\pi}}e^{-x^2/2}$ |
| Bartlett - Epanechnikov | $\frac{3}{4}(1 - x^2/5)/\sqrt{5}$ for $ x < \sqrt{5}$, 0 otherwise |

$$P(r|i_b) = \frac{\sum_{u_a} c(u_a, i_b, r)}{\sum_{u_a, r} c(u_a, i_b, r)} = \frac{|S_{r_{i_b}}|}{|S_{i_b}|} \quad (6.17b)$$

$$P(r) = \frac{\sum_{u_a, i_b} c(u_a, i_b, r)}{\sum_{u_a, i_b, r} c(u_a, i_b, r)} = \frac{|S_r|}{|S|}, \quad (6.17c)$$

where $c(u_a, i_b, r) \in \{0, 1\}$ denotes the co-occurrence function; $c(u_a, i_b, r)$ equals one if user u_a rated item i_b as r , zero otherwise. $S_{(\cdot)}$ denotes the set of observed samples where event (\cdot) happened, $|S_{(\cdot)}|$ its cardinality. For example, $S_{r_{i_b}}$ denotes the set of observed samples with event $(R = r, I = i_b)$, so $|S_{r_{i_b}}|$ corresponds to the number of times that this event happened, $\sum_{u_a} c(u_a, i_b, r)$. S denotes the entire set of observed samples and $|S|$ its size, equal to $\sum_{u_a, i_b, r} c(u_a, i_b, r)$.

6.4.2.2 Density Estimation for Preference Models

Given some observed samples of a population, the Parzen-window method extrapolates the data to the entire population by averaging from neighborhood observed samples. The neighborhood samples are selected and weighted according to some pre-defined Parzen kernel window functions. Usually, the Parzen kernel function is required to be non-negative and symmetric, and it should integrate to one. Table 6.2 lists the most commonly used Parzen kernels for univariate data. The multi-dimensional feature spaces in which users and items are represented require a multivariate Parzen kernel (denoted as $\mathbf{K}(\mathbf{y})$), that can be obtained using a product of univariate Parzen kernels:

$$\mathbf{K}(\mathbf{y}) = \prod_{q=1}^Q K_q(y_q), \quad (6.18)$$

where $\mathbf{y} = [y_1, \dots, y_Q]$ is a Q -dimensional vector. K_q is the univariate Parzen kernel function for the q th dimension. We assume all univariate kernels to be equal: $K_q = K$. The product of the univariate Parzen kernel assumes that the dimensions (features) are locally independent (where the locality is defined by

the univariate Parzen kernel function K). Plugging in the bandwidth parameter, we have:

$$\frac{1}{h^Q} \mathbf{K}\left(\frac{\mathbf{y}}{h}\right) = \frac{1}{h^Q} \mathbf{K}\left(\frac{y_1}{h}, \dots, \frac{y_Q}{h}\right) = \frac{1}{h^Q} \prod_{q=1}^Q K\left(\frac{y_q}{h}\right), \quad (6.19)$$

where to reduce the complexity of the model, we have assumed the bandwidth of each Parzen window to be equal in each dimension, defined as h .

First, take preference models $P(\mathbf{u}|r_{i_b})$ and $P(\mathbf{i}|r_{u_a})$ in the user-based and the item-based relevance derivations. Both depend on a single feature vector; either the user vector \mathbf{u} or the item vector \mathbf{i} . Directly applying the Parzen-window method by using the univariate product Parzen kernel, we obtain the following equations for the density estimations ([25]):

$$P(\mathbf{u}|r_{i_b}) = \frac{1}{|S_{r_{i_b}}|} \sum_{\mathbf{u}' \in S_{r_{i_b}}} \frac{1}{h_u^B} \mathbf{K}\left(\frac{\mathbf{u} - \mathbf{u}'}{h_u}\right) \quad (6.20a)$$

$$P(\mathbf{i}|r_{u_a}) = \frac{1}{|S_{r_{u_a}}|} \sum_{\mathbf{i}' \in S_{r_{u_a}}} \frac{1}{h_i^A} \mathbf{K}\left(\frac{\mathbf{i} - \mathbf{i}'}{h_i}\right), \quad (6.20b)$$

where h_i and h_u are the bandwidth window parameters for the user vector and item vector, respectively.

Next, consider the preference model in the unified relevance case, $P(\mathbf{u}_a, \mathbf{i}_b|r)$. Probability estimation requires a density in the joint user-item feature space. We employ a product of two univariate kernel density estimators:

$$P(\mathbf{u}_a, \mathbf{i}_b|r) = \frac{1}{|S_r|} \sum_{(\mathbf{u}', \mathbf{i}') \in S_r} \frac{1}{h_u^B} \mathbf{K}\left(\frac{\mathbf{u} - \mathbf{u}'}{h_u}\right) \frac{1}{h_i^A} \mathbf{K}\left(\frac{\mathbf{i} - \mathbf{i}'}{h_i}\right), \quad (6.21)$$

where S_r denotes the set of observed samples when event $(R = r)$ happens, while $|S_r|$ denotes its size, equal to $\sum_{u_a, i_b} c(u_a, i_b, r)$.

The kernel density estimation can be interpreted as a mixture of the component densities with an equal weight. Each mean of the component densities is located in the place where the neighborhood observation is available. The shape of the density is controlled by the Parzen kernel window function and its bandwidth parameter. Thus, the final prediction can be expressed by the summation over the Parzen kernels which are situated in the location of the neighborhood samples.

6.4.3 Rating Predictions

Consider now the problem of rating prediction in the user-based relevance model. Substituting the user-based rating model of Eq. 6.17a and the user preference model of Eq. 6.20a in the generic user-based relevance model of Eq. 6.12 gives:

$$\hat{x}_{a,b} = \frac{\sum_{r=1}^{|R|} r \frac{1}{|S_{r_{i_b}}|} \left(\sum_{\mathbf{u}' \in S_{r_{i_b}}} \frac{1}{h_u^B} \mathbf{K}\left(\frac{\mathbf{u}-\mathbf{u}'}{h_u}\right) \right) \frac{|S_{r_{i_b}}|}{|S_{i_b}|}}{\sum_{r=1}^{|R|} \frac{1}{|S_{r_{i_b}}|} \left(\sum_{\mathbf{u}' \in S_{r_{i_b}}} \frac{1}{h_u^B} \mathbf{K}\left(\frac{\mathbf{u}-\mathbf{u}'}{h_u}\right) \right) \frac{|S_{r_{i_b}}|}{|S_{i_b}|}} \quad (6.22)$$

Cancelling out the factors $|S_{r_{i_b}}|$ and $|S_{i_b}|$ simplifies Eq. 6.22 to:

$$\hat{x}_{a,b} = \frac{\sum_{r=1}^{|R|} r \left(\sum_{\mathbf{u}' \in S_{r_{i_b}}} \mathbf{K}\left(\frac{\mathbf{u}-\mathbf{u}'}{h_u}\right) \right)}{\sum_{r=1}^{|R|} \left(\sum_{\mathbf{u}' \in S_{r_{i_b}}} \mathbf{K}\left(\frac{\mathbf{u}-\mathbf{u}'}{h_u}\right) \right)} \quad (6.23)$$

Combining the outer summation over r with the inner over \mathbf{u}' completes our approach to perform rating prediction using the user-based relevance model:

$$\hat{x}_{a,b} = \frac{\sum_{\mathbf{u}' \in S_{i_b}} r_{\mathbf{u}', i_b} \mathbf{K}\left(\frac{\mathbf{u}-\mathbf{u}'}{h_u}\right)}{\sum_{\mathbf{u}' \in S_{i_b}} \mathbf{K}\left(\frac{\mathbf{u}-\mathbf{u}'}{h_u}\right)}, \quad (6.24)$$

where $\mathbf{u}' \in S_{i_b}$ denotes the (other) users who have rated item i_b , and, $r_{\mathbf{u}', i_b}$ denotes the rating of user \mathbf{u}' for item i_b .

Rating prediction in the item-based relevance model is derived analogously:

$$\hat{x}_{a,b} = \frac{\sum_{\mathbf{i}' \in S_{u_a}} r_{\mathbf{i}', u_a} \mathbf{K}\left(\frac{\mathbf{i}-\mathbf{i}'}{h_i}\right)}{\sum_{\mathbf{i}' \in S_{u_a}} \mathbf{K}\left(\frac{\mathbf{i}-\mathbf{i}'}{h_i}\right)}, \quad (6.25)$$

where $\mathbf{i}' \in S_{u_a}$ denotes the other items rated by user u_a , and, $r_{\mathbf{i}', u_a}$ denotes the rating of user \mathbf{i}' for item u_a .

Finally, rating prediction with the unified relevance model:

$$\hat{x}_{a,b} = \frac{\sum_{(\mathbf{u}', \mathbf{i}') \in S} r_{\mathbf{u}', \mathbf{i}'} \mathbf{K}\left(\frac{\mathbf{u}-\mathbf{u}'}{h_u}\right) \mathbf{K}\left(\frac{\mathbf{i}-\mathbf{i}'}{h_i}\right)}{\sum_{(\mathbf{u}', \mathbf{i}') \in S} \mathbf{K}\left(\frac{\mathbf{u}-\mathbf{u}'}{h_u}\right) \mathbf{K}\left(\frac{\mathbf{i}-\mathbf{i}'}{h_i}\right)}, \quad (6.26)$$

where $r_{\mathbf{u}', \mathbf{i}'}$ denotes the rating of user \mathbf{u}' for item \mathbf{i}' . Again, S denotes the entire sample set $S = S_1 \cup \dots \cup S_{|R|}$.

The next step specifies the Parzen-window kernel function K and its bandwidth parameter h . A common choice is to use the Gaussian density function as the univariate Parzen kernel function:

$$\frac{1}{h^Q} \mathbf{K}\left(\frac{\mathbf{y}}{h}\right) = \frac{1}{(\sqrt{2\pi}h)^Q} e^{-\frac{\|\mathbf{y}\|^2}{2h^2}} = \frac{1}{(\sqrt{2\pi}h)^Q} \prod_q e^{-\frac{\|y_q\|^2}{2h^2}} \quad (6.27)$$

Substituting the Gaussian Parzen-window in the above equations results in the following preference models:

$$\begin{aligned} P(\mathbf{u}_a | r_{i_b}) &= \frac{1}{|S_{r_{i_b}}|(\sqrt{2\pi}h_u)^B} \sum_{\mathbf{u}' \in S_{r_{i_b}}} e^{-\frac{\|\mathbf{u}-\mathbf{u}'\|^2}{2h_u^2}} \\ P(\mathbf{i} | r_{u_a}) &= \frac{1}{|S_{r_{u_a}}|(\sqrt{2\pi}h_i)^A} \sum_{\mathbf{i}' \in S_{r_{u_a}}} e^{-\frac{\|\mathbf{i}-\mathbf{i}'\|^2}{2h_i^2}} \\ P(\mathbf{u}_a, \mathbf{i}_b | r) &= \frac{1}{|S_r|} \sum_{(\mathbf{u}', \mathbf{i}') \in S_r} \frac{1}{(\sqrt{2\pi}h_u)^B (\sqrt{2\pi}h_i)^A} e^{-\frac{\|\mathbf{u}-\mathbf{u}'\|^2}{2h_u^2}} e^{-\frac{\|\mathbf{i}-\mathbf{i}'\|^2}{2h_i^2}} \end{aligned} \quad (6.28)$$

Using these three probability estimates gives the final three equations for rating prediction, corresponding to the three rating prediction models proposed in this paper:

User-based Relevance Model:

$$\hat{x}_{a,b} = \frac{\sum_{\mathbf{u}' \in S_{i_b}} r_{\mathbf{u}', i_b} \cdot e^{-\frac{\|\mathbf{u}-\mathbf{u}'\|^2}{2h_u^2}}}{\sum_{\mathbf{u}' \in S_{i_b}} e^{-\frac{\|\mathbf{u}-\mathbf{u}'\|^2}{2h_u^2}}} \quad (6.29a)$$

Item-based Relevance Model:

$$\hat{x}_{a,b} = \frac{\sum_{\mathbf{i}' \in S_{u_a}} r_{\mathbf{i}', u_a} \cdot e^{-\frac{\|\mathbf{i}-\mathbf{i}'\|^2}{2h_i^2}}}{\sum_{\mathbf{i}' \in S_{u_a}} e^{-\frac{\|\mathbf{i}-\mathbf{i}'\|^2}{2h_i^2}}} \quad (6.29b)$$

Unified Relevance Model:

$$\hat{x}_{a,b} = \frac{\sum_{(\mathbf{u}', \mathbf{i}') \in S} r_{\mathbf{u}', \mathbf{i}'} \cdot e^{-\frac{\|\mathbf{u}-\mathbf{u}'\|^2}{2h_u^2}} e^{-\frac{\|\mathbf{i}-\mathbf{i}'\|^2}{2h_i^2}}}{\sum_{(\mathbf{u}', \mathbf{i}') \in S} e^{-\frac{\|\mathbf{u}-\mathbf{u}'\|^2}{2h_u^2}} e^{-\frac{\|\mathbf{i}-\mathbf{i}'\|^2}{2h_i^2}}} \quad (6.29c)$$

6.4.4 Cross-validated EM algorithm

Previous studies have shown that the type of Parzen kernel function has usually only a marginal effect on the quality of density estimation; however, the choice of the bandwidth window parameter h has a significant influence [100]. In our case, the two bandwidth window parameters h_u and h_i need to be tuned to the data. If their value is too small, the estimated density is a collection of sharp peaks positioned at the sample points, such that the density estimation still suffers from the data sparsity. If their value is however too large, the density estimate is over-smoothed, such that the underlying structure in the data is not preserved in the estimated density.

The optimal bandwidth parameters can be found by maximising the following cross-validated (leave the estimated \mathbf{u} or \mathbf{i} out) likelihood function [26]:

$$\hat{h}_u, \hat{h}_i = \arg \max_{h_u, h_i} \prod_{(\mathbf{u}, \mathbf{i}) \in S_r} \frac{1}{|S_r - 1|} \sum_{(\mathbf{u}', \mathbf{i}') \in S_r \wedge (\mathbf{u}' \neq \mathbf{u}) \wedge (\mathbf{i}' \neq \mathbf{i})} \frac{1}{h_u^B} \mathbf{K}\left(\frac{\mathbf{u} - \mathbf{u}'}{h_u}\right) \frac{1}{h_i^A} \mathbf{K}\left(\frac{\mathbf{i} - \mathbf{i}'}{h_i}\right) \quad (6.30)$$

This maximisation problem can be solved using the iterative Expectation maximisation (EM) algorithm [22, 75]. For readability, we give the final expectation (E) and maximisation (M) steps here but leave the detailed derivation for Appendix 6.A:

- E step:

$$P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) = \frac{e^{-\frac{\|\mathbf{u} - \mathbf{u}'\|^2}{2h_u^2}} e^{-\frac{\|\mathbf{i} - \mathbf{i}'\|^2}{2h_i^2}}}{\sum_{\mathbf{u}' \neq \mathbf{u}, \mathbf{i}' \neq \mathbf{i}} e^{-\frac{\|\mathbf{u} - \mathbf{u}'\|^2}{2h_u^2}} e^{-\frac{\|\mathbf{i} - \mathbf{i}'\|^2}{2h_i^2}}} \quad (6.31a)$$

- M step:

$$h_u^{(t+1)} = \sqrt{\frac{1}{B|S_r|} \sum_{\mathbf{u}, \mathbf{i}} \sum_{\mathbf{u}' \neq \mathbf{u}, \mathbf{i}' \neq \mathbf{i}} P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) (\|\mathbf{u} - \mathbf{u}'\|^2)} \quad (6.31b)$$

$$h_i^{(t+1)} = \sqrt{\frac{1}{A|S_r|} \sum_{\mathbf{u}, \mathbf{i}} \sum_{\mathbf{u}' \neq \mathbf{u}, \mathbf{i}' \neq \mathbf{i}} P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) (\|\mathbf{i} - \mathbf{i}'\|^2)} \quad (6.31c)$$

where t equals the number of the iteration.

6.4.5 A Generalised Distance Measure

The univariate Gaussian Parzen kernel used in the previous measures the distance between users and item using Euclidean or L_2 distance ($\|\mathbf{u} - \mathbf{u}'\|^2$ and

$\|\mathbf{i} - \mathbf{i}'\|^2$). However, many alternative distances could be considered. A previous study [36] shows that the mean-squared difference is less effective for collaborative filtering than Pearson correlation and the cosine measure. We could of course adapt our framework using a different Parzen-window function, and try to set things up such that the density estimation is based on the cosine measure instead of Euclidean distance. Ideally, we would however like to *generalise* the Parzen-window density estimation from the specific distance measure of our choice.

We can achieve this goal using the mathematics that has become known as the *kernel trick* in the machine learning community [96]. Notice that the term kernel will refer to a *different* kernel function than the one in the Parzen-window density estimation; to avoid confusion, we use the term *projection kernel*. The kernel trick transforms any algorithm that solely depends on the dot product between two vectors, by replacing this dot product with the application of the projection kernel. It has been proven (based on the Moore-Aronszajn-Theorem) that a positive definite projection kernel determines a unique function ϕ such that:

$$\mathcal{K}(\mathbf{y}, \mathbf{y}') = \langle \phi(\mathbf{y}), \phi(\mathbf{y}') \rangle, \quad (6.32)$$

where \langle, \rangle denotes the dot product. This equation is often used without knowing the exact form of function ϕ ; it suffices to know that it exists and is defined uniquely. Schölkopf has shown that an even larger class of projection kernels (referred to as conditionally positive definite functions) satisfies Eq. 6.32 [95].

Now, basic linear algebra allows us to relate the Euclidean distance in projected space to the application of a projection kernel in user- or item-space:

$$\begin{aligned} & \|\phi(\mathbf{y}) - \phi(\mathbf{y}')\|^2 \\ &= \sum_i (\phi(y_i) - \phi(y'_i))^2 \\ &= \sum_i \{\phi(y_i)^2 - 2\phi(y_i)\phi(y'_i) + \phi(y'_i)^2\} \\ &= \sum_i \phi(y_i)^2 + \sum_i \phi(y'_i)^2 - 2 \sum_i \phi(y_i)\phi(y'_i) \\ &= \langle \phi(\mathbf{y}), \phi(\mathbf{y}) \rangle + \langle \phi(\mathbf{y}'), \phi(\mathbf{y}') \rangle - 2 \langle \phi(\mathbf{y}), \phi(\mathbf{y}') \rangle \\ &= \mathcal{K}(\mathbf{y}, \mathbf{y}) + \mathcal{K}(\mathbf{y}', \mathbf{y}') - 2\mathcal{K}(\mathbf{y}, \mathbf{y}') \end{aligned} \quad (6.33)$$

Using a length-normalised projection kernel for which $\mathcal{K}(\mathbf{y}, \mathbf{y}) = 1$ gives

$$\|\phi(\mathbf{y}) - \phi(\mathbf{y}')\| = \mathcal{K}(\mathbf{y}, \mathbf{y}) + \mathcal{K}(\mathbf{y}', \mathbf{y}') - 2\mathcal{K}(\mathbf{y}, \mathbf{y}') = 2 - 2\mathcal{K}(\mathbf{y}, \mathbf{y}') \quad (6.34)$$

So, computing a Euclidean distance in projected space is equivalent to using a positive definite projection kernel \mathcal{K} to compute distances in the original space.

This property allows us to perform Parzen-window density estimation with the Gaussian kernel in the projected space, without actually knowing the function ϕ (which is however defined uniquely by the choice of the projection kernel).

In the remainder, we use the length-normalised dot product as the projection kernel (also known as the cosine kernel, denoted as $Cos(\mathbf{y}, \mathbf{y}')$ [62]):

$$\mathcal{K}(\mathbf{y}, \mathbf{y}') = Cos(\mathbf{y}, \mathbf{y}') = \frac{\langle \mathbf{y}, \mathbf{y}' \rangle}{\sqrt{\langle \mathbf{y}, \mathbf{y} \rangle \langle \mathbf{y}', \mathbf{y}' \rangle}} \quad (6.35)$$

Thus, we have:

$$\|\phi(\mathbf{y}) - \phi(\mathbf{y}')\|^2 = 2 - 2Cos(\mathbf{y}, \mathbf{y}') \quad (6.36)$$

In this specific case, function ϕ is actually known and corresponds to vector normalization:

$$\phi(\mathbf{y}) = \frac{\mathbf{y}}{\sqrt{\langle \mathbf{y}, \mathbf{y} \rangle}} = \frac{\mathbf{y}}{\sqrt{\sum_i (y_i)^2}} \quad (6.37)$$

Eq. 6.36 demonstrates that the cosine *dissimilarity* measure is indeed equivalent to a Euclidean distance measure in the projected space. So, we can perform Parzen-window density estimation with a Gaussian window function in the projected space:

$$\mathbf{K}_h(\phi(\mathbf{y}), \phi(\mathbf{y}')) = \frac{J}{h} e^{-\frac{\|\phi(\mathbf{y}) - \phi(\mathbf{y}')\|^2}{2h^2}} = \frac{J}{h} e^{-\frac{1 - Cos(\mathbf{y}, \mathbf{y}')}{h^2}}, \quad (6.38)$$

where J is a normalization factor to obtain a Parzen window function, i.e., to satisfy

$$\int_{\mathbf{y}} \frac{J}{h} e^{-\frac{1 - Cos(\mathbf{y}, \mathbf{y}')}{h^2}} d\mathbf{y} = 1 \quad (6.39)$$

It is easy to show that $J \in \mathbb{R}$ since $\frac{1}{h} e^{-\frac{1 - Cos(\mathbf{y}, \mathbf{y}')}{h^2}}$ is bounded in the \mathbf{y} space.

By employing Eq. 6.38 instead of Eq. 6.27, we integrate the cosine similarity measure into our final rating prediction models:

User-based Relevance Model:

$$\hat{x}_{a,b} = \frac{\sum_{\mathbf{u}' \in S_{i_b}} r_{\mathbf{u}', i_b} e^{-\frac{1 - Cos(\mathbf{u}, \mathbf{u}')}{h_u^2}}}{\sum_{\mathbf{u}' \in S_{i_b}} e^{-\frac{1 - Cos(\mathbf{u}, \mathbf{u}')}{h_u^2}}} \quad (6.40a)$$

Table 6.3: Relationship between the choice of Parzen kernel, bandwidth parameters and the projection kernel and CF algorithms.

| Bandwidth Parameters | Parzen Kernel | Projection Kernel | CF Algorithm |
|----------------------|-----------------------|-------------------|--|
| $h_u \in \mathbb{R}$ | Gaussian | Cosine | User-based Relevance Model (Eq. 6.40a) |
| $h_i = \infty$ | Bartlett-Epanechnikov | Null | Eq. 6.29a and PD ([77]) |
| | | Cosine | User-based method, VS ([9]) |
| | | Null | User-based, Ringo ([97]) |
| $h_i \in \mathbb{R}$ | Gaussian | Cosine | Item-based Relevance Model (Eq. 6.40b) |
| $h_u = \infty$ | Bartlett-Epanechnikov | Null | Eq. 6.29b |
| | | Cosine | Item-based method, VS ([92]), |
| | | Null | Item-based Relevance Model, Euclidean Distance |
| $h_u \in \mathbb{R}$ | Gaussian | Cosine | Unified Relevance Model (Eq. 6.40c) |
| $h_i \in \mathbb{R}$ | Bartlett-Epanechnikov | Null | Eq. 6.29c |
| | | Cosine | ([116]) and ([43]) |
| | | Null | Unified Relevance Model, Euclidean Distance |

Item-based Relevance Model:

$$\hat{x}_{a,b} = \frac{\sum_{\mathbf{i}' \in S_{u_a}} r_{\mathbf{i}', u_a} e^{-\frac{1 - \text{Cos}(\mathbf{i}, \mathbf{i}')}{h_i^2}}}{\sum_{\mathbf{i}' \in S_{u_a}} e^{-\frac{1 - \text{Cos}(\mathbf{i}, \mathbf{i}')}{h_i^2}}} \quad (6.40b)$$

Unified Relevance Model:

$$\hat{x}_{a,b} = \frac{\sum_{(\mathbf{u}', \mathbf{i}') \in S} r_{\mathbf{u}', \mathbf{i}'} e^{-\frac{1 - \text{Cos}(\mathbf{u}, \mathbf{u}')}{h_u^2}} e^{-\frac{1 - \text{Cos}(\mathbf{i}, \mathbf{i}')}{h_i^2}}}{\sum_{(\mathbf{u}', \mathbf{i}') \in S} e^{-\frac{1 - \text{Cos}(\mathbf{u}, \mathbf{u}')}{h_u^2}} e^{-\frac{1 - \text{Cos}(\mathbf{i}, \mathbf{i}')}{h_i^2}}} \quad (6.40c)$$

Estimation of parameters h_u and h_i follows the same procedure as in the Euclidean distance case, see Appendix 6.A.

6.4.6 Discussions

The General Framework The proposed combination of Parzen-window kernel density estimation with the relevance models provides a general framework for collaborative filtering. The three rating prediction models listed in Eq. 6.40a, 6.40b and 6.40c, show how the final predictions are expressed by summations over rating influences from the neighborhood samples (from user neighbors Eq. 6.40a, item neighbors Eq. 6.40b, or both the user and item neighbors Eq. 6.40c). Three factors determine the influence of the neighborhood samples on the prediction: the type of Parzen kernel, its bandwidth parameters, and, the distance of the neighborhood samples from the sample to be predicted. The

Parzen kernel smoothes the prediction, while the projection kernel allows us to select the right distance measure. Different choices for the bandwidth parameters, the Parzen-window kernel function or the projection kernel lead to different approaches to collaborative filtering. For instance, it is easy to see that using the Bartlett-Epanechnikov kernel (given in Table 6.2) with h_i equal to one and h_u to ∞ simplifies the unified relevance prediction formula in Eq. 6.26 to the item-based cosine similarity method (using Eq. 6.36):

$$\hat{x}_{a,b} = \frac{\sum_{\mathbf{i}' \in S_{u_a}} r_{\mathbf{i}', u_a} \text{Cos}(\mathbf{i}, \mathbf{i}')}{\sum_{\mathbf{i}' \in S_{u_a}} \text{Cos}(\mathbf{i}, \mathbf{i}')} \quad (6.41)$$

Table 6.3 summarises how the general framework can be specialised to various previously known approaches.

The User-based and Item-Based Views The user-based relevance model and item-based relevance model represent two different views for the problem. In the user-based relevance model, we fix the target item i_b . Conditioned on it, we build up a user representation $P(\mathbf{u}_a | i_b, r)$ (See Fig. 6.3 (a)). This is analogous to the document-oriented approaches in text retrieval [68, 56], where queries represented by the terms are conditioned on the fixed target document model $P(\mathbf{Q} | d_b, r)$.¹ Conversely, in the item-based relevance model, we fix the target user u_a and conditioned on it, we build up an item representation $P(\mathbf{i}_b | u_a, r)$ (see Fig. 6.3 (b)). This is analogous to the query-oriented approach in text retrieval [86], where documents \mathbf{D} are represented by the terms and these representations are conditioned on the fixed query terms $P(\mathbf{D} | q_a, r)$.²

The Unified View Unlike the above two models, the unified relevance model in Eq. 6.40c however provides a rather completed and unified view of the problem. In this model, we do not fix the two variables: user and item. Instead, we construct a unified model that relies on both the user representation and the item representation $P(\mathbf{u}_a, \mathbf{i}_b | r)$ (see Fig. 6.3 (c)). The model is solved by applying the kernel density estimation and it provides a practical solution for the unification advocated in [85]. It can also be treated as a generalised version of the similarity fusion approach to CF [116].

The model intuitively provides a unified probabilistic framework to fuse user-based and item-based approaches. In addition, the ratings from similar users for similar items (the SUIR ratings) are employed to smooth the predictions. We highlight the relationship to the similarity fusion method of [116] by dividing the entire set of observations S into three sets: the ratings from test user $S_{\mathbf{u}}$,

¹In the language modelling literature, the document-oriented approach is also referred to as a *query-generation* model.

²The query-oriented approach is also known as a *document-generation* model for information retrieval.

the ratings for test item S_i , and the remaining ratings $S_{\bar{u},\bar{i}}$. Eq. 6.40c gives

$$\begin{aligned}\hat{x}_{a,b} &= \frac{1}{H} \left(\sum_{(\mathbf{u}',\mathbf{i}') \in S} r_{\mathbf{u}',\mathbf{i}'} e^{-\frac{1-\text{Cos}(\mathbf{u},\mathbf{u}')}{h_u^2}} e^{-\frac{1-\text{Cos}(\mathbf{i},\mathbf{i}')}{h_i^2}} \right) \\ &= \frac{1}{H} \left(\sum_{(\mathbf{u}',\mathbf{i}) \in S_i} r_{\mathbf{u}',\mathbf{i}} E + \sum_{(\mathbf{u},\mathbf{i}') \in S_u} r_{\mathbf{u},\mathbf{i}'} F + \sum_{(\mathbf{u}',\mathbf{i}') \in S_{\bar{u},\bar{i}}} r_{\mathbf{u}',\mathbf{i}'} G \right),\end{aligned}\quad (6.42)$$

where H is a normalization factor equal to

$$\sum_{(\mathbf{u}',\mathbf{i}') \in S} e^{-\frac{1-\text{Cos}(\mathbf{u},\mathbf{u}')}{h_u^2}} e^{-\frac{1-\text{Cos}(\mathbf{i},\mathbf{i}')}{h_i^2}} \quad (6.43)$$

The three types of ratings $r_{\mathbf{u}',\mathbf{i}}$, $r_{\mathbf{u},\mathbf{i}'}$ and $r_{\mathbf{u}',\mathbf{i}'}$ that contribute to the prediction are precisely the similar user ratings (SUR), the similar item ratings (SIR) and the similar user towards similar item ratings (SUIR). E , F and G determine three weights for averaging these ratings:

$$\begin{aligned}E &= e^{-\frac{1-\text{Cos}(\mathbf{u},\mathbf{u}')}{h_u^2}} \\ F &= e^{-\frac{1-\text{Cos}(\mathbf{i},\mathbf{i}')}{h_i^2}} \\ G &= e^{-\frac{1-\text{Cos}(\mathbf{u},\mathbf{u}')}{h_u^2}} e^{-\frac{1-\text{Cos}(\mathbf{i},\mathbf{i}')}{h_i^2}}\end{aligned}\quad (6.44)$$

The bandwidth parameters h_u and h_i control the width of the kernel function. A small bandwidth value leads to spiky estimates while larger bandwidth values over-smooth the observations with data from far away samples. The bandwidth parameters also balance the contributions from the user side and the item side. A small h_u emphasizes user correlations, and a small h_i emphasizes the item correlations (see also the experiments corresponding to Fig. 6.9 and Fig. 6.10). When h_u approaches ∞ , the unified relevance model corresponds to item-based collaborative filtering, while an h_i of ∞ results in user-based recommendation.

6.4.7 Computational Complexity

This section discusses the scalability of our collaborative filtering framework. The computational complexity of the framework consists of the amount of time needed for building the model (i.e. the EM estimation of the two bandwidth parameters and the probability estimations), and that of making online recommendations for a new user from the model. The EM algorithm is only needed during the model building phase. Thus there are no iterative steps required in the online recommendation phase. Furthermore, we propose an efficient method to calculate the kernel-based similarities, largely reducing the computational complexity.

6.4.7.1 Offline Computation

In the model building phase, Eq. 6.29c simplifies the probability estimations to the calculations of the kernel-based similarities. That is, for any user pair and item pair, we need to calculate their kernel similarities $e^{-\frac{\|\mathbf{u}-\mathbf{u}'\|^2}{2h_u^2}}$ and $e^{-\frac{\|\mathbf{i}-\mathbf{i}'\|^2}{2h_i^2}}$, respectively. To make the calculation efficient, we decompose them into the distance measure part ($\|\mathbf{u}-\mathbf{u}'\|$ or $\|\mathbf{i}-\mathbf{i}'\|$) and the kernel smoothing part ($e^{-\frac{\cdot^2}{2h_u^2}}$ or $e^{-\frac{\cdot^2}{2h_i^2}}$). Thus, model building consists of two steps: first computing two distance (dis-similarity) matrices and then estimating the two bandwidth parameters.

For each element in the matrix, we require either B arithmetic operations for the user-to-user distance or A arithmetic operations for the item-to-item distance. In total the upper bound of the computational complexity is $O(A^2B + B^2A)$, which is roughly equal to a sum of the complexity of a user-based method [37] and that of an item-based method [92]. Since the data is extremely sparse, with a proper data structure, the computation can be largely reduced. This paper proposes to use two inverted files, respectively indexing users and items about their ratings. When we calculate the distances, for instance, for a given user, we do not need to go through *all* other users about their agreement on *all* items. Instead, we first from the user indexing file get the set of items that he or she has rated, and then go through these items, accessing a set of users who have rated any of these items (from the item index file). By doing this, not only do we exclude the users who do not share any commonly-rated item from the computations, we also restrict the operations to those items that the two users both rated. Thus the overall computation is much faster than the original user-based or item-based methods, only requiring a linear time complexity that approximately equals $O((A+B)mn)$, where m is the average number of user ratings per item and n is the average number of item ratings per user. In addition, it is unnecessary to store all the non-zero elements in the distance matrices, as we shall see in our experiment (Fig. 6.8) that keeping only the top- N nearest user neighbors and the top- N nearest item neighbors improves prediction accuracy, where N is typically in the range of (30...70).

Once we have the user (item) distance matrix that stores the top- N nearest neighbors, the EM estimation of h_i and h_u becomes a relatively simple task because it essentially averages the user or item distance from the distance matrices in an iterative manner (see Eq. 6.31). For the sake of time efficiency, the E step and M step can be computed together, where the computational complexity in each iteration is given by $O(ABN^2)$ because we need to average all possible user-item pair ($A \times B$ operations) and for each pair, we need to access $N \times N$ neighbors. In practice, the complexity of the EM algorithm can be further reduced by sub-sampling both training users and training items; our

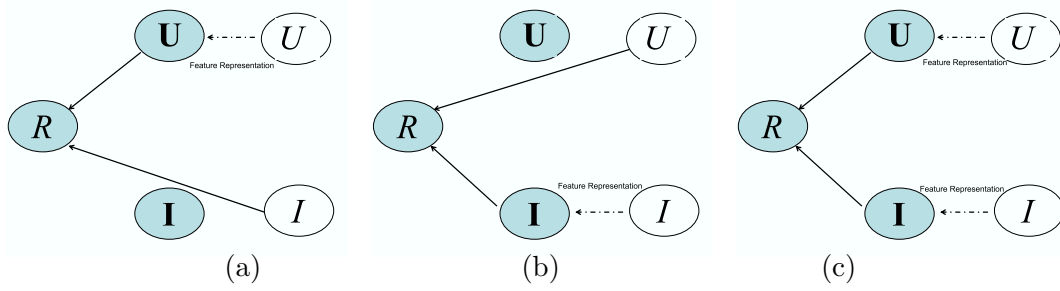


Figure 6.3: Illustration of the three different models. (a) User-based Relevance Model. (b) Item-based Relevance Model. (c) Unified Relevance Model.

Table 6.4: Characteristics of the test data sets.

| | MovieLens 1 | MovieLens 2 | EachMovies 1 | EachMovies 2 |
|-----------------------------------|-------------|-------------|--------------|--------------|
| Num. of Users | 943 | 500 | 2,000 | 10,000 |
| Num. of Items | 1682 | 1000 | 1,648 | 1,648 |
| Avg. Num. of Rated Item Per User | 106.0 | 87.7 | 90.0 | 96.3 |
| Avg. Num. of User Rating Per Item | 59.5 | 43.9 | 114.2 | 611.0 |
| Sparsity | 6.30% | 8.77% | 5.7% | 6.11% |
| Rating Scales | 5(1-5) | 5(1-5) | 6(1-6) | 6(1-6)) |

additional experiments (not reported) verified that a small amount of training user-item pairs is sufficient to get the stable and accurate estimations of h_u and h_i . We shall see in our experiment (Fig. 6.4) that the EM algorithm converges fast, and a small number of iterations (3-5) suffices to get relatively stable estimations in the tested data sets.

6.4.7.2 Online Computation

The online recommendation can be computed very efficiently if we again utilize both the user and item indexing files. For a new user, the computational complexity of his or her kernel similarity towards other users is approximately given as $O(mn)$ because on average we access his or her m rated items and for each of these items access n users who rate it. The final prediction corresponds to a weighted average from the ratings of the top- N nearest items and the top- N nearest users, resulting in a computational complexity of $O(N^2)$, which is independent of the number of users and items.

6.5 Experiments

6.5.1 Data Sets

We experimented with two movie rating data sets: the MovieLens [19] and the EachMovie [20] data sets.

The MovieLens data set was collected by the GroupLens group through the MovieLens web site during the period from September 1997 through April 1998. It contains ratings by 943 users for 1682 movies (items). Each user has rated at least 20 movies. The rating scale takes values from 1 (the lowest rating) to 5 (the highest rating). In addition, to compare with other approaches we also adopt a widely-used subset [21], which contains 500 users and 1000 movies (items), where each user has rated at least 40 items.

The EachMovie data set was collected by the Digital Equipment Research Center during the period from 1995 to 1997. The rating scale was originally indicated as the values from 0 (no star), 0.2 (one star) and up to 1 (five stars). For consistency with the MovieLens data set, we transformed the rating scales to the values 1 – 6, with 1 being the lowest rating (i.e., no star) and 6 being the highest one (i.e., five stars). To compare with other approaches, we adopt the two subsets described in [99] and [123], which respectively contain 2,000 users and 10,000 users. In both cases, each user has rated as least 40 items. The basic characteristics of these two data sets with the different size are summarised in Table 6.4. We mainly use the MovieLens 1 data set to conduct empirical analyses on our models while using the other sets to conduct performance tests.

6.5.2 Evaluation Protocols

For testing, we assigned the users in the data set at random to a training user and a test user set. Users in the training set are used as the basis for making predictions, while our test users are considered the ground truth for measuring prediction accuracy. Each test user's ratings have been split into a set of *observed items* and one of *held-out items*. The ratings of the observed items are input for predicting the ratings of the held-out items (the test items). To improve our measurements, each of the experimental setups has been repeated 20 times with different random seeds.

For consistency with experiments reported in the literature, e.g., [48, 92, 123]), we report our results using the mean absolute error (MAE) evaluation metric. MAE corresponds to the average absolute deviation of predictions to the ground

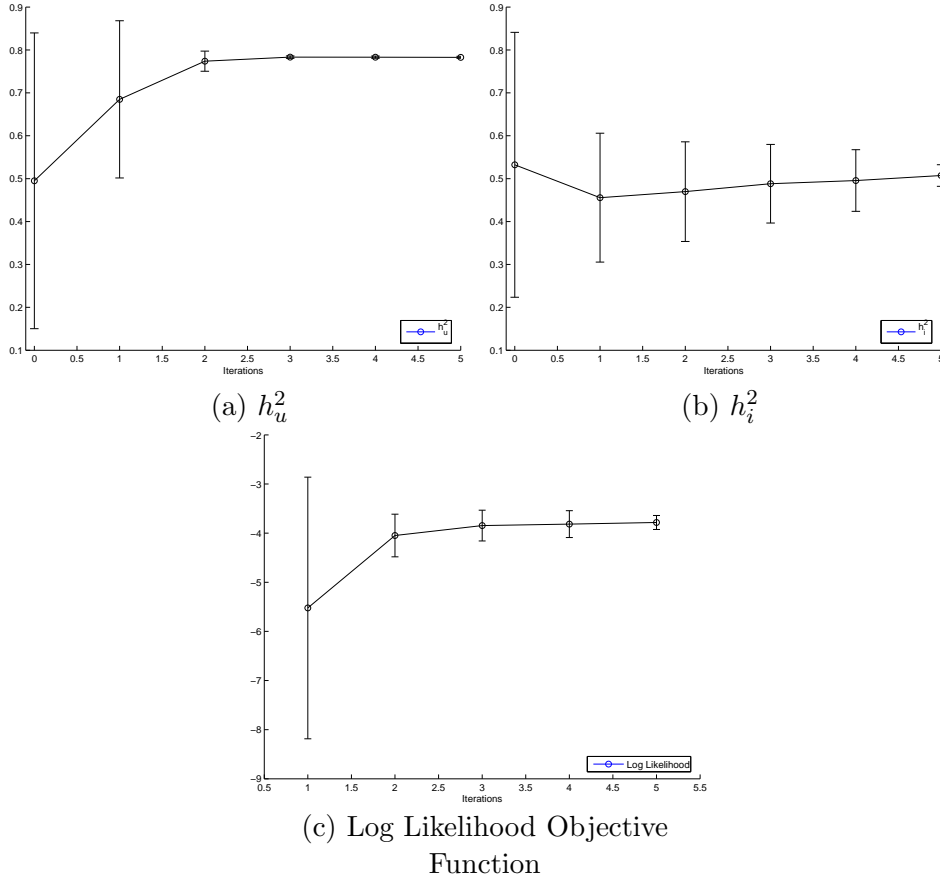


Figure 6.4: Convergence behavior of the cross-validated EM algorithm: Using the cosine projection kernel (400 training users in the MovieLens 1 data set).

truth data, for all test item ratings and test users:

$$MAE = \frac{\sum_{a,b} |x_{a,b} - \hat{x}_{a,b}|}{L}, \quad (6.45)$$

where L denotes the number of tested ratings. A smaller value indicates a better performance.

6.5.3 Results

6.5.3.1 Parameter Estimation

This section conducts experiments on the EM estimation for the parameters h_u and h_i . We first test the convergence behavior of the cross-validated EM

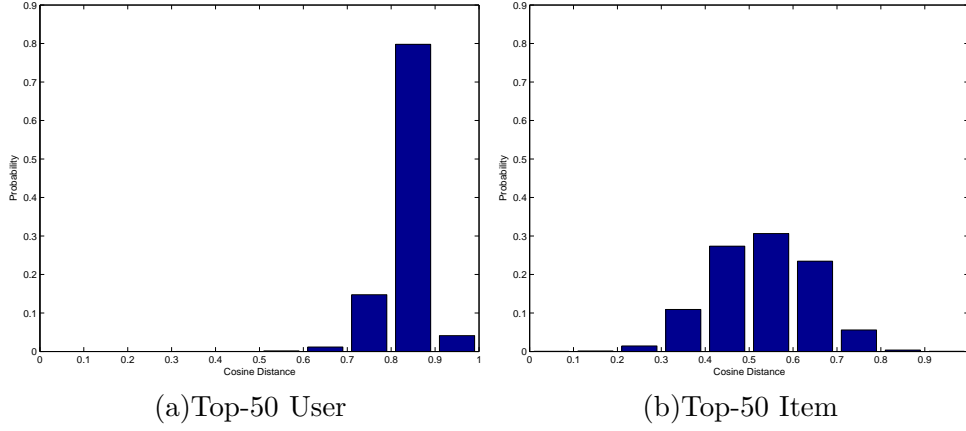


Figure 6.5: Distribution of cosine distance in the MovieLens data set (400 training users in the MovieLens 1 data set).

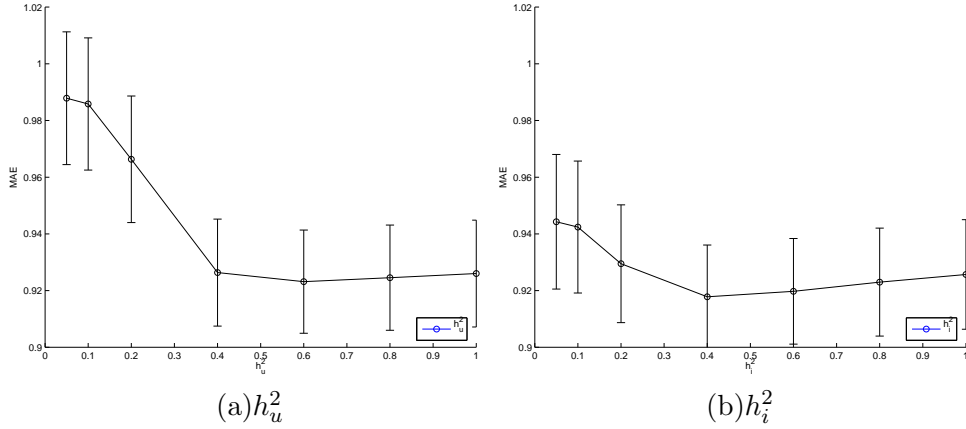


Figure 6.6: The sensitivity of the two parameters regarding to the MAE measurement (the remaining 543 test users in the MovieLens 1 data set).

method using MovieLens set 1. To reduce redundancy, we only show the estimation results when we randomly select 400 users as the training data. Notice that the estimation over other number of training users behaves consistently. Fig. 6.4 shows that the EM algorithm converges in few iterations (about 3) to the optimal bandwidth parameters ($h_u^2 = 0.79$ (Fig. 6.4(a)) and $h_i^2 = 0.49$ (Fig. 6.4(b))) with respect to the log likelihood object function (Fig. 6.4 (c)). Repeating this experiment with different random initial values of h_u and h_i (have a relatively large standard deviation in the figures), the EM algorithm converges to (almost) the same optimal values (have a relatively small standard deviation in the figures). Observe that the obtained bandwidth parameter h_u is relatively larger than the parameter h_i . To explain this result, we investigate the influence

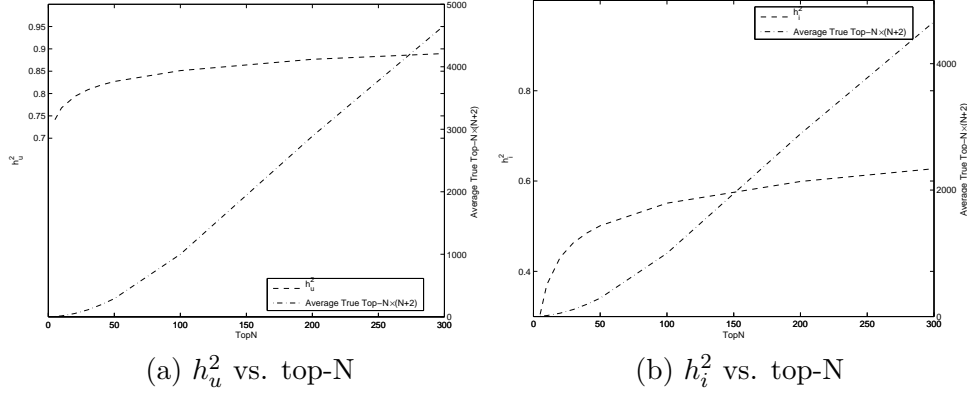


Figure 6.7: The impact of the neighbor size on the parameter estimation. (400 training users in the MovieLens 1 data set).

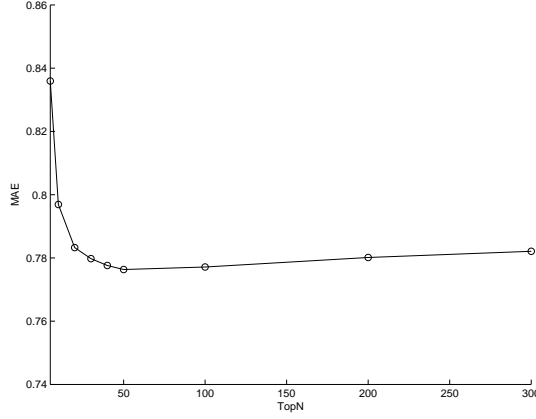


Figure 6.8: The MAE performance under different neighbor size (the remaining 543 test users in the MovieLens 1 data set).

of the distribution of neighboring vectors on the parameter estimation. In our models, both the user distance and item distance are measured by the cosine distance, i.e. $1 - \text{Cos}(\mathbf{u}, \mathbf{u}')$ and $1 - \text{Cos}(\mathbf{i}, \mathbf{i}')$ (Eq. 6.40a–6.40c). Fig. 6.5 plots the distributions of the top-50 nearest users and items as a histogram of 10 bins in the distance range of $[0, 1]$. It shows that, on average, the user distances are larger than the item distances. So, estimating the user density needs a bigger bandwidth parameter to smooth from the neighborhood than that of the item density.

To show the sensitivity of the two bandwidth parameters regarding to the recommendation performance, we plot the value of the bandwidth parameter (either h_u or h_i) against the MAE measurement in Fig. 6.6. It shows that the two bandwidth parameters are relatively stable in a wide range regarding to the

MAE performance. Also, comparing the optimal bandwidth parameters with the ones shown in Fig. 6.4 we can see that, although the EM algorithm uses a different objective function (log likelihood), it does give a reasonably good estimation of the two bandwidth parameters in terms of the MAE.

In practice, recommendation systems make a trade-off between prediction accuracy and run-time system efficiency by pre-selecting the top- N nearest user neighbors (SUR) and the top- N nearest item neighbors (SIR). These form a rating pool, extended with the top- $N \times N$ nearest similar user to similar item neighbors (SUIR). In total, we would have $N \times (N + 2)$ neighbors. However, only the neighbors whose ratings are available in the pool contribute to the predictions. Clearly, the parameter N affects the parameter estimation and therefore also the performance of our fusion methods. Fig. 6.7 plots the estimated parameter values of h_u and h_i under different neighbor sizes, where the x axis represents the size of the pre-selected top- N and the left y axis corresponds to the estimated value of the parameter (h_u in Fig. 6.7(a) and h_i in Fig. 6.7(b)). The right y axis shows the ‘true’ number of SUR, SIR and SUIR within the pool that have to be retrieved to obtain non-empty ratings for estimation. The graph shows that for low values of N , both parameters h_u and h_i increase fast. This shows that the new ratings introduced by increasing the top- N contribute to improve the prediction accuracy, such that the corresponding Parzen-window should cover the newly introduced ratings (so, the bandwidth parameters increase). Gradually however, this increase diminishes, because a large top- N introduces more and more noisy ratings. Consequently, the parameter values converge and the Parzen-window excludes the distant ratings from the estimation process. Fig. 6.8 displays the MAE of the unified relevance model under different neighbor sizes, to illustrate the effect of the estimated parameters on prediction accuracy. The optimal result corresponds to $N \simeq 50$. The error increases only slowly with larger values of N , due to the fact that the Parzen-window reduces the effect of the noisy (distant) neighbors when they are introduced. We select 50 as the optimal choice of N for the subsequent experiments.

6.5.3.2 Sparsity

This section investigates the effect of data sparsity on the performance of our collaborative filtering methods in more detail. For this, we set up the following configurations: 1) *Test User Sparsity*: vary the number of items rated by test users in the observed set, e.g. 5, 10, or 20 ratings per user. 2) *Test Item Sparsity*: vary the number of users who have rated test items in the held-out set, e.g. less than 15, 20, 25 (denoted as ‘< 15’, ‘< 20’, or ‘< 25’), or, unconstrained (denoted as ‘No constraint’). Notice that the configurations of the user sparsity and the item sparsity are not completely symmetrical in order

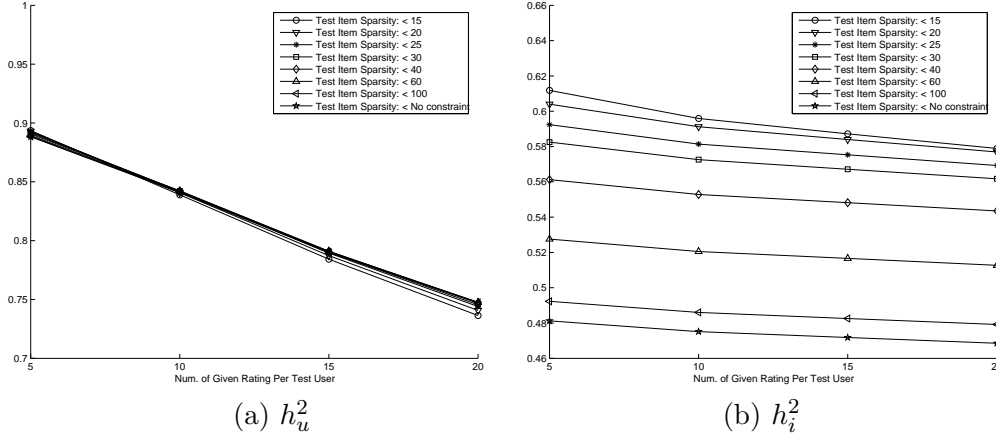


Figure 6.9: The optimal bandwidth parameters with different user sparsity.

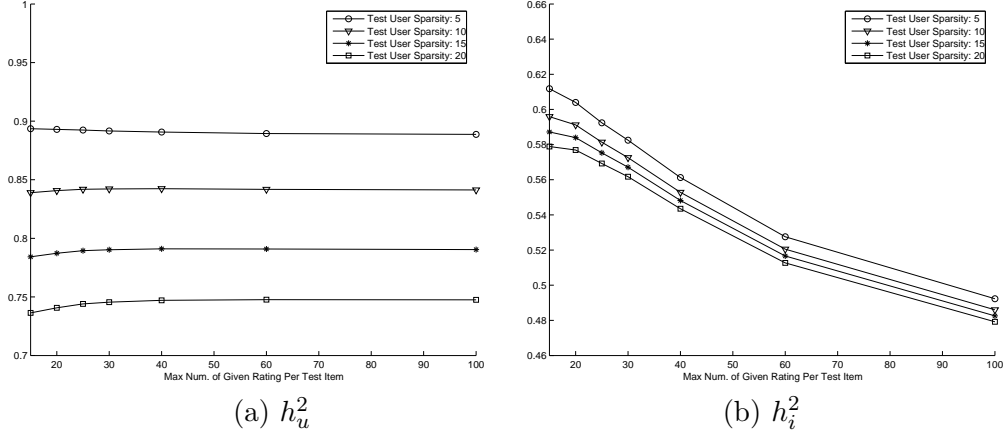


Figure 6.10: The optimal bandwidth parameters under different item sparsity.

to reflect the practical situation.

The first experiment investigates the effect of data sparsity on parameter estimation using the EM algorithm. We use the different sparsity configurations: user sparsity: number of given ratings per user (5, 10, 15, 20) and item sparsity: maximum number of user rating per item (<15, <20, <25, <30, <40, <60, <100, *No constraint*). For each configuration, we select 400 users in the MovieLens data set to run the EM algorithm to obtain the optimal parameters h_i and h_u .

Fig. 6.9 (a) and (b) show, respectively, the optimal values of h_u^2 (y axis in Fig. 6.9 (a)) and h_i^2 (y axis in Fig. 6.9 (b)) under different user sparsity conditions (x axis). The figures demonstrate that when the user sparsity decreases (and therefore the number of given ratings per user increases), the optimal user bandwidth parameter h_u becomes smaller while the optimal item bandwidth

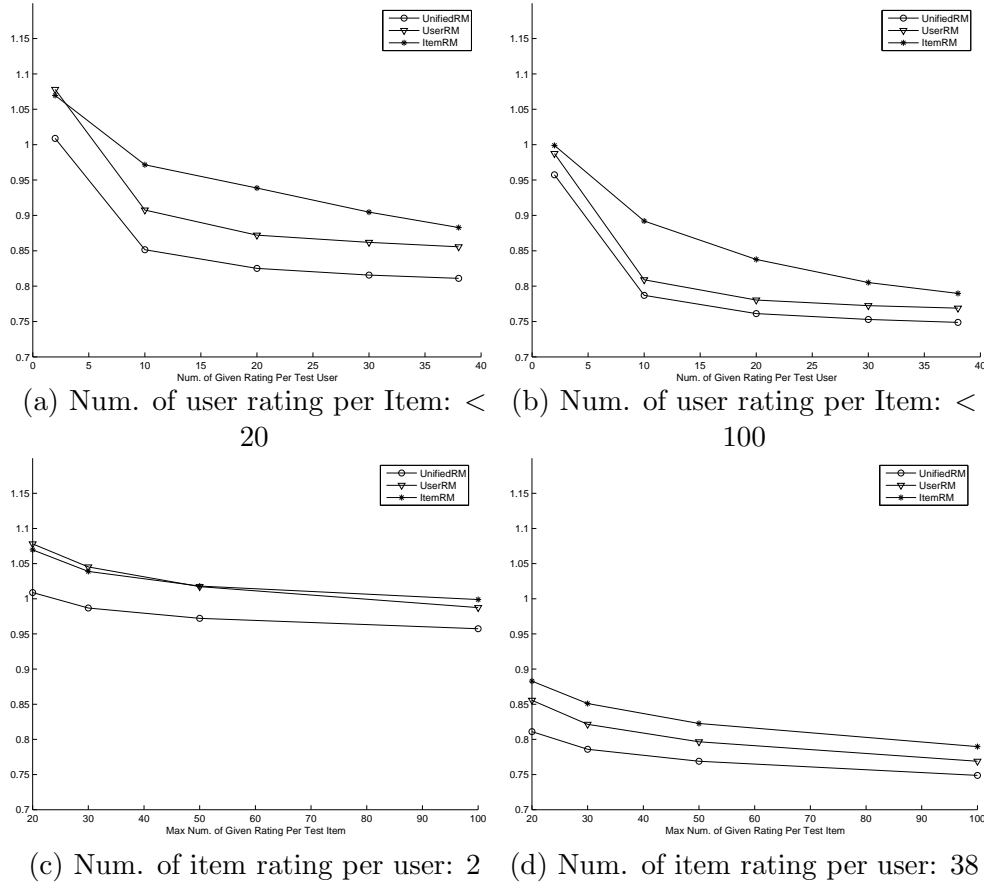


Figure 6.11: Performance of the three derived models under different sparsity in the MovieLens 1 data set.

parameter h_i stays relatively constant. This is due to the fact that, for a given test user, when the number of item ratings provided by this user is small, it is difficult for the test user to find other users who share ratings among the small amount of item ratings provided. Consequently, the test user has less neighbors, calling for a wide Parzen-window (a larger bandwidth parameter h_u) such that the users with relatively large distance can still contribute and smooth the density estimation (rating prediction). However, as the number of item ratings per user increases, for a given test user, he or she has more item ratings to be used to find similar users. In this case, the test user has more neighbors. Thus, it is expected to have a smaller bandwidth parameter to produce a narrow kernel so as to give more emphasis on the most similar users for the density estimation. In both cases, bandwidth parameter h_i varies less than h_u , because the item sparsity remains relatively constant. Fig. 6.10(a) and (b) demonstrate the same behaviour when varying item sparsity. A low number of user ratings per item results in a relatively large bandwidth parameter h_i . Conversely, when

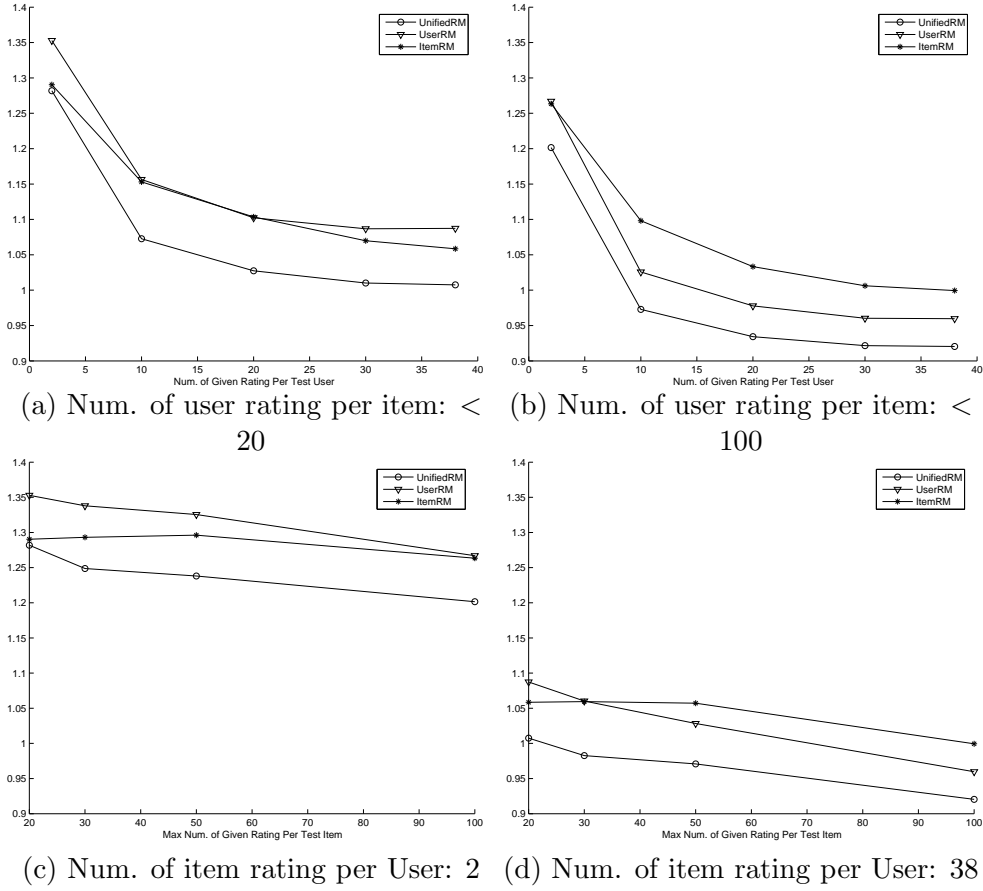


Figure 6.12: Performance of the three derived models under different sparsity in the EachMovie 1 data set.

the number of user ratings per item increases, a small bandwidth parameter h_i is obtained.

We conclude that the EM algorithm adapts the bandwidth parameters successfully to the different sparsity situations. Let us now compare the performance of the three different models for rating prediction: the unified relevance model, the user-based relevance model and the item-based relevance model (Eq. 6.40a–6.40c).

We vary user sparsity at 2, 10, 20, 30 and 38, and item sparsity ranging from <20, <30, <50 to <100. In both the MovieLens 1 and EachMovie 1 data sets, we randomly assign 400 users to the training set, and use the remaining users as the test set. Fig. 6.11 and 6.12 summarise the results on the MovieLens 1 and the Eachmovie 1 data sets, respectively. The experiments with varying user and item sparsity settings show that the MAE performance of each rating prediction model improves with the number of given ratings per test user. Figures 6.11(c),

Table 6.5: Comparison among the three derived models on the MovieLens 1 data set. The MAE and P-value of the 10-fold Wilcoxon signed-rank test are reported (the minimum P-value for 10-fold is 0.002).

| User Sparsity | 5 | 10 | 20 | 30 | 38 |
|------------------|----------------|----------------|----------------|----------------|----------------|
| Unified RM | 1.009 | 0.851 | 0.825 | 0.816 | 0.811 |
| User-based RM | 1.078 | 0.908 | 0.872 | 0.862 | 0.856 |
| Item-based RM | 1.070 | 0.972 | 0.939 | 0.905 | 0.883 |
| - | P-value | P-value | P-value | P-value | P-value |
| Unified - UserRM | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) |
| Unified - ItemRM | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) |

(a) Num. of user rating per item: <20

| User Sparsity | 5 | 10 | 20 | 30 | 38 |
|------------------|----------------|----------------|----------------|----------------|----------------|
| Unified RM | 0.987 | 0.826 | 0.799 | 0.791 | 0.786 |
| User-based RM | 1.045 | 0.869 | 0.836 | 0.826 | 0.821 |
| Item-based RM | 1.039 | 0.947 | 0.905 | 0.872 | 0.851 |
| - | P-value | P-value | P-value | P-value | P-value |
| Unified - UserRM | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) |
| Unified - ItemRM | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) |

(b) Num. of user rating per item: <30

| User Sparsity | 5 | 10 | 20 | 30 | 38 |
|------------------|----------------|----------------|----------------|----------------|----------------|
| Unified RM | 0.972 | 0.806 | 0.781 | 0.773 | 0.769 |
| User-based RM | 1.017 | 0.839 | 0.809 | 0.800 | 0.797 |
| Item-based RM | 1.018 | 0.922 | 0.875 | 0.841 | 0.823 |
| - | P-value | P-value | P-value | P-value | P-value |
| Unified - UserRM | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) |
| Unified - ItemRM | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) |

(c) Num. of user rating per item: <50

| User Sparsity | 5 | 10 | 20 | 30 | 38 |
|------------------|----------------|----------------|----------------|----------------|----------------|
| Unified RM | 0.957 | 0.787 | 0.761 | 0.753 | 0.749 |
| User-based RM | 0.987 | 0.809 | 0.780 | 0.772 | 0.769 |
| Item-based RM | 0.999 | 0.892 | 0.838 | 0.805 | 0.790 |
| - | P-value | P-value | P-value | P-value | P-value |
| Unified - UserRM | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) |
| Unified - ItemRM | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) |

(d) Num. of user rating per item: <100

6.12(c) and (d), demonstrate that the user-based relevance model improves more from a higher number of given ratings per test item than the item-based relevance model does, especially in the EachMovie case. At first sight, this is surprising as we would expect the item-based relevance model to improve most from a reduced item sparsity (i.e., from having a more reliable item-to-item similarity measure). Careful investigating of this finding shows however that prediction accuracy does not only depend upon the reliability of the similarity measure, but *also* relies on the number of similar ratings that contribute to the predictions. The larger number of given ratings per test item improves the reliability of the item-to-item similarity measure in the item-based relevance model, but it *also* increases the number of ratings by users that are similar to the test users (the SURs) in the user-based relevance model. Both effects

Table 6.6: Comparison among the three derived models on the EachMovie 1 data set. The MAE and P-value of the 10-fold Wilcoxon signed-rank test are reported (the minimum P-value for 10-fold is 0.002).

| User Sparsity | 5 | 10 | 20 | 30 | 38 |
|------------------|----------------|----------------|----------------|----------------|----------------|
| Unified RM | 1.282 | 1.073 | 1.027 | 1.010 | 1.007 |
| User-based RM | 1.353 | 1.156 | 1.102 | 1.087 | 1.087 |
| Item-based RM | 1.290 | 1.153 | 1.104 | 1.070 | 1.059 |
| - | P-value | P-value | P-value | P-value | P-value |
| Unified - UserRM | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) |
| Unified - ItemRM | 0.557(>0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.004(<0.05) | 0.010(<0.05) |

(a) Num. of user rating per item: <20

| User Sparsity | 5 | 10 | 20 | 30 | 38 |
|------------------|----------------|----------------|----------------|----------------|----------------|
| Unified RM | 1.249 | 1.040 | 0.999 | 0.983 | 0.983 |
| User-based RM | 1.338 | 1.128 | 1.076 | 1.062 | 1.060 |
| Item-based RM | 1.293 | 1.155 | 1.098 | 1.069 | 1.059 |
| - | P-value | P-value | P-value | P-value | P-value |
| Unified - UserRM | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) |
| Unified - ItemRM | 0.006(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) |

(b) Num. of user rating per item: <30

| User Sparsity | 5 | 10 | 20 | 30 | 38 |
|------------------|----------------|----------------|----------------|----------------|----------------|
| Unified RM | 1.238 | 1.025 | 0.986 | 0.972 | 0.971 |
| User-based RM | 1.326 | 1.095 | 1.049 | 1.028 | 1.028 |
| Item-based RM | 1.296 | 1.159 | 1.096 | 1.067 | 1.057 |
| - | P-value | P-value | P-value | P-value | P-value |
| Unified - UserRM | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) |
| Unified - ItemRM | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) |

(c) Num. of user rating per item: <50

| User Sparsity | 5 | 10 | 20 | 30 | 38 |
|------------------|----------------|----------------|----------------|----------------|----------------|
| Unified RM | 1.202 | 0.973 | 0.934 | 0.922 | 0.920 |
| User-based RM | 1.267 | 1.026 | 0.978 | 0.960 | 0.960 |
| Item-based RM | 1.263 | 1.098 | 1.033 | 1.006 | 0.999 |
| - | P-value | P-value | P-value | P-value | P-value |
| Unified - UserRM | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) |
| Unified - ItemRM | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) | 0.002(<0.05) |

(d) Num. of user rating per item: <100

contribute to better rating predictions, but increasing the number of SURs proves to be more beneficial.

Fig. 6.12(d) shows how the user-based relevance model gradually outperforms the item-based relevance model as the number of given user rating per item increases. More importantly however, we find that the unified relevance model outperforms the user-based relevance model and the item-based relevance model in all sparsity settings. Tables 6.5 and 6.6 list more details of the performance comparison over the two different data sets, to investigate the statistical significance of the performance improvement obtained by the unified relevance model. It shows the P-value of a Wilcoxon signed-rank test [45] applied to each configuration. We conclude that the unified relevance model consistently and

significantly improves the recommendation performance over the user-based and item-based relevance models, irrespective of the sparsity (except for the one exception in the top left corner of Table 6.6, where the difference with the item-based relevance model is not significant). We conclude that the unified relevance model is indeed effective at fusing the predictions from user and item aspects.

Table 6.7: Comparison with other approaches on the EachMovie 1 and MovieLens 1 data sets. The MAE and P-value of the 10-fold Wilcoxon signed-rank test are reported (the minimum P-value for 10-fold is 0.002).

| Given Ratings | 5 | 10 | 20 | 30 |
|-------------------|---------------|---------------|---------------|---------------|
| Unified RM | 1.011 | 0.919 | 0.887 | 0.876 |
| UserVS | 1.108 | 1.041 | 1.023 | 1.018 |
| UserPCC | 1.079 | 0.954 | 0.911 | 0.893 |
| ItemVS | 1.101 | 1.025 | 0.998 | 0.988 |
| ItemPCC | 1.120 | 1.013 | 0.969 | 0.954 |
| PD | 1.187 | 1.084 | 1.055 | 1.050 |
| - | P-value | P-value | P-value | P-value |
| Unified - UserVS | 0.002 (<0.05) | 0.002 (<0.05) | 0.002 (<0.05) | 0.002 (<0.05) |
| Unified - UserPCC | 0.002 (<0.05) | 0.002 (<0.05) | 0.002 (<0.05) | 0.002 (<0.05) |
| Unified - ItemVS | 0.002 (<0.05) | 0.002 (<0.05) | 0.002 (<0.05) | 0.002 (<0.05) |
| Unified - ItemPCC | 0.002 (<0.05) | 0.002 (<0.05) | 0.002 (<0.05) | 0.002 (<0.05) |
| Unified - PD | 0.002 (<0.05) | 0.002 (<0.05) | 0.002 (<0.05) | 0.002 (<0.05) |

(a) the EachMovie 1 data set.

| Given Ratings | 5 | 10 | 20 | 30 |
|-------------------|---------------|---------------|---------------|---------------|
| Unified RM | 0.837 | 0.769 | 0.749 | 0.741 |
| UserVS | 0.900 | 0.845 | 0.832 | 0.828 |
| UserPCC | 0.888 | 0.803 | 0.775 | 0.762 |
| ItemVS | 0.910 | 0.829 | 0.803 | 0.794 |
| ItemPCC | 0.954 | 0.865 | 0.813 | 0.795 |
| PD | 0.927 | 0.865 | 0.837 | 0.827 |
| - | P-value | P-value | P-value | P-value |
| Unified - UserVS | 0.002 (<0.05) | 0.002 (<0.05) | 0.002 (<0.05) | 0.002 (<0.05) |
| Unified - UserPCC | 0.002 (<0.05) | 0.002 (<0.05) | 0.002 (<0.05) | 0.002 (<0.05) |
| Unified - ItemVS | 0.002 (<0.05) | 0.002 (<0.05) | 0.002 (<0.05) | 0.002 (<0.05) |
| Unified - ItemPCC | 0.002 (<0.05) | 0.002 (<0.05) | 0.002 (<0.05) | 0.002 (<0.05) |
| Unified - PD | 0.002 (<0.05) | 0.002 (<0.05) | 0.002 (<0.05) | 0.002 (<0.05) |

(b) the MovieLens 1 data set.

6.5.3.3 Comparison to other approaches

We continue with a comparison to results obtained with other methods. Each setting uses the optimal h_u and h_i learned with the EM algorithm.

Recall that our unified model provides a very general framework for collaborative filtering, particularly for those that make use of the neighborhood concept. Thus, we first compare our unified model with other popular methods that

need to compute the paired similarities, for instance, the memory-based approaches. Also, the Personality Diagnosis method (see [77]) can be considered as a neighborhood-based approach as it requires to pre-compute the conditional probabilities (similarities) between two paired users. Table 6.7 presents the comparison of our unified model to the the Personality Diagnosis (PD) method and the four memory-based approaches, i.e., the used-based Pearson Correlation Coefficient (UserPCC) and Vector Space (UserVS) methods (see [9]), and the item-based Pearson Correlation Coefficient (ItemPCC) and Vector Space (ItemVS) methods (see [92]). In addition, we also conduct a significance test, showing the P-value of a Wilcoxon signed-rank test applied to each configuration. From the table, we can see that the recommendation performance of our unified model is indeed significantly better than that of other alternatives that have been considered.

Next we adopt the MovieLens 2 data set [21] (called the MovieRating test bed in [48, 123]) as well as the two EachMovie data sets. We followed the evaluation procedure described in [123] and [99], aiming to compare the performance of our unified model with the state-of-art results of the mixture models [99] and the cluster-based models [123]. Table 6.8 presents the comparison of our experimental results to the six methods of [99], i.e., the two extensions of the Aspect Models (AM_c, AM_d, see [99]), ‘Personality Diagnosis’ (PD) ([77]), the user-based Pearson Correlation Coefficient (PCC) ([9]), Vector Space (VC), and, Flexible Mixture Model (FMM) ([99]). On the Eachmovie 1 data set, our method outperforms all of these methods in all configurations. In the MovieLens 2 data set, only FMM attains comparable results. However the FMM method has more computation complexity than our unified model in the online recommendation phase as it requires the EM iterations called “fold-in” to find both the hidden user clusters and item clusters for new users.

Table 6.9 shows our experimental results as well as the results listed in [123], i.e., the cluster-based Pearson Correlation Coefficient (SCBPCC) and the cluster-based collaborative filtering (CBCF) ([123]), the Aspect Models (AM) ([41]), ‘Personality Diagnosis’ (PD) ([77]), and the user-based Pearson Correlation Coefficient (PCC) and Vector Space (VC) ([9]). For both two test sets, our method outperforms these methods in all configurations. By unifying the ratings from both user and item aspects for prediction, our unified relevance model is found to be effective in improving the prediction accuracy for recommendation consistently.

Table 6.8: Comparison with the result reported in [99]. The MAE is reported.

| Training Users: | 200 | | | 400 | | |
|-----------------|-------------|-------------|-------------|-------------|-------------|-------------|
| Given Ratings: | 5 | 10 | 20 | 5 | 10 | 20 |
| Unified Model | 1.05 | 0.97 | 0.94 | 1.04 | 0.96 | 0.93 |
| PCC | 1.22 | 1.16 | 1.13 | 1.22 | 1.16 | 1.13 |
| VS | 1.25 | 1.24 | 1.26 | 1.32 | 1.33 | 1.37 |
| PD | 1.19 | 1.16 | 1.15 | 1.18 | 1.16 | 1.15 |
| AM_a(20) | 1.27 | 1.18 | 1.14 | 1.28 | 1.19 | 1.16 |
| AM_a(10) | 1.18 | 1.17 | 1.16 | 1.15 | 1.14 | 1.13 |
| FMM | 1.07 | 1.04 | 1.02 | 1.05 | 1.03 | 1.01 |

(a) the EachMovie 1 data set

| Training Users: | 100 | | | 200 | | |
|-----------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Given Ratings: | 5 | 10 | 20 | 5 | 10 | 20 |
| Unified Model | 0.848 | 0.779 | 0.796 | 0.828 | 0.767 | 0.781 |
| PCC | 0.881 | 0.832 | 0.809 | 0.878 | 0.828 | 0.801 |
| VS | 0.859 | 0.834 | 0.823 | 0.862 | 0.950 | 0.854 |
| PD | 0.839 | 0.826 | 0.818 | 0.835 | 0.816 | 0.806 |
| AM_a(5) | 0.882 | 0.856 | 0.836 | 0.891 | 0.850 | 0.818 |
| AM_a(2) | 0.869 | 0.857 | 0.850 | 0.837 | 0.833 | 0.825 |
| FMM | 0.829 | 0.822 | 0.807 | 0.800 | 0.787 | 0.768 |

(b) the MovieLens 2 data set

Table 6.9: Comparison with the results reported in [123]. The MAE is reported.

| Training Users: | 500 | | | 2000 | | | 6000 | | |
|-----------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Given Ratings: | 5 | 10 | 20 | 5 | 10 | 20 | 5 | 10 | 20 |
| Unified Model | 1.061 | 0.969 | 0.938 | 1.054 | 0.957 | 0.921 | 1.061 | 0.954 | 0.918 |
| PCC | 1.157 | 1.075 | 1.048 | 1.124 | 1.052 | 1.020 | 1.118 | 1.039 | 0.988 |
| PD | 1.148 | 1.145 | 1.140 | 1.129 | 1.087 | 1.043 | 1.101 | 1.063 | 1.051 |
| AM | 1.157 | 1.082 | 1.057 | 1.125 | 1.078 | 1.054 | 1.117 | 1.069 | 1.046 |
| CBCF | 1.207 | 1.132 | 1.089 | 1.187 | 1.113 | 1.063 | 1.197 | 1.111 | 1.060 |
| SCBPCC | 1.105 | 1.041 | 1.004 | 1.085 | 1.014 | 0.973 | 1.073 | 1.001 | 0.956 |

(a) the EachMovie 2 data set

| Training Users: | 100 | | | 200 | | | 300 | | |
|-----------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|---------------|--------------|
| Given Ratings: | 5 | 10 | 20 | 5 | 10 | 20 | 5 | 10 | 20 |
| Unified Model | 0.848 | 0.779 | 0.796 | 0.828 | 0.767 | 0.781 | 0.799 | 0.7552 | 0.764 |
| PCC | 0.874 | 0.836 | 0.818 | 0.859 | 0.829 | 0.813 | 0.849 | 0.841 | 0.820 |
| PD | 0.849 | 0.817 | 0.808 | 0.836 | 0.815 | 0.792 | 0.827 | 0.815 | 0.789 |
| AM | 0.963 | 0.922 | 0.887 | 0.849 | 0.837 | 0.815 | 0.820 | 0.822 | 0.796 |
| CBCF | 0.924 | 0.896 | 0.890 | 0.908 | 0.879 | 0.852 | 0.847 | 0.846 | 0.821 |
| SCBPCC | 0.848 | 0.819 | 0.789 | 0.831 | 0.813 | 0.784 | 0.822 | 0.810 | 0.778 |

(b) the MovieLens 2 data set

6.6 Conclusions and Future Work

This paper presented a unified probabilistic model for collaborative filtering. We explain how to use Parzen-window density estimation for acquiring the probabilities of the proposed unified relevance model. We generalised the kernel density estimation by applying the ‘kernel trick’, and showed that the often used cosine measure is a suitable projection kernel function. The resulting method has been shown to produce highly accurate predictions on common benchmark

data.

The probabilistic framework calls for interesting future work. Firstly, we intend to explore smoothing techniques as an extra technique to tackle data sparsity. For instance, it is possible to use interpolation smoothing to introduce a background model into the density estimation. Secondly, we plan to look at other IR models for collaborative filtering problems, especially in the situation where we need to pose collaborative filtering as an item ranking problem. The well-known Probability Ranking Principle (PRP) of information retrieval [83] is of particular interest as it provides a theoretical guideline for ranking documents (items). In this regard, we will investigate the possible usages of other ranking models such as the language modelling of information retrieval [18, 114] and the BM25 ranking formulas [87]. Thirdly, to deal with different scenarios in recommender systems, we will investigate the possible integration of other text retrieval techniques (more specifically, query expansion and relevance feedback). Fourthly, since our methods are general models for co-occurrence data, it is also worthwhile seeking the possible usage of the models beyond collaborative filtering. We are particularly interested in applying the unified relevance model for the unification of document and query generation in text retrieval.

6.A Cross-validated EM algorithm

This appendix derives a cross-validated expectation maximisation algorithm to select an optimal value of the smoothing parameters h_u and h_i . Of course, this depends for a large part on the data: on the number of data points and their distribution. Other factors of influence are the Parzen window function and the optimality criterion.

The goal is to select h_u and h_i such that they maximise the likelihood function:

$$\begin{aligned}\hat{h}_u, \hat{h}_i &= \arg \max_{h_u, h_i} \prod_{(\mathbf{u}, \mathbf{i}) \in S_r} P(\mathbf{u}, \mathbf{i} | r) \\ &= \arg \max_{h_u, h_i} \prod_{(\mathbf{u}, \mathbf{i}) \in S_r} \frac{1}{|S_r|} \sum_{(\mathbf{u}', \mathbf{i}') \in S_r} \frac{1}{h_u^B} \mathbf{K}\left(\frac{\mathbf{u} - \mathbf{u}'}{h_u}\right) \frac{1}{h_i^A} \mathbf{K}\left(\frac{\mathbf{i} - \mathbf{i}'}{h_i}\right)\end{aligned}\quad (6.46)$$

It is easy to see that the joint distribution reaches an absolute maximum when $h_u = 0$ and $h_i = 0$. [26] has proposed cross-validated maximum likelihood estimation to remove this anomaly,

$$\hat{h}_u, \hat{h}_i = \arg \max_{h_u, h_i} \prod_{(\mathbf{u}, \mathbf{i}) \in S_r} \frac{1}{|S_r - 1|} \sum_{(\mathbf{u}', \mathbf{i}') \in S_r \wedge (\mathbf{u}' \neq \mathbf{u}) \wedge (\mathbf{i}' \neq \mathbf{i})} \frac{1}{h_u^B} \mathbf{K}\left(\frac{\mathbf{u} - \mathbf{u}'}{h_u}\right) \frac{1}{h_i^A} \mathbf{K}\left(\frac{\mathbf{i} - \mathbf{i}'}{h_i}\right)\quad (6.47)$$

This equation can be solved using the iterative expectation maximisation (EM) algorithm (e.g., [75]).

The Parzen-window density estimation method can be interpreted as a generative model with a large mixture of $|S_r - 1|$ (because of cross-validation) component densities with equal weight, where the means of the component densities (assuming a symmetric window function) are located at each observation. Test samples \mathbf{u} and \mathbf{i} are generated from component densities with means \mathbf{u}' and \mathbf{i}' , *i.e.*, $P(\mathbf{u}, \mathbf{i} | \mathbf{u}', \mathbf{i}' : \mathbf{h}_u, \mathbf{h}_i)$, with a prior probability of selecting that component equal to $P(\mathbf{u}', \mathbf{i}') = 1/|S_r - 1|$. Applying Bayes' rule to turn $P(\mathbf{u}, \mathbf{i} | \mathbf{u}', \mathbf{i}')$ into $P(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i})$ gives :

$$\begin{aligned}P(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) &= \frac{P(\mathbf{u}', \mathbf{i}') P(\mathbf{u}, \mathbf{i} | \mathbf{u}', \mathbf{i}' : \mathbf{h}_u, \mathbf{h}_i)}{\sum_{\mathbf{u}' \neq \mathbf{u}, \mathbf{i}' \neq \mathbf{i}} P(\mathbf{u}', \mathbf{i}') P(\mathbf{u}, \mathbf{i} | \mathbf{u}', \mathbf{i}' : \mathbf{h}_u, \mathbf{h}_i)} \\ &= \frac{(1/|S_r - 1|) \frac{1}{h_u^B} \mathbf{K}\left(\frac{\mathbf{u} - \mathbf{u}'}{h_u}\right) \frac{1}{h_i^A} \mathbf{K}\left(\frac{\mathbf{i} - \mathbf{i}'}{h_i}\right)}{\sum_{\mathbf{u}' \neq \mathbf{u}, \mathbf{i}' \neq \mathbf{i}} (1/|S_r - 1|) \frac{1}{h_u^B} \mathbf{K}\left(\frac{\mathbf{u} - \mathbf{u}'}{h_u}\right) \frac{1}{h_i^A} \mathbf{K}\left(\frac{\mathbf{i} - \mathbf{i}'}{h_i}\right)} \\ &= \frac{\mathbf{K}\left(\frac{\mathbf{u} - \mathbf{u}'}{h_u}\right) \mathbf{K}\left(\frac{\mathbf{i} - \mathbf{i}'}{h_i}\right)}{\sum_{\mathbf{u}' \neq \mathbf{u}, \mathbf{i}' \neq \mathbf{i}} \mathbf{K}\left(\frac{\mathbf{u} - \mathbf{u}'}{h_u}\right) \mathbf{K}\left(\frac{\mathbf{i} - \mathbf{i}'}{h_i}\right)}\end{aligned}\quad (6.48)$$

Equation 6.48 gives the E-step of the EM algorithm. The M-step of the EM algorithm uses a lower-bound $\Lambda(h_u, h_i | h_u^{(t)}, h_i^{(t)})$ to approximate the original maximum likelihood function, and then maximises the lower-bound. The E-step and M-step are iteratively applied until the algorithm converges to a (local) maximum [109]. This lower-bound towards the log form of the cross-validated likelihood function is obtained from *Jensen's Inequality*, which states that for any concave function \mathbf{f} , such that:

$$\mathbf{f}\left(\sum_i p_i x_i\right) \geq \sum_i p_i \mathbf{f}(x_i),$$

where $\sum_i p_i = 1, p_i \geq 0$ and $x_i \geq 0$. Because the logarithm is concave in the range of $(0, 1]$, Jensen's Inequality can be used to derive a lower bound for the likelihood function shown in Eq. 6.47:

$$\begin{aligned} & \prod_{\forall(\mathbf{u}, \mathbf{i}): (\mathbf{u}, \mathbf{i}) \in S_r} \frac{1}{|S_r - 1|} \sum_{(\mathbf{u}', \mathbf{i}') \in S_r \wedge (\mathbf{u}' \neq \mathbf{u}) \wedge (\mathbf{i}' \neq \mathbf{i})} \frac{1}{h_u^B} \mathbf{K}\left(\frac{\mathbf{u} - \mathbf{u}'}{h_u}\right) \frac{1}{h_i^A} \mathbf{K}\left(\frac{\mathbf{i} - \mathbf{i}'}{h_i}\right) \\ \propto & \sum_{\forall(\mathbf{u}, \mathbf{i}): (\mathbf{u}, \mathbf{i}) \in S_r} \ln \left(\frac{1}{|S_r - 1|} \sum_{(\mathbf{u}', \mathbf{i}') \in S_r \wedge (\mathbf{u}' \neq \mathbf{u}) \wedge (\mathbf{i}' \neq \mathbf{i})} \frac{1}{h_u^B} \mathbf{K}\left(\frac{\mathbf{u} - \mathbf{u}'}{h_u}\right) \frac{1}{h_i^A} \mathbf{K}\left(\frac{\mathbf{i} - \mathbf{i}'}{h_i}\right) \right) \\ = & \sum_{\forall(\mathbf{u}, \mathbf{i}): (\mathbf{u}, \mathbf{i}) \in S_r} \ln \left(\sum_{(\mathbf{u}', \mathbf{i}') \in S_r \wedge (\mathbf{u}' \neq \mathbf{u}) \wedge (\mathbf{i}' \neq \mathbf{i})} \frac{1}{|S_r - 1|} \frac{1}{h_u^B} \mathbf{K}\left(\frac{\mathbf{u} - \mathbf{u}'}{h_u}\right) \frac{1}{h_i^A} \mathbf{K}\left(\frac{\mathbf{i} - \mathbf{i}'}{h_i}\right) \right) \\ = & \sum_{\forall(\mathbf{u}, \mathbf{i}): (\mathbf{u}, \mathbf{i}) \in S_r} \ln \left(\sum_{(\mathbf{u}', \mathbf{i}') \in S_r \wedge (\mathbf{u}' \neq \mathbf{u}) \wedge (\mathbf{i}' \neq \mathbf{i})} P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) \frac{\frac{1}{|S_r - 1|} \frac{1}{h_u^B} \mathbf{K}\left(\frac{\mathbf{u} - \mathbf{u}'}{h_u}\right) \frac{1}{h_i^A} \mathbf{K}\left(\frac{\mathbf{i} - \mathbf{i}'}{h_i}\right)}{P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i})} \right) \\ \geq & \sum_{\forall(\mathbf{u}, \mathbf{i}): (\mathbf{u}, \mathbf{i}) \in S_r} \sum_{(\mathbf{u}', \mathbf{i}') \in S_r \wedge (\mathbf{u}' \neq \mathbf{u}) \wedge (\mathbf{i}' \neq \mathbf{i})} P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) \ln \frac{\frac{1}{|S_r - 1|} \frac{1}{h_u^B} \mathbf{K}\left(\frac{\mathbf{u} - \mathbf{u}'}{h_u}\right) \frac{1}{h_i^A} \mathbf{K}\left(\frac{\mathbf{i} - \mathbf{i}'}{h_i}\right)}{P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i})} \\ = & \Lambda(h_u, h_i | h_u^{(t)}, h_i^{(t)}) \end{aligned} \tag{6.49}$$

Thus we have the following lower-bound towards the log form of the cross-validated likelihood function:

$$\begin{aligned} \Lambda(h_u, h_i | h_u^{(t)}, h_i^{(t)}) &= \sum_{\mathbf{u}, \mathbf{i}} \sum_{\mathbf{u}' \neq \mathbf{u}, \mathbf{i}' \neq \mathbf{i}} P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) \ln \frac{\frac{1}{|S_r - 1|} \frac{1}{h_u^B} \mathbf{K}\left(\frac{\mathbf{u} - \mathbf{u}'}{h_u}\right) \frac{1}{h_i^A} \mathbf{K}\left(\frac{\mathbf{i} - \mathbf{i}'}{h_i}\right)}{P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i})} \\ &= \sum_{\mathbf{u}, \mathbf{i}} \sum_{\mathbf{u}' \neq \mathbf{u}, \mathbf{i}' \neq \mathbf{i}} P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) \ln \frac{1}{|S_r - 1|} \frac{1}{h_u^B} \mathbf{K}\left(\frac{\mathbf{u} - \mathbf{u}'}{h_u}\right) \frac{1}{h_i^A} \mathbf{K}\left(\frac{\mathbf{i} - \mathbf{i}'}{h_i}\right) \\ &\quad - \sum_{\mathbf{u}, \mathbf{i}} \sum_{\mathbf{u}' \neq \mathbf{u}, \mathbf{i}' \neq \mathbf{i}} P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) \ln P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) \end{aligned} \tag{6.50}$$

The last term can be dropped since it is independent of h_u and h_i :

$$\begin{aligned}
h_u^{(t+1)} &= \arg \max_{h_u} \Lambda(h_u, h_i | h_u^{(t)}, h_i^{(t)}) \\
&= \arg \max_{h_u} \sum_{\mathbf{u}, \mathbf{i}} \sum_{\mathbf{u}' \neq \mathbf{u}, \mathbf{i}' \neq \mathbf{i}} P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) \ln \frac{1}{|S_r - 1|} \frac{1}{h_u^B} \mathbf{K}\left(\frac{\mathbf{u} - \mathbf{u}'}{h_u}\right) \frac{1}{h_i^A} \mathbf{K}\left(\frac{\mathbf{i} - \mathbf{i}'}{h_i}\right)
\end{aligned} \tag{6.51}$$

Solve the maximisation problem of Eq. 6.51 by taking the derivative of Λ with respect to h_u . In the case of a Gaussian kernel, first convert the product inside the natural logarithm into a sum of $-B \ln h_u$, $-\frac{\|\mathbf{u} - \mathbf{u}'\|^2}{2h_u^2}$ and a part that does not depend on h_u :

$$\begin{aligned}
&\frac{\partial}{\partial h_u} \Lambda(h_u, h_i | h_u^{(t)}, h_i^{(t)}) \\
&= \frac{\partial}{\partial h_u} \sum_{\mathbf{u}, \mathbf{i}} \sum_{\mathbf{u}' \neq \mathbf{u}, \mathbf{i}' \neq \mathbf{i}} P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) \ln \frac{1}{|S_r - 1|} \frac{1}{h_u^B} e^{-\frac{\|\mathbf{u} - \mathbf{u}'\|^2}{2h_u^2}} \frac{1}{h_i^A} e^{-\frac{\|\mathbf{i} - \mathbf{i}'\|^2}{2h_i^2}} \\
&= \sum_{\mathbf{u}, \mathbf{i}} \sum_{\mathbf{u}' \neq \mathbf{u}, \mathbf{i}' \neq \mathbf{i}} P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) \left(\frac{-B}{h_u} + \frac{\|\mathbf{u} - \mathbf{u}'\|^2}{h_u^3} \right) \\
&= \sum_{\mathbf{u}, \mathbf{i}} \sum_{\mathbf{u}' \neq \mathbf{u}, \mathbf{i}' \neq \mathbf{i}} P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) \frac{-B}{h_u} + \sum_{\mathbf{u}, \mathbf{i}} \sum_{\mathbf{u}' \neq \mathbf{u}, \mathbf{i}' \neq \mathbf{i}} P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) \frac{\|\mathbf{u} - \mathbf{u}'\|^2}{h_u^3} \\
&= \frac{-B|S_r|}{h_u} + \sum_{\mathbf{u}, \mathbf{i}} \sum_{\mathbf{u}' \neq \mathbf{u}, \mathbf{i}' \neq \mathbf{i}} P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) \frac{\|\mathbf{u} - \mathbf{u}'\|^2}{h_u^3} = 0
\end{aligned} \tag{6.52}$$

Therefore, we have:

$$h_u^{(t+1)} = \sqrt{\frac{1}{B|S_r|} \sum_{\mathbf{u}, \mathbf{i}} \sum_{\mathbf{u}' \neq \mathbf{u}, \mathbf{i}' \neq \mathbf{i}} P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) (\|\mathbf{u} - \mathbf{u}'\|^2)} \tag{6.53}$$

Similarly, we have for h_i :

$$h_i^{(t+1)} = \sqrt{\frac{1}{A|S_r|} \sum_{\mathbf{u}, \mathbf{i}} \sum_{\mathbf{u}' \neq \mathbf{u}, \mathbf{i}' \neq \mathbf{i}} P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) (\|\mathbf{i} - \mathbf{i}'\|^2)} \tag{6.54}$$

In all, we have our cross-validated EM algorithm to estimate the two bandwidth parameters:

- E step:

$$P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) = \frac{e^{-\frac{\|\mathbf{u} - \mathbf{u}'\|^2}{2h_u^2}} e^{-\frac{\|\mathbf{i} - \mathbf{i}'\|^2}{2h_i^2}}}{\sum_{\mathbf{u}' \neq \mathbf{u}, \mathbf{i}' \neq \mathbf{i}} e^{-\frac{\|\mathbf{u} - \mathbf{u}'\|^2}{2h_u^2}} e^{-\frac{\|\mathbf{i} - \mathbf{i}'\|^2}{2h_i^2}}} \tag{6.55a}$$

- M step:

$$h_u^{(t+1)} = \sqrt{\frac{1}{B|S_r|} \sum_{\mathbf{u}, \mathbf{i}} \sum_{\mathbf{u}' \neq \mathbf{u}, \mathbf{i}' \neq \mathbf{i}} P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) (||\mathbf{u} - \mathbf{u}'||^2)} \quad (6.55b)$$

$$h_i^{(t+1)} = \sqrt{\frac{1}{A|S_r|} \sum_{\mathbf{u}, \mathbf{i}} \sum_{\mathbf{u}' \neq \mathbf{u}, \mathbf{i}' \neq \mathbf{i}} P^{(t)}(\mathbf{u}', \mathbf{i}' | \mathbf{u}, \mathbf{i}) (||\mathbf{i} - \mathbf{i}'||^2)} \quad (6.55c)$$

Commentary on Chapter 6

Making use of the additional ratings from similar users for similar items solves the data sparsity problem to some extent. But like any other user profile based collaborative filtering approach, the unified model requires a user to provide item ratings before it can make any *personalized* recommendation for that user [94]. Similarly, a new item cannot be recommended until its ratings have been collected. In this regard, recommender systems need bootstrapping, especially in their early stage. Making an initial recommendation on the basis of item popularity could motivate new users to interact with the system. We have shown in Chapters 2 and 3 that the item-based relevance-ranking algorithms have already incorporated this into the recommendation process. Unfortunately, the acquisition of user preferences may be time-consuming and annoying. Thus, it is desirable to explore active learning techniques in which most informative item can be identified, and asking new users to rate only these items could minimize the number of rating requests [49, 65].

Fig. 6.1 (b) illustrates that, to be able to make a recommendation of new items, we require a feature representation of these items so as to correlate them to the existing items in the collection. Content descriptions are helpful but they are difficult to obtain for non-textual items such as images and videos and subject indexing that manually describes or summarizes these information items by index terms is not cost-effective. As we have seen in Chapter 4, collaborative tagging alleviates the problem by collaboratively annotating and categorizing content from millions of online users.

Part III

Applications

Peer-to-Peer Recommendation

The advent of Peer-to-Peer (P2P) networks has changed the way in which people share and seek information. Given the large amount of information available, it is of great interest to design a distributed recommender system, so as to personalize the information seeking in these networks. However, when applying the proposed relevance models of collaborative filtering to P2P networks, we are confronted with a challenge that there is no central server managing the network. This chapter combines work in the field of information retrieval and that of P2P networks, approaching distributed collaborative filtering in an interdisciplinary manner. To achieve this, we introduce an overlay network that efficiently decomposes the computation loads and preference data for the relevance models. We argue that, with respect to the two factorizations of the relevance models, there are two views on approaching an overlay network, namely the user-oriented view and item oriented view. For the user-oriented view, we

This chapter is based on the following published papers [117, 118, 119, 124]:

- “Wi-Fi walkman: a wireless handheld that shares and recommends music on peer-to-peer networks”, J. Wang, M. J. Reinders, J. Pouwelse, and R. L. Lagendijk, in Proc. of Embedded Processors for Multimedia and Communications II, part of the SPIE Symposium on Electronic Imaging 2005.
- “Distributed collaborative filtering for peer-to-peer file sharing systems”, J. Wang, J. Pouwelse, R. Lagendijk, and M. R. J. Reinders, in Proc. of the 21st Annual ACM Symposium on Applied Computing, 2006.
- “Personalization on a peer-to-peer television system”, J. Wang, J. Pouwelse, J. Fokker, A. P. de Vries, and M. J. Reinders, Special Issue on Multimedia Tools and Applications, 2006.
- “An epidemic-based P2P recommender system”, J. Yang, J. Wang, M. Clements, J. A. Pouwelse, A. P. de Vries, and M. Reinders, in Workshop on Large Scale Distributed Systems for Information Retrieval (LSDS-IR) in SIGIR07, 2007.

take user proximity into account, proposing an user-preference exchange algorithm on the basis of an epidemic protocol. For the item-oriented view, we employ item proximity, introducing an item similarity update scheme that is self-organizing and operates in a distributed way. Our experiments show that the proposed overlay networks provide effective yet simple ways to distribute the computation that is needed for the recommendations. To conclude the chapter, we introduce two P2P applications, demonstrating the practical usage of our algorithms.

7.1 Introduction

The rapid progress in multimedia processing, communications, and storage technologies not only changes the availability of data, but also the way in which people interact with it. Particularly, peer-to-peer (P2P) networks have become a new and popular medium for people to exchange information, stored on their local devices. Examples of P2P file sharing systems include: Emule (Emule-Project.net) and BitTorrent (BitTorrent.com). These networks increase content availability dramatically, since the involvement of third parties that manage the centrally stored information can be avoided.

A schematic overview of a P2P file sharing system is given in Fig. 7.1, showing that each peer performs two roles in the network: information seeker and sharer. The overwhelming information availability in most P2P networks causes the information seeking task to be difficult and time consuming. A filter based on the peer's preference offers a solution to extract interesting content from the wealth of data in the network. Another problem in P2P networks lies in the fact that both users and content are distributed and dynamically changing, these characteristics make the design of content filters or search engines a challenging task.

To manage the poorly organized information and filter relevant content that fits the user's interest within a P2P network, techniques to create a semantic overlay need to be explored. P2P research has proposed semantic structures [17, 105, 113], that deploy the similarity between content descriptions or user demographics. In practice, the use of meta data descriptions is problematic, since it is often unavailable or incomplete. Another way to describe the content is by making use of low-level features of the data, such as proposed in content-based multimedia analysis [34, 101]. However, these approaches often require that the data to be compared is from the same modality, which is an undesirable property when one wants to share *multimedia* files. Thus, there is a need for a more generic content similarity measure that does not demand meta data and/or low level features.

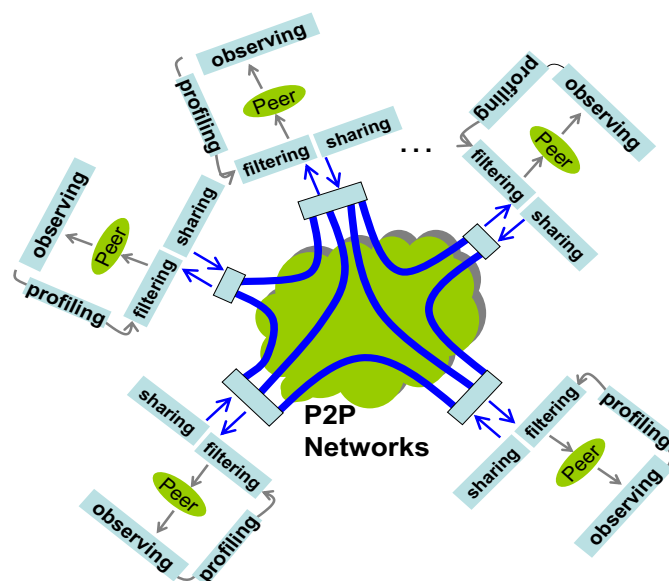


Figure 7.1: A schematic overview of a peer to peer file sharing system.

Research on recommender systems has provided similarity measures that can be used to derive the relevance of an item to the preference of a user. Many current recommender systems are based on collaborative filtering, a filtering technique that derives similarities between users (user-based) or items (item-based) from a database of the users' rating or viewing profiles. Within the context of P2P networks there is, however, no centralized user profile database, so that current collaborative filtering approaches cannot be applied directly.

This chapter presents our research on distributed collaborative filtering. We aim at decentralizing the proposed relevance models of collaborative filtering so that they can be readily applied to P2P environments. Recall that we introduced two different factorizations in the relevance model, and their differences lead to two different ways to approaching collaborative filtering. For each of the factorizations (i.e. user-based and item-based), this chapter respectively introduces its corresponding overlay network, efficiently distributing computation loads and user preference data into the whole network.

The remainder of the chapter is organized as follows. We first summarize related work and introduce our two decentralized recommendation approaches. We then look at two applications that we worked on: *Tribler*, a P2P file sharing system and *Wi-Fi Walkman*, a P2P Music sharing handheld device. Finally, a conclusion is given with discussion on future directions.

7.2 Related Work

7.2.1 Collaborative Filtering

Recently, a few early attempts towards decentralized collaborative filtering have been introduced [10, 71, 74, 76, 111]. In [71], five architectures are proposed to find and store user rating data to make rating based recommendation, namely, a central server, random discovery similar to **Gnutella**, transitive traversal, Distributed Hash Tables (DHT), and secure Blackboard. These solutions aim to aggregate the rating data in order to make a recommendation and they hold independently of any semantic structure of the networks. This inevitably increases the amount of traffic within the network.

Different from these methods, we implicitly learn user interest from user interaction data. We build our semantic overlay network from either the user-oriented view or item-oriented view, making any static structures unnecessary.

7.2.2 Peer-to-Peer Networks

Different indexing techniques for content located at different peers exist, such as: a local index (the owner of the data is only able to index the data, like in early **Gnutella**), the central index (a centralized server organizes indices to data residing at peers, like in **Napster**), and the distributed index (other peers are also able to index the data residing at a peer, like in **Freenet**). For a recent comprehensive survey on P2P networks we refer to [64].

Jelasity and Van Steen [47] introduced newscast, an epidemic (or gossip) protocol that exploits randomness to disseminate information without keeping any static structures or requiring any sort of administration. Although this type of protocol successfully operates in dynamic networks, its lack of content-awareness restricts its efficiency towards content search.

Another major indexing approach makes use of DHTs [64, 106]. In a DHT each location (index) is mapped to a unique key, and each peer maintains a certain range of the keys. In this way each peer generates a well-defined structure that can be used for routing queries that is scalable to some extent. However, an extension is necessary to perform a search based on arbitrary queries, rather than key lookups [64].

Recently, semantic indexing and routing techniques have been proposed to capture relationships between content [64, 107]. Distinct semantic groups of documents [17], or users [105, 113] are identified to create Semantic Overlay Networks (SONs). A document request is then handled by the overlay to which this document presumably belongs (based on either clusters of documents or peers).

However, to match queries to documents, a content description (in the form of meta-data) is required. Identifying similarities between peers or non-textual content turns out to be difficult to establish in the absence of this information.

In this chapter, meta-data and demographics are redundant, as user similarity is calculated by the co-occurrence of two users' preference profiles and a semantic overlay is created among users by randomly exchanging so-called *Buddycast* messages. We shall see that such an overlay network lies at the heart of the item ranking in P2P networks as it provides an effective way to collect the most valuable profiles for the purpose of ranking.

7.3 User-oriented Overlay Network¹

In this section, we first describe a user-based item ranking model for recommendation. Taking user proximity into account, we propose an epidemic-based algorithm for exchanging user profiles, and realize the ranking in a distributed and dynamic manner. The proposed distributed recommendation algorithm has been implemented in Tribler, an Open Source file sharing software [79]. For readability, we leave the introduction of the Tribler system for Section 7.5.1.

7.3.1 User-based Ranking Model

Our task for collaborative filtering is to find items that are relevant (useful) to a given user (his or her interest is implicitly indicated by a user profile). The well-known Probability Ranking Principle (PRP) of information retrieval [83] states that ranking documents by their probability of relevance in descending order produces “optimal” performance under reasonable assumptions [83]. Thus, it is natural to adopt the PRP for our recommendation task, treating the user profile as a query and rank items. For this, we need to introduce the concept of “relevancy” into collaborative filtering. By analogy with the relevance models in text retrieval [56, 86], the top- N recommendation items can be then generated by ranking items in order of their probability of relevance to the user profile or the underlying user interest.

To be able to rank items, this section formulates the estimation of the probability of relevance between an item and a user (profile), using the following notation. Let u be a discrete random variable over the sample space of users² $\Phi_U = \{1, \dots, M\}$, let i be a random variable over the sample space of items

¹This section is based on the publication: “An epidemic-based P2P recommender system”, J. Yang, J. Wang, M. Clements, J. A. Pouwelse, A. P. de Vries, and M. Reinders, in Workshop on Large Scale Distributed Systems for Information Retrieval (LSDS-IR) in SIGIR07, 2007.

²Throughout the chapter, we use the terms peer and user interchangeably.

$\Phi_I = \{1, \dots, K\}$, and let R be a random variable over the relevance space Φ_R , where R is either ‘relevant’ r or ‘non-relevant’ \bar{r} .

In a probabilistic framework, we can generate a top-N item ranking list in order of their estimated log-odd of relevance: $\ln \frac{p(r|u,i)}{p(\bar{r}|u,i)}$. Using Bayes’ rule gives the following ranking formula:

$$o_u(i) = \ln \frac{p(r|u,i)}{p(\bar{r}|u,i)} = \ln \frac{p(i|u,r)}{p(i|u,\bar{r})} + \ln \frac{p(r|u)}{p(\bar{r}|u)} \quad (7.1)$$

where the last term can be discarded as it is independent on the target item. Notice that this is not the only derivation. For other detailed derivations, please refer to Chapter 1 or [114].

The relevancy between items and users can be explicitly obtained by asking users to rate items (content) that they know. However these explicit ratings are hard to gather in a real system. It is highly desirable to infer user profiles from implicit observations of user interactions with the system. In our system, for the sake of simplicity but without loss of generality, we only observe the positive evidence. By following the language modelling of information retrieval [56], we now assume equal priors for item i in the non-relevant case; Notice that these two negative terms in Eq. 7.1 can always be added to the model when the negative evidences are captured. Then, the non-relevance term can be removed and the ranking formula becomes:

$$o_u(i) \propto p(i|u,r) \quad (7.2)$$

where the probability $p(i|u,r)$ cannot be directly estimated because we need to predict “new” items, i.e., those that do not exist in the given user profile. To address this problem, we represent items explicitly by the judgments of the different users, such that they can be linked to the target user u . Formally, we introduce a list L_i for each item i , where $u \in L_i$ denotes that user u is in the list. This list enumerates the users who have expressed interest in the item i . Replacing i with L_i , we have:

$$o_u(i) \propto \sum_{\forall u': u' \in L_i} \log p(u'|u,r) \quad (7.3)$$

where the probability $p(u'|u,r)$ depicts the similarity between two users u and u' , which can be estimated using user profiles: counting the number of items that both users liked (denoted as $c(u',u)$), divided by the total number of items that user u liked (denoted as $c(u)$):

$$p(u'|u,r) = \frac{p(u',u|r)}{p(u|r)} := \frac{c(u',u)}{c(u)} \quad (7.4)$$

However due to the data sparsity, many co-occurrence counts of two users may be zero. To counter the sparsity and remove the zero probabilities, we propose

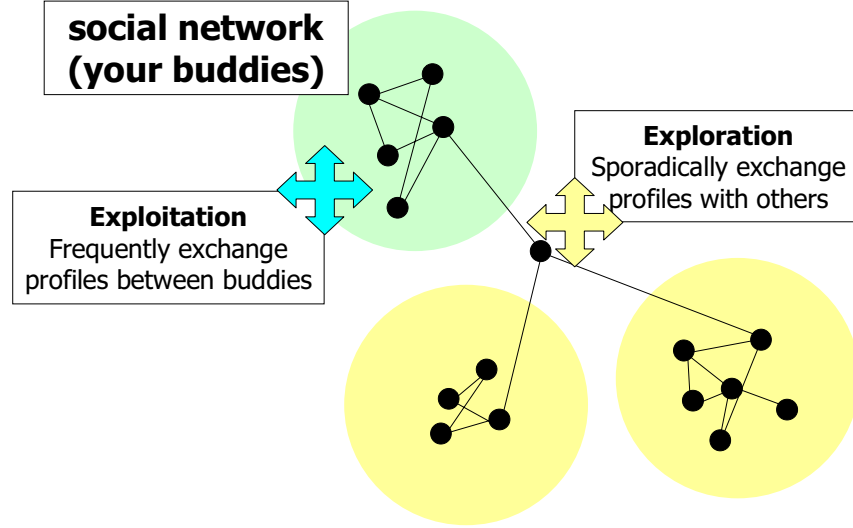


Figure 7.2: Exploitation v.s. Exploration.

to use the Bayes-smoothing technique [126] to further smooth the estimation. More formally, we have:

$$p(u'|u, r) := \frac{c(u', u) + \mu \cdot p(u'|r)}{c(u) + \mu} \quad (7.5)$$

where μ is the smoothing parameter and $p(u'|r) := \frac{c(u')}{\sum_{u'} c(u')}$.

In summary, we now have our final ranking formula if we replace Eq. 7.5 into Eq. 7.3.

$$o_u(i) \propto \sum_{\forall u': u' \in L_i} \log \frac{c(u', u) + \mu \cdot \frac{c(u')}{\sum_{u'} c(u')}}{c(u) + \mu} \quad (7.6)$$

7.3.2 Decentralized Ranking

It is easy to compute the ranking in a centralized server but it is non-trivial in a P2P network because user profiles are distributed in the entire P2P network and for a particular peer (user), there is no centralized user profile database to be used to calculate the co-occurrences. Thus the ranking of items for a particular user (peer) has to be calculated locally in that peer.

Eq. 7.6 gives a formal ranking solution for collaborative filtering, indicating that, in order to make a recommendation, i.e. ranking the relevance of a target

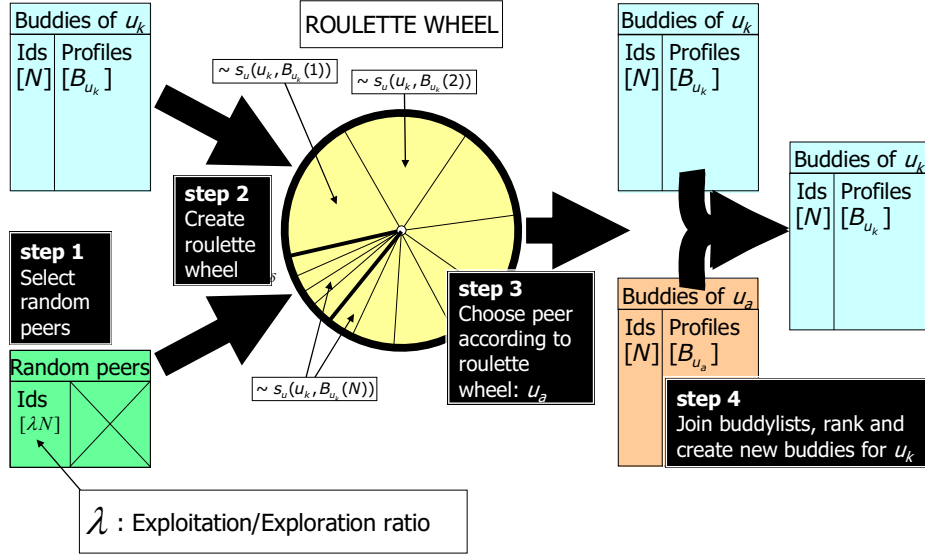


Figure 7.3: Peer Selection.

item towards the user, we mainly need to estimate the co-occurrence $c(u', u)$ of the target user u towards other users u' who have already expressed interest in the target item. Closely looking at Eq. 7.6, we find that, for the target user, those items that have been liked by the similar users will have high ranking scores when we rank the relevance to that target user. Thus an efficient and scalable way to calculate the ranking formula in Eq. 7.6 is to find the similar users to the target user and only rank items that they liked, rather than collecting *all* user profiles and ranking *all* items.

In this regard, we introduce the Buddycast algorithm, which is used to efficiently exchange the profiles of similar users. The Buddycast algorithm is based on an epidemic protocol [47] and works as follows (see Fig. 7.4). Each peer maintains a list of its top- N most similar peers along with their current preference lists. Similarities between preference lists are measured using the co-occurrence $c(u', u)$. Periodically, a peer connects to either (a) one of its buddies to exchange social networks and preference lists (exploitation), or (b) to a new peer, randomly chosen, to exchange this information (exploration). This is illustrated in Fig. 7.2. To maximize the exploration of the social network, every peer also maintains a list with the K most recently visited random peers, and avoids reconnecting to a peer already present in the list.

In contrast to other epidemic protocols such as Newscast [47], we use both exploitation and exploration branches, we limit the randomness of peer selection during the exploration, and we implicitly cluster peers into social groups

```

1  BuddyCast( $p_i$ ) {
2    do forever {
3       $e = \text{waitForEvent}()$ ;
4      if  $e$  is TIMEOUT {
5         $p_j = \text{selectPeer}(B_i)$ ;
6        send  $B_i$  to  $p_j$ ;
7        receive  $B_j$  from  $p_j$ ;
8         $B_i = \text{merg}(B_i; B_j)$ ;
9         $B_i = \text{updateSim}(B_i)$ ;
10        $B_i = \text{selectTopN}(B_i)$ ;
11     }
12     if  $e$  is message  $B_j$  from  $p_j$  {
13       send  $B_i$  to  $p_j$ ;
14        $B_i = \text{merg}(B_i; B_j)$ ;
15        $B_i = \text{updateSim}(B_i)$ ;
16        $B_i = \text{selectTopN}(B_i)$ ;
17     }
18   }
19 }

```

(a) Main Routine

```

1  selectPeer( $B_i$ ) {
2     $\{p_\lambda\} = \text{randomPeers}(p_i, \lambda \cdot N)$ ;
3     $\{w_\lambda\} = \text{minWeight}(B_i)$ ;
4     $B_i = B_i + \{(p_\lambda, w_\lambda)\}$ ;
5     $B_i = \text{normalizeWeight}(B_i)$ ;
6    randomly select  $p$  according to  $w$ .
7  }

```

(b) Peer Selection

Figure 7.4: The BuddyCast Algorithm. λ is the exploration/exploitation ratio, B_i is the Buddy Cache.

(having common interest). To find a good balance between exploitation and exploration, the following procedure is adopted (see Fig. 7.4 (b)). First, $\lambda \cdot N$ random peers are chosen, where $\lambda \geq 0$ is the exploitation-to-exploration ratio. Then, these random peers are joined with the N buddies in a single ranked list, with the random peers being assigned the lowest ranks. Then, one peer is randomly chosen from this ranked list according to a roulette wheel approach (probabilities proportional to the ranks), which gives taste buddies a higher probability of being selected than the random peers. Once a peer has selected some other peer, the buddy lists of the two peers are merged. The first peer then ranks the composite list according to the preference list similarities with its own preference list, and retains only the top- N best ranked peers. The peer selection step is illustrated in Fig. 7.3.

After collecting similar user profiles, each peer applies the ranking formula in Eq. 7.6 using the calculated co-occurrences to rank the items that are presented in these profiles.

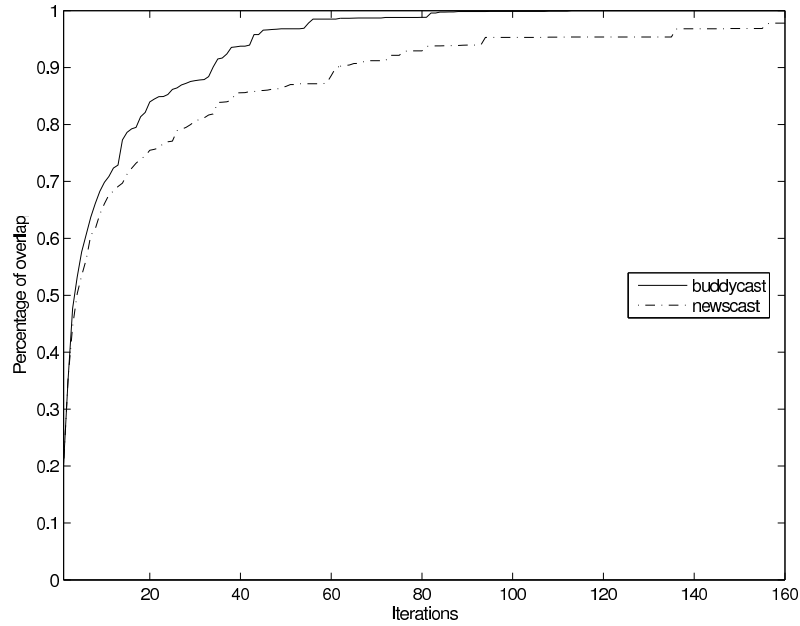


Figure 7.5: Convergence of our Buddycast algorithm.

Practical Considerations The proposed P2P recommendation algorithm has been used in our P2P file sharing software Tribler since March 2006, which has been downloaded more than 100,000 times. For such a real P2P environment, we have addressed some of the practical issues. Firstly, we establish a “strong” identifier for each peer since we do not have a central indexing server to index online users and their network addresses (IP addresses). Tribler creates a public key as a permanent identifier for each peer (client). Secondly, the network is very dynamic. Peers arrive and leave frequently. According to our observation, there are about 1/3 of the peers behind firewalls. To avoid including dead or unreachable peers in the Buddycast messages, each peer always maintains an online peer list, checking their connectability and keeping connected with those peers that have been contacted recently. Thirdly, we limit our storage of user profiles by removing dissimilar and old peers.

Our algorithm is scalable because each peer only keeps a small view of the whole network in which he or she is interested. Thus, there is less local storage required for each peer and therefore the computation is also reduced. In addition, the algorithm can cope with the dynamic nature of the network as our recommendation can update its results once more peers and their preferences are discovered.

Evaluation We have emulated our Buddycast algorithm using a cluster of the DAS-2 system (`asci.tudelft.nl`). The network consisted of 480 peers

distributed uniformly over 32 nodes. We used a data set of TV watching habits of 480 users from the SKO foundation (kijkonderzoek.nl) as rating data (users watched or did not watch TV programs). Each peer maintained a list of 10 taste buddies ($N = 10$) and the 10 last visited peers ($K = 10$). The system was initialized by giving each peer a random other peer. The exploitation-to-exploration ratio, r , was set to 1.

Figure 7.5 compares the convergence of Buddycast to that of Newscast (randomly select connecting peers, i.e., $r \rightarrow \infty$). After each update we compared the list of top- N taste buddies with a pre-compiled list of top- N taste buddies generated using all data (centralized approach). In Figure 7.5, the percentage of overlap is shown as a function of time (represented by the number of updates). The figure shows that the convergence of Buddycast is much faster than that of the Newscast approach.

7.4 Item-oriented Overlay Network³

In this section, we take an item-oriented view, introducing our self-organizing distributed relevance model. We first introduce an item-based relevance model between an user and an information item. We then introduce a method to update these relevance ranks in a distributed and dynamic way. Finally, we present how to make recommendations based on this relevance model.

7.4.1 Item-based Ranking Model

We consider the following formal setting: Multimedia files (e.g. image, movie, or audio files) are represented as a set of *items*. There are N items, denoted as I_a , $a = 1, \dots, N$, distributed throughout the network, and M users, denoted as P_i , $i = 1, \dots, M$.

We adopt a probabilistic relevance model proposed earlier for text retrieval ([56, 78]). Since a user profile represents the current interest of that user, we can treat a user profile as a query and introduce random variables r and \bar{r} to denote whether an item is “relevant” or “irrelevant” for the user. Note this treatment is only valid for the filtering problem. For the retrieval problem, other than the user profile, content examples or keywords should also be presented to give a more specific query. To avoid estimating $P(I_a, P_i)$, the relevance rank (denoted as R_{I_a, P_i}) of the item I_a for a peer P_i can be formulated as the odds

³This section is based on the publication: “Distributed collaborative filtering for peer-to-peer file sharing systems”, J. Wang, J. Pouwelse, R. Lagendijk, and M. R. J. Reinders, in Proc. of the 21st Annual ACM Symposium on Applied Computing, 2006.

of relevance:

$$R_{I_a, P_i} = \log \frac{P(r|I_a, P_i)}{P(\bar{r}|I_a, P_i)} \quad (7.7)$$

By factorizing $P(\bullet|I_a, P_i)$ with $\frac{P(P_i|I_a, \bullet)P(\bullet|I_a)}{P(P_i|I_a)}$, the following log-odds ratio can be obtained:

$$R_{I_a, P_i} = \log \frac{P(r|I_a, P_i)}{P(\bar{r}|I_a, P_i)} = \log \frac{P(P_i|I_a, r)}{P(P_i|I_a, \bar{r})} + \log \frac{P(r|I_a)}{P(\bar{r}|I_a)} \quad (7.8)$$

Hence, the evidence for the relevance of an item is based on both the positive evidence (indicating the relevance) as well as negative evidence (indicating the irrelevance). In our user profiling method, for the sake of simplicity but without loss of generality, we only observe the positive evidence. By following the language modelling approach ([56]), we now assume that 1) P_i and I_a are assumed independent in the irrelevant case (\bar{r}), i.e. $P(P_i, |I_a, \bar{r}) = P(P_i|\bar{r})$; and, 2) equal priors for both P_i and I_a , given that the item is irrelevant. Then the two irrelevance terms can be removed and the relevance rank becomes:

$$R_{I_a, P_i} \propto \log P(P_i|I_a, r) + \log P(r|I_a) \quad (7.9)$$

Note that these two negative terms in Eq. (7.8) can always be added to the model when the negative evidences are captured.

7.4.1.1 Incorporation of User Profiles

The interest of users towards the available multimedia content is represented by user profiles. User profiles can be obtained by either explicitly asking users to rate the content or implicitly by observing the interactions of a user with the content. Since previous research ([15]) has shown that users are very unlikely to provide an explicit rating because they find this annoying, we adopt an implicit approach (as shown in Fig. 7.1).

The items that a user previously interacted with (e.g. watched, played, or downloaded) represent positive evidence for the interest of the user. For example, one could use the download action of the users as an implicit interest indicator. Thus, items that are downloaded are stored into a *download list* which represents the user profile. Let L_i denote the download list of user P_i . $L_i(I_b) = 1$ (or $I_b \in L_i$) indicates that item I_b , $b = 1, \dots, N$, is in the list while $L_i(I_b) = 0$ (or $I_b \notin L_i$) otherwise.

We assume that items are conditionally independent from each other given that they are downloaded by the same user. Although this naive Bayes assumption does not hold in many real situations, it has been empirically shown to be a

competitive approach (e.g. in text classification domain ([27]) as well as in our experiments (see Sec. 7.4.4)). Then Eq. (7.9) becomes:

$$R_{I_a, P_i} \propto \sum_{\forall I_b: I_b \in L_i} \log P(I_b|I_a, r) + \log P(r|I_a) \quad (7.10)$$

When applying the Bayes rule once more:

$$R_{I_a, P_i} \propto \sum_{\forall I_b: I_b \in L_i} \log \frac{P(r|I_a, I_b)}{P(r|I_a)} + \log P(r|I_a) \quad (7.11)$$

Eq. (7.11) shows that, in order to make a recommendation (ranking the relevance of a target item towards the user profile), we need to estimate the prior probability of the relevance of an item, $P(r|I_a)$, as well as the probability of the relevance between two items $P(r|I_a, I_b)$, $a, b \in \{1, \dots, N\}$.

Different from the previous item-based collaborative filtering techniques ([52, 61]), we use a probabilistic framework to convert the initial relevance rank between user and item into a relevance rank between items. By doing so, the prior probability of the item (i.e. the popularity of item) is systematically incorporated into the model. Thus, our model can overcome the disadvantage that it tends to recommend the most popular items only in the traditional item based collaborative filtering [52, 61, 71],

In the following, we propose to build a scalable, efficient, and fully distributed and dynamic approach to estimate these relevance probabilities.

7.4.2 Self-organizing Distributed Buddy Tables

In this section, first, we propose a dynamic approach to update the relevance probabilities. Then, we introduce the *buddy table* that stores these relevance ranks and relevance links in a distributed way.

7.4.2.1 Dynamically Updating Relevance Ranks

Unlike content-based analysis [34, 101] where relevance (or similarity) between items is obtained by using low-level features, here we adopt user profiles to infer the item similarities (see item-based collaborative filtering [52, 61]).

If a user likes two items I_a and I_b , then this increases the relevance (similarity) of item I_a with respect to item I_b (and vice versa). Consequently, in a centralized situation (i.e. when all user profiles are available), the relevance probability of item I_a and I_b and the prior relevance probability of an item can be calculated

from the profiles of all users (see also, [52]):

$$P(r|I_a, I_b) = (\sum_{i=1}^M L_i(I_a) \cap L_i(I_b)) / M, \quad (7.12)$$

$$P(r|I_a) = (\sum_{i=1}^M L_i(I_a)) / M \quad (7.13)$$

where $\sum_{i=1}^M L_i(I_a) \cap L_i(I_b)$ is the number of times that items I_a and I_b appear in the same download list (i.e. the co-occurrence frequency between items). This item-based similarity measure is computationally inexpensive and generic (i.e. for different media formats).

In a P2P network the user profiles are, however, distributed throughout the entire network. In the previous section, the user-based overlay, we need to contact only a subset of good user profiles, but for an item-based overlay, we would have to visit all peers to obtain a similar subset of all item profiles. A naive way to collect user profiles is to broadcast the user profiles throughout the P2P network as in [111, 71] or using DHTs to map keys to profiles [76]. Obviously, this is not efficient. We have found the solution in *dynamically* updating these relevance probabilities.

At a given moment in time, the relevance between two items is updated according to:

$$P_t(r|I_a, I_b) = P_{t-\Delta t}(r|I_a, I_b) + \Delta P_t(r|I_a, I_b) \quad (7.14)$$

where $\Delta P_t(r|I_a, I_b)$ is the update of the relevance between two items from time $t - \Delta t$ to t . The update is only non-zero when there is a user that downloads one of the items I_a or I_b while that user in the past already expressed interest in the opposite item (listed in the download list). Hence, relevance updates only occur when multimedia files are downloaded. The relevance update can thus be expressed in terms of the transactions that take place between $t - \Delta t$ and t , i.e.:

$$\Delta P_t(r|I_a, I_b) = \sum_{\forall T_k: t-\Delta t < k < t} \Delta P_k(r, T_k|I_a, I_b) \quad (7.15)$$

where T_k denotes the download transaction at time k , and the notation $\Delta P_k(r, T_k|I_a, I_b)$ represents the relevance update between items I_a and I_b when considering transaction T_k .

Since the relevance between I_a and I_b is not changed when considering a transaction that does not involve the downloading of either two items, the relevance

| Rank | ItemID | Owner | Location | Meta data(Optional) |
|------|----------|-------|---------------------------------|------------------------|
| 0.88 | I_{25} | 100 | c:/upload/...@P ₁₀₀ | All My Loving-Beatles |
| 0.80 | I_{36} | 6 | d:/mystuff/...@P ₆ | Take It Easy-Eagles |
| 0.78 | I_7 | 5 | d:/sharefile/...@P ₅ | Lost in Love-AirSupply |
| ... | ... | ... | ... | ... |

Table 7.1: A hypothetical example of a buddy table for an item: a song track {Miss You - RollingStones}.

update can be simplified to:

$$\begin{aligned}
\Delta P_t(r|I_a, I_b) = & \\
& \sum_{\forall T_k: t-\Delta t < k < t} \Delta P_k(r, I_a = \text{item}(T_k), I_b \in L_{\text{peer}(T_k)} | I_a, I_b) + \\
& \sum_{\forall T_k: t-\Delta t < k < t} \Delta P_k(r, I_b = \text{item}(T_k), I_a \in L_{\text{peer}(T_k)} | I_a, I_b)
\end{aligned} \tag{7.16}$$

and

$$\begin{aligned}
\Delta P_k(r, I_b = \text{item}(T_k), I_a \in L_{\text{peer}(T_k)} | I_a, I_b) = \\
\Delta P_k(r, I_a = \text{item}(T_k), I_b \in L_{\text{peer}(T_k)} | I_a, I_b) = 1/M
\end{aligned} \tag{7.17}$$

where $\text{item}(T_k)$ indicates the item being downloaded in transaction T_k and $\text{peer}(T_k)$ indicates the peer that performs the download. $I_b \in L_{\text{peer}(T_k)}$ means item I_b is in the downloading list of the peer that performs the download. Eq. (7.17) indicates that – when I_a is downloaded by a peer ($\text{peer}(T_k)$) with I_b in the download list ($L_{\text{peer}(T_k)}$) (or vice versa) – the relevance between item I_a and I_b increases $1/M$. This is because there is one more download list in which the item I_a and I_b both exist together over all the download lists.

A further investigation of Equation (7.16) shows that the relevance between two items can be updated using only the information about the item that is being downloaded ($\text{item}(T_k)$) and the user profile of the peer that is downloading the item (e.g. $L_{\text{peer}(T_k)}$). Now we show how to store these between-item relevances and the prior relevances in a *distributed* way.

7.4.2.2 Distributed Item-to-Item Relevance Ranking

Equations (7.15) and (7.16) show that the between-item relevance probabilities can be calculated incrementally. To store the relevance ranks in a fully distributed way, we propose to store the between-item relevance ranks locally at the location of both items. This is realized by attaching to each item a so called *buddy table*, which is denoted as B_{I_a} for item I_a .

Table 7.1 shows an example of such a buddy table. The buddy table stores the information (including an index to their location) about the *top-N* relevant items. The location information can be used to locate items when these items are being recommended to a peer. Most importantly, the buddy tables automatically create a self-organizing semantic overlay that implicitly cluster similar multimedia files (see Sec. 7.4.4)).

The relevance ranks stored in the buddy table can be updated according to the following strategy. For each transaction T_k , the buddy table of the item that is being downloaded ($I_a = \text{item}(T_k)$), is updated, based on the user profile of the peer that performs the download ($L_{\text{peer}(T_k)}$). For all items in that user profile, $\forall I_b : I_b \in L_{\text{peer}(T_k)}$, the between-item relevance ranks recorded in the buddy table are updated: $R_{I_a}(I_b) = R_{I_a}(I_b) + 1/M$. The prior relevance rank is also updated: $R_{I_a} = R_{I_a} + 1/M$.

The overall protocol for updating buddy tables is illustrated in Fig. 7.6. It can be simply run as a *daemon* program in each peer that is only activated in two events: 1) UPLOAD request from other peers (leading to an update of the buddy table of the requested item); and 2) DOWNLOAD request from the local user (leading to sending the user profile to the peer that owns the downloaded file). The protocol is scalable, and simple to implement even on small network-enabled computing devices.

Note that the privacy can be preserved since user profiles are only used anonymously to update the buddy tables. They do not exist in the remote peer after the updating. Moreover, the entire update procedures take advantage of the connections for downloading items, no extra communication connection are required.

Eq. (7.16) shows that the relevance probability between item I_a and I_b only needs to be updated in two situations: either item I_a is downloaded while I_b is in the list, or, vice versa. Careful investigation of the buddy table update protocol shows that the update of this relevance probability is stored in the buddy table of item I_a while I_a is being downloaded and stored in the buddy table of item I_b when I_b is downloaded. That is, the accumulated relevance probability between the two items in a given time is equal to the sum of the two relevance ranks from the buddy tables of *both* items at that time:

$$P_t(r|I_a, I_b) = R_{I_a}(I_b) + R_{I_b}(I_a) \quad (7.18)$$

To increase the efficiency and to minimize the communication between the peers, we would like to use *only one* of the relevance ranks stored in either of the two buddy tables to approximate this accumulated between-item relevance probability. Therefore, we make use of the following proposition.

```

1  do forever {
2     $e = \text{waitForEvent}();$ 
3    /*UPLOAD request from another peer*/
4    if  $e$  is UPLOAD request for  $I_a$  from remote peer  $P_j$  {
5      upload  $I_a$  to  $P_j$ ;
6       $R_{I_a} = R_{I_a} + 1$ ; /*Update the prior probability*/
7      get  $L_j$  from  $P_j$ ; /*Receive the user profile*/
8      for  $b = 1:N$  {
9        if  $L_j(I_b) == 1$  {
10         /*Update between-item relevance*/
11          $R_{I_a}(I_b) = R_{I_a}(I_b) + 1/M$ ;
12       }
13     }
14   }
15   /*DOWNLOAD request from local peer*/
16   if  $e$  is DOWNLOAD request for  $I_b$  in remote peer  $P_j$  from local user  $P_i$  {
17     download  $I_b$  and  $B_{I_b}$  from  $P_j$ ;
18     send  $L_i$  to  $P_j$ ; /*Send user profile anonymously*/
19      $L_i = L_i \cup I_b$ ; /*Add the item into download list*/
20   }
21 }

```

Figure 7.6: A *Demon* program running in each peer P_i for updating the buddy tables of its own items.

Proposition 7.4.1 *If items in one download list are conditionally independent to each other, then:*

$$P_t(r|I_a, I_b) = 2 * R_{I_a}(b) = 2 * R_{I_b}(I_a)$$

With this proposition the accumulated relevance probability between the two items can be calculated using only the information stored in one of the buddy tables. The proof follows:

Recall that we have a naive Bayesian assumption in Eq. (7.10) that items in one download list are conditionally independent to each other. This implies that the *order* in which items are being downloaded is arbitrary. This can be expressed as follows:

$$\begin{aligned}
& P(I_a = \text{item}(T), I_b \in L_{\text{peer}(T)} | I_a, I_b) \\
& = P(I_b = \text{item}(T), I_a \in L_{\text{peer}(T)} | I_a, I_b) = 0.5
\end{aligned} \tag{7.19}$$

where T represents all transactions.

On the other hand, the relevance rank in the buddy tables of an item ($R_{I_a}(I_b)$) is equal to the joint probability between the relevance and the transaction for downloading that item: $R_{I_a}(I_b) = P_t(r, I_a = \text{item}(T), I_b \in L_{\text{peer}(T)} | I_a, I_b)$ (See Eq. (7.16) and (7.18)). Since these two events are independent from each other,

we can factorize the probability as follows:

$$\begin{aligned} R_{I_a}(I_b) &= P_t(r, I_a = \text{item}(T), I_b \in L_{\text{peer}(T)} | I_a, I_b) \\ &= P_t(r | I_a, I_b) P(I_a = \text{item}(T), I_b \in L_{\text{peer}(T)} | I_a, I_b) \end{aligned} \quad (7.20)$$

From Eq. (7.19), we know that $P(I_a = \text{item}(T), I_b \in L_{\text{peer}(T)} | I_a, I_b) = 0.5$. We then can derive the final result from the above equation:

$$P_t(r | I_a, I_b) = 2 \cdot R_{I_a}(I_b) \quad (7.21)$$

Similarly, we can also derive $P_t(r | I_a, I_b) = 2 \cdot R_{I_b}(I_a)$.

Caching: In practice, it is not necessary to maintain a large number of relevance ranks in the buddy tables. We can cache the recent updates of the relevance ranks while keeping only the *top-N* highest ranked items in the buddy tables. An item may move from the cache to the buddy table when its relevance rank is higher than the relevance rank of the *N*th item in the buddy. Conversely, an item may move from the buddy table to the cache when its relevance rank drops below the relevance rank of the *N*th item in the buddy table. During recommendation, only the highest ranked items are enough to generate a recommendation. This is shown in the experiments section.

Item availability: One of the characteristics of P2P networks is that peers are frequently not online. This causes: 1) The items stored at these peers are then also not accessible. The *top-N* highest ranked items in the buddy table can be periodically screened for item availability so that no unavailable items will be recommended. 2) The locations stored in the buddy table may become invalid when the host peers change their ip addresses. This can be solved by applying the recent DHT techniques ([106]) to *lookup* from an item *Key* (stored in buddy tables) to its location.

7.4.3 Distributed Recommendation

By using the relevance ranks stored in the buddy tables a recommendation can be generated as follows:

$$R_{I_a, P_i} \propto \sum_{\forall I_b: I_b \in L_i} \log \frac{2 \cdot R_{I_b}(I_a)}{R_{I_a}} + \log R_{I_a} \quad (7.22)$$

Items in the cache can be safely ignored because their relevance ranks are small (lower than the relevance rank of the *N*th item in the buddy table). The final ranking for recommendation then becomes:

$$R_{I_a, P_i} = \sum_{\forall I_b: I_b \in L_i \cap I_a \in B_{I_b}} \log R_{I_b}(I_a) - (q - 1) \log R_{I_a} \quad (7.23)$$

```

1  /*Input:  $L_i$  download list*/
2  /*Output:  $\{I_a\}$ ; Top-N Recommendation list*/
3  recommend( $L_i$ ) {
4    for  $I_b \in L_i$  { /*For each item in the list*/
5       $\{B_{I_b}\} = \{B_{I_b}\} + B_{I_b}$ ; /*Get buddy tables*/
6    }
7    for  $I_a \in \{B_{I_b}\}$  {
8      get  $R_{I_b}(I_a)$ ,  $R_{I_a}$  from  $\{B_{I_b}\}$ ; /*Get relevance rank*/
9       $R_{I_a, P_i} = \text{eq. (7.23)}$ ; /*Calculate rec.*/
10   }
11   return topN( $\{I_a, R_{I_a, P_i}\}$ ); /*Return rec. items*/
12 }

```

Figure 7.7: Recommendation Procedure.

where q is number of the elements in the first term. $I_a \in B_{I_b}$ means I_a is in the buddy table of item I_b and its relevance rank $R_{I_b}(I_a)$ is available.

Finally, the procedure to perform a recommendation goes as follows: Firstly, for a given user, the buddy tables of all the items within the profile of the user (L_i) are downloaded. Then, for all the items in the collected buddy tables, the relevance ranks towards the user are calculated based on our user-content relevance model (applying Eq. (7.23)). As a result, the *top-N* ranked items (with their location indicated in the buddy tables) are recommended to the user. This is illustrated in Fig. 7.7.

7.4.4 Experiments

To validate our proposed self-organizing distributed collaborative filtering approach, we simulated a situation in which users exchange music files on a P2P network.

The data sets we use are collected from the *Audioscrobbler* (Last.FM) community. The audioscrobbler data set collects the play-lists of the users in the community by using a plug-in in the users' media players (for instance, Winamp, iTunes, XMMS etc). Plug-ins send the title (song name and artist name) of every song users play to the Audioscrobbler server, which updates the user's musical profile with the new song. That is, when a user plays a song in a certain time, this transaction is recorded as a form of $\{\text{userID}, \text{itemID}, t\}$ tuple in the database. This data set is continuously updated and at the time we captured it, it contained 1,862,766 transactions from 6,359 user IDs and 857,020 item IDs.

Pre-processing: The data set was strongly polluted and needed to be processed first:

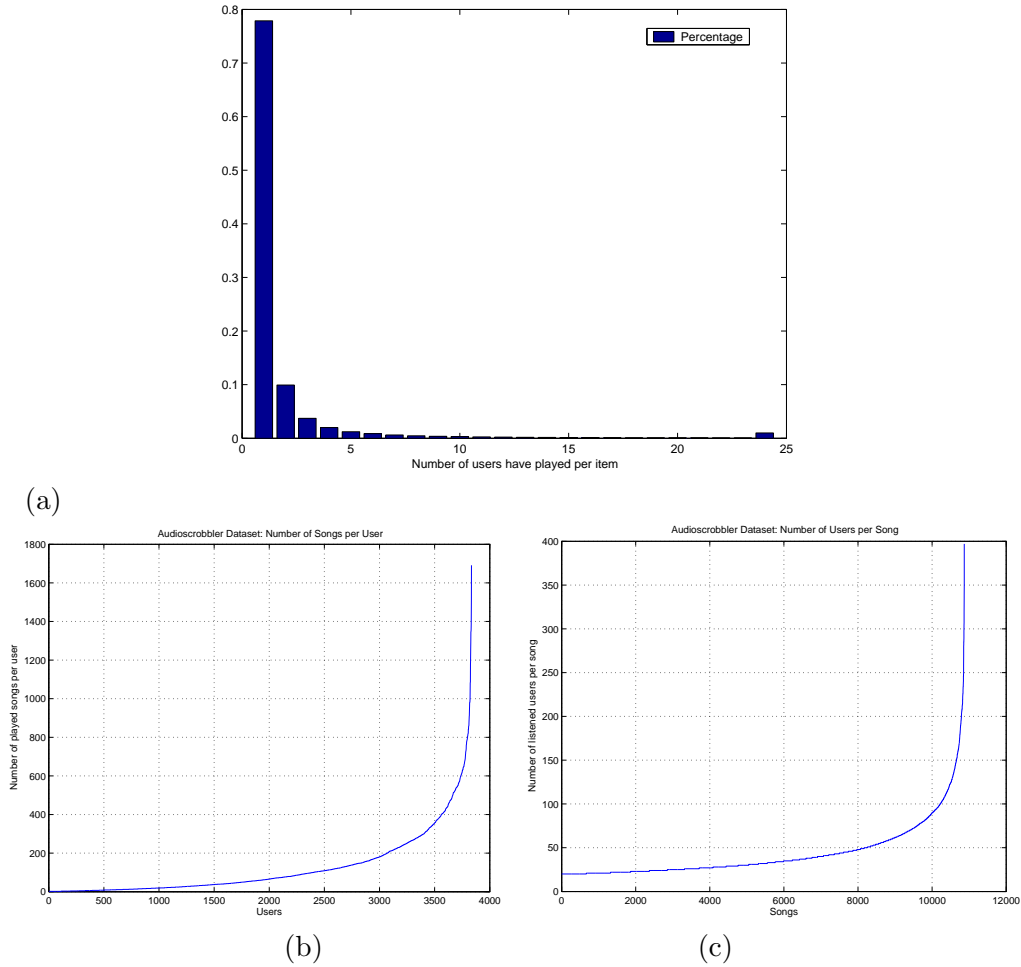


Figure 7.8: Audioscrobbler data set. (a) The distribution of the number of users per item in the data set. Each bin indicates the percentage of items that were played by the indicated number of users. The last bin represents the percentage of items that have 25 or more users. (b) The number of items per user after pre-processing. The users are ranked with respect to the number of items that they have been playing. (c) The number of users per item after pre-processing. The items are ranked according to the number of users that played it.

- 385 redundant item IDs and 2903 inactive user IDs (users that once registered but never played any song) were removed.
- Items that were played by less than 20 users were removed. This was done since: 1) many titles of the songs were incorrect (because usually the title of the song is extracted from the name of the file name), and 2) 77.9% of the songs were only played by one user (see Fig. 7.8(a)). When

| Song | Artist |
|--|----------------------------------|
| Lean On Me (duet with terry callier) | 04 |
| Vlkommen hit - 03 - Vran hemlighet | Jumper |
| Faceless(Retail):The Awakening | Godsmack |
| Closure - Aeon Flux: Eyes To Despise | Drew Neumann |
| Sweet July | 14 |
| Body Heat - from "Body Heat" | Jazz At The Movies Band |
| Mansion on the Hill | Bruce Springsteen and the E Stre |
| Napalm Brain,Scatter Brain | dj shadow |
| Anna Begins | 01-09 |
| Fight To be Free [From Survive (1988)] | Nuclear Assault |
| B Side - E-Bow The Letter | Radiohead |
| War | Henry Cow -01/0 |
| Les rues de San Francisco | GENERATION TV |

(a)

| Song | Artist |
|----------------------|------------------------|
| Universal Love | 4 Hero |
| Pluto | Les Savy Fav |
| Call Letter Blues | Bob Dylan |
| Shameless | Billy Joel |
| Spine | Machine Head |
| Conduit | Converge |
| Nightingale | Yanni |
| Mad Man Moon | Genesis |
| Sorrow | Life Without Buildings |
| No Love | Dinky |
| Under Par | Thrice |
| Hype | Tegan And Sara |
| Handbags And Gladrag | Rod Stewart |

(b)

Table 7.2: An example of the pollution of titles of songs within the Audioscrobbler data set. (a) Most of the songs which are only played by one user have either an incorrect song name or artist name. (b) Titles of songs (and the corresponding artist) played by more than five users are correct.

we randomly selected 100 songs played by only one user, 80% titles were wrong or odd (see Table 7.2(a)). The percentage of incorrect titles is extremely reduced when items played by more than 5 users are considered (see Table 7.2(b)).

- Users that had played less than 2 items were removed since their profiles do not add relevance within the network (they had only one item in their download list).

After the pre-processing, we were left with 475531 transactions from 3854 userIDs and 10869 itemIDs. The sparsity is 98.86%. Fig. 7.8(b) and (c) show the distributions of the obtained data set.

| Item | I Should Have Known... -Beatles | 2+2=5 -Radiohead | I Still Haven't Found... I'm -U2 |
|---------|---------------------------------------|--------------------------------------|---|
| Buddy1 | <u>Drive My Car -Beatles</u> | <u>Backdrifts -Radiohead</u> | <u>With or Without You -U2</u> |
| Buddy2 | <u>Love Me Do -Beatles</u> | <u>Where I End... -Radiohead</u> | 1979 -Smashing Pumpkins |
| Buddy3 | <u>Magical Mystery Tour -Beatles</u> | <u>sail to the moon -Radiohead</u> | <u>Pride (in the name of love) -U2</u> |
| Buddy4 | <u>Think For Yourself -Beatles</u> | <u>The Gloaming -Radiohead</u> | <u>Bad -U2</u> |
| Buddy5 | <u>A Hard Day's Night -Beatles</u> | <u>Myxamatoxis -Radiohead</u> | Porcelain -Moby |
| Buddy6 | Nowhere Man -Beatles | there there -Radiohead | Sunday Bloody Sunday -U2 |
| Buddy7 | I Am The Walrus -Beatles | scatterbrain -Radiohead | Where The Streets... -U2 |
| Buddy8 | Michelle -Beatles | I will -Radiohead | Sweetest Thing -U2 |
| Buddy9 | Fixing A Hole -Beatles | Go to sleep -Radiohead | Yesterday -Beatles |
| Buddy10 | Wait -Beatles | Paranoid Android -Radiohead | Honey -Moby |
| Item | Mother And Father -Madonna | Verse Chorus Verse -Nirvana | Something is Calling... -Norah Jones |
| Buddy1 | <u>American Life -Madonna</u> | <u>My Hero -Foo Fighters</u> | <u>Turn Me On -Norah Jones</u> |
| Buddy2 | <u>Intervention -Madonna</u> | <u>Coffee and TV -Blur</u> | Don't Know Why -Norah Jones |
| Buddy3 | <u>Die Another Day -Madonna</u> | <u>Loser -Beck</u> | <u>The Nearness Of You -Norah Jones</u> |
| Buddy4 | <u>Easy Ride -Madonna</u> | <u>in the End -Linkin Park</u> | <u>Shoot the Moon -Norah Jones</u> |
| Buddy5 | <u>Love Profusion -Madonna</u> | <u>No surprises -Radiohead</u> | <u>Nightingale -Norah Jones</u> |
| Buddy6 | X-Static Process -Madonna | Schism -Tool | big yellow taxi -Counting Crows |
| Buddy7 | I'm So Stupid -Madonna | Dumb -Nirvana | Dear Prudence -Beatles |
| Buddy8 | Hollywood -Madonna | On A Plain -Nirvana | Painter Song -Norah Jones |
| Buddy9 | Nobody Knows Me -Madonna | Castaway -Green Day | Seven Nation Army -White Stripes |
| Buddy10 | Sleeping With Ghosts -Placebo | Shiver -Coldplay | Love Song for No One -John Mayer |
| Item | Too Much To ... -Avril Lavigne | Most girls -Pink | Poem to a Horse -shakira |
| Buddy1 | <u>Losing Grip -Avril Lavigne</u> | <u>Forgot About Dre... -Dr. Dre</u> | <u>Whenever, Wherever -shakira</u> |
| Buddy2 | <u>Anything But... -Avril Lavigne</u> | <u>in the End -Linkin Park</u> | <u>Fool -shakira</u> |
| Buddy3 | <u>Complicated -Avril Lavigne</u> | <u>My Own Worst Enemy -Lit</u> | <u>The One -shakira</u> |
| Buddy4 | <u>Things I'll ... -Avril Lavigne</u> | <u>Come On... -Smash Mouth</u> | <u>Rules -shakira</u> |
| Buddy5 | <u>Unwanted -Avril Lavigne</u> | <u>Gin and ... -Snoop Doggy Dogg</u> | Ready For The ... -shakira |
| Buddy6 | Naked -Avril Lavigne | Superman -Five For Fighting | Underneath Your... -shakira |
| Buddy7 | Mobile -Avril Lavigne | Crawling -Linkin Park | Something In The... -Nirvana |
| Buddy8 | My World -Avril Lavigne | Psycho -System Of A Down | Clocks -Coldplay |
| Buddy9 | Tomorrow -Avril Lavigne | Last Kiss -Pearl Jam | Dumb -Nirvana |
| Buddy10 | Nobody's Fool -Avril Lavigne | Breathe -Prodigy | Pennyroyal Tea -Nirvana |

Table 7.3: The top-10 buddy tables for nine songs, each of a different artist, after 374.530 transactions. The relevance links shown in figure 7.9 (f) have been created from the first five items; these links are shown underlined here.

We randomly divided the data set into a training set (80% of the users) and a test set (20% of the users). We used the training set to calculate the buddy tables, the relevance links and to build the recommendations. The test set was used for evaluating the accuracy of the recommendations.

In the training set, there were 3067 users and 374530 transactions. Each transaction (play action in the data set) was labelled with a time index. As before, each transaction then represents a user (*userID*) that downloads an item (*itemID*) from another peer at the attached time (*t*).

In the test data set there were 767 users. The play actions of these users were used to test the accuracy of the recommendations. For each test user, 50% of the items of a test user were put into the download list of that user (the user profile). The other 50% of the items were used to test the recommendations. Thus, the number of items in the download lists of the users reflect the distribution in the overall data set.

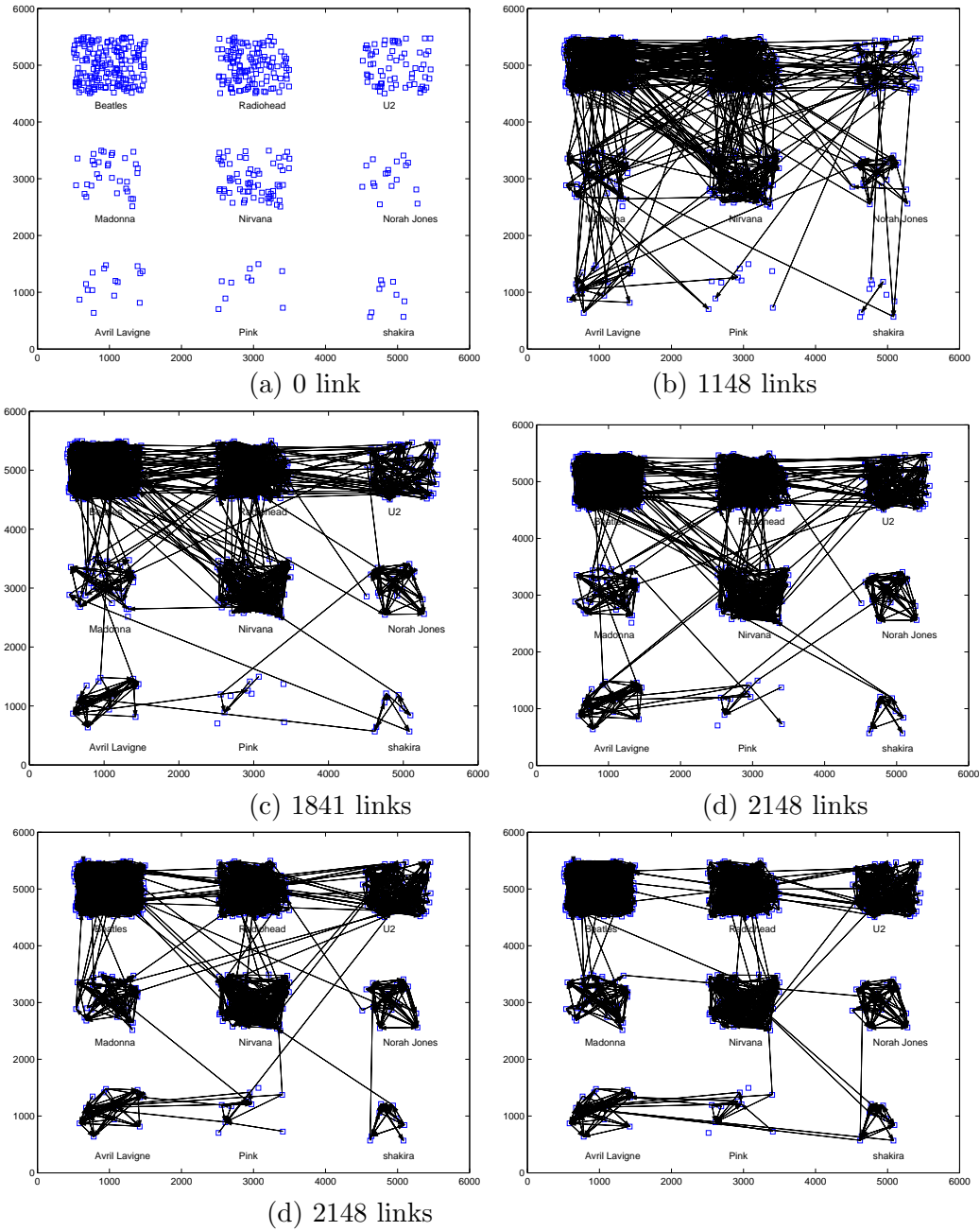
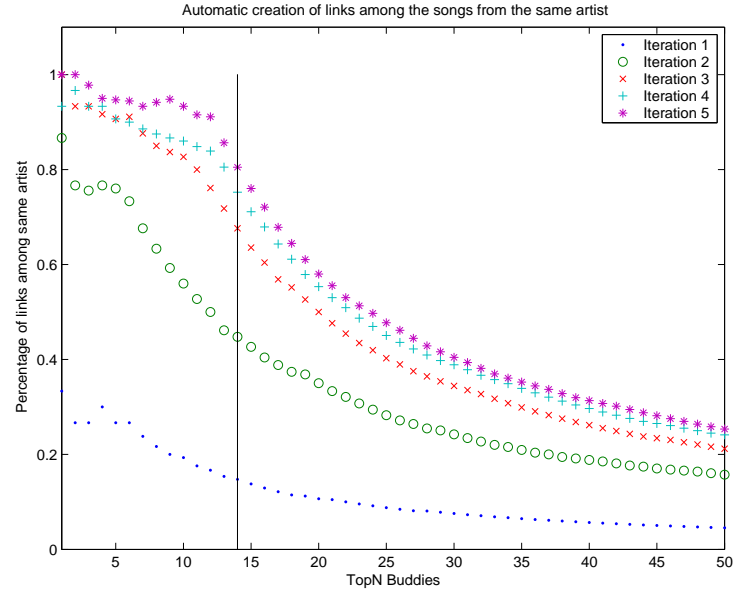
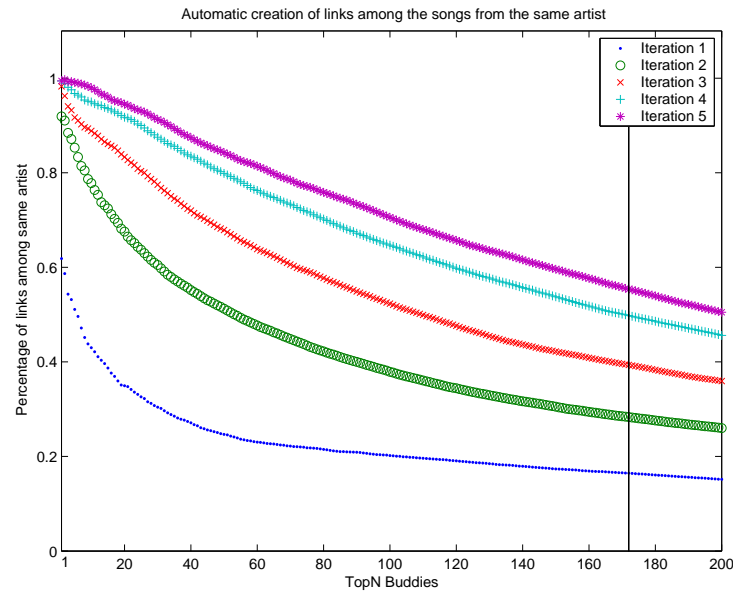


Figure 7.9: Illustration of dynamically created relevance links between the songs of nine artists. Each song is represented by a rectangle. Songs from the same artist are clustered within a grid, resulting in the nine rectangle regions. For clarity, the links to songs of other artists (than the nine showed) have been removed. The panels show the relevance links after (a) 0, (b) 74.906 (iteration 1), (c) 149.812 (iteration 2), (d) 224.718 (iteration 3), (e) 299.624 (iteration 4), and, (f) 374.530 (iteration 5) transactions.



(a)



(b)

Figure 7.10: Percentage of relevance links towards songs of the same artist within the *top-N* ranked items in a buddy table for different settings of *N* and after a different number of transactions (iterations). (a) The average percentage of links between the 15 songs from *Avril Lavigne* within their respective *top-N* buddy tables. (b) Similarly for 173 items from the Beatles.

7.4.5 Self-organizing Relevance Links

To simulate media files sharing in the P2P network, we uniformly distributed each item across the different peers. Each peer runs a *demon* program described in Fig. 7.6 to update the buddy tables during each transaction: Each time a transaction takes place (i.e. when a multimedia file is downloaded), the relevance links in the buddy table of the item that is being downloaded are updated. The dynamic behavior of the relevance between items can thus be studied. Figure 7.9(a) to (f) illustrate the links that were created after: (a) 0, (b) 74.906, (c) 149.812, (d) 224.718, (e) 299.624, and, (f) 374.530 transactions.

The figure shows the songs (items) of nine artists selected such that they reflect different genre of music and different amounts of songs within the database. Songs from the same artist are grouped together. This results in nine clusters shown in the figure. For each item (song), links to the *top-5* relevant items (according to their buddy table) are displayed as directed arrows (pointing outwards the buddy table item). For reasons of clarity, only the links between the displayed items are shown.

From the figure, we observe the following:

- The number of relevance links increases with an increase in the number of transactions.
- The relevance links converge and cluster songs (items) of the same artists. This can be seen from the large number of links between songs of the same artist that arises with an increase in the number of transactions. This can also be seen from Table 7.3, which shows the buddy tables of nine songs, each of a different artist (only top-10 ranked items are displayed). To measure this further, we plotted the percentage of links among songs from the same artist as a function of the number of relevant links considered (*top-N*) and the number of transactions. Figure 7.10 shows these dependencies for two artists: *Avril Lavigne* and the *Beatles*. This figure confirms the observation that with an increase of number of transactions, the percentage of the links between songs of the same artist increases. It also shows that songs from the same artists are more relevant (have a better ranking position) than songs of other artists. This can be noted from the increase in the percentage of songs by the same artists in the *top-N* ranked items from the buddy tables of the songs by that artists when N is decreased.
- Figure 7.9 also shows that, besides relevance links between songs by the same artist, relevance links have been created between songs of the same genre (reflecting the interest of a group of users). For instance, relevance links have been created between *U2*, *Radiohead* and *Nirvana*, groups

that belong to the rock genre. Links between *Avril Lavigne*, *Pink* and *Shakira* may indicate a group of young female pop music artists. *Norah Jones* is somehow isolated since she belongs to the jazz genre, a different style compared to those of the other eight artists.

7.4.6 Recommendation Performance

We treat each user in the test set as a target user in the system. For each target user, a recommendation was calculated by applying the algorithm described in Fig. 7.7, based on the calculated buddy tables from the training users. The resulting recommended items were then compared to the ground truth items.

The performance was measured using the *coverage* (or *recall*) and *precision*. The coverage measures the proportion of ground truth items (known by the download lists) that are recommended. The precision measures the proportion of the recommended items that are ground truth items. Note that the items in the download list of the test user represent only a fraction of the items that the user *truly* liked. Therefore, the resulting precision is smaller than the true precision.

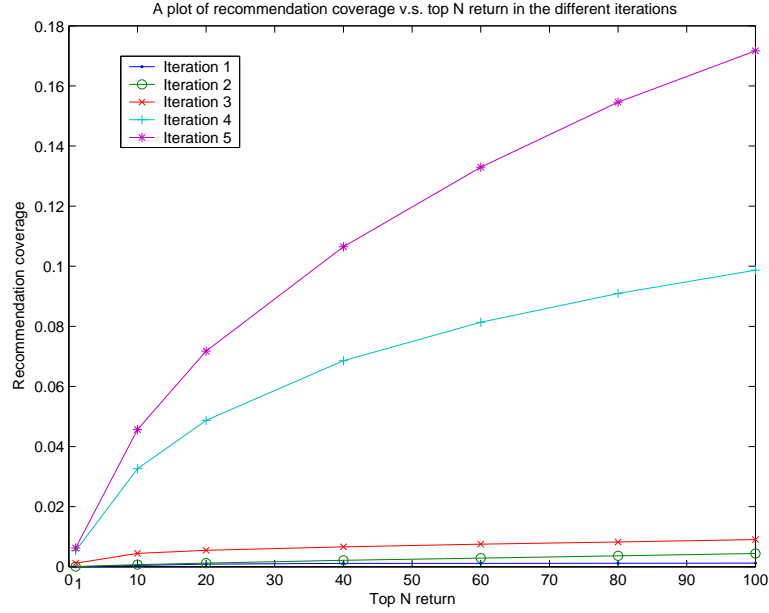
First, we investigated the impact of user interaction (transaction) on the recommendation performance. The coverage and precision of the recommendation in the five iterations are shown in Fig. 7.11 (a) and (b). The results indicate that as the number of transactions increases, the recommendation results become better.

Next, we compared our distributed collaborative filtering approach with the *Top-N suggest* recommendation engine, a well-known centralized collaborative filtering approach ([52])⁴. Both the item-based version and the a user-based version were compared. The parameters had been set according to the user manual. Additionally, we compared the three recommenders to a non-personalized recommendation approach. For each item, its prior relevance $P(r|I_T)$ was used. The items were then ranked and recommended accordingly.

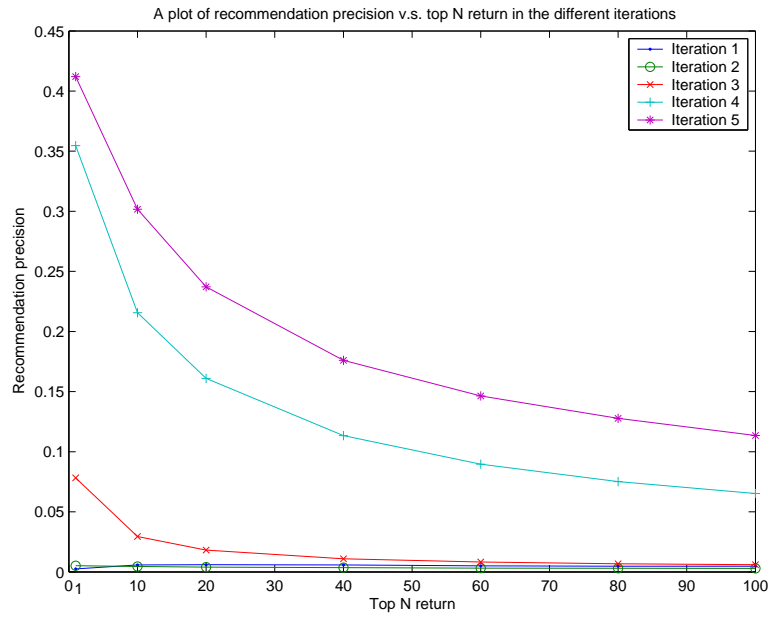
For our distributed approach, we show the results of two different settings: one with the prior relevance (second term of Eq. (7.11)) and one without it.

For computational reasons, we randomly sampled the pre-processed data set to limit the number of users to 1300 and the number of items to 4807. We then randomly divided the sampled data set: 80% (1040) users were included in the training set and the remaining 20% (260) users were in the test set. In the training set there were 159036 transactions. For our approach these were transactions labelled with a time index and they are used to build up the

⁴<http://www-users.cs.umn.edu/~karypis/suggest/>



(a)



(b)

Figure 7.11: Recommendation Convergence. (a) Coverage as a function of the N (*top-N*) most relevant items after training using the five different iterations. (b) Similarly for the precision.

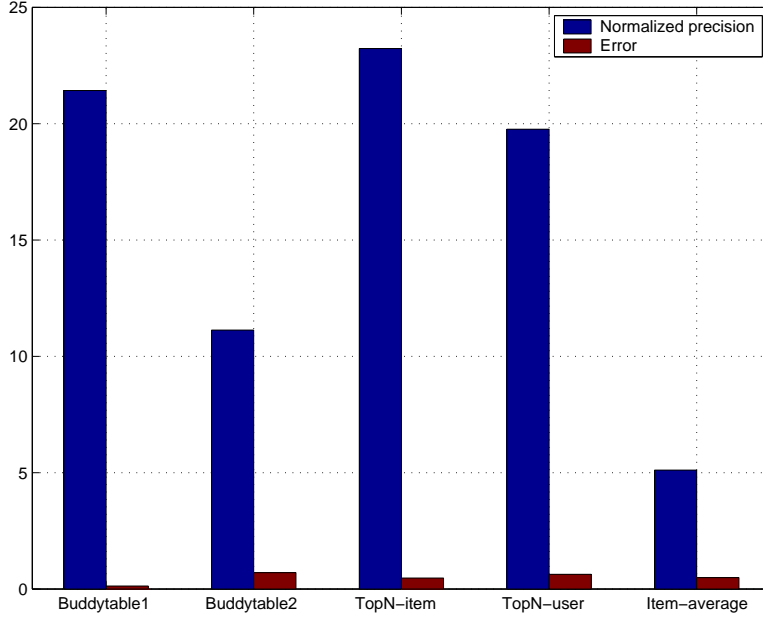


Figure 7.12: Recommendation results. The normalized precision for: (1) **Buddytable1** our proposed distributed collaborative filtering approach with the prior term of equation 7.11; (2) **Buddytable2** without the prior term; (3) **TopN-item** the centralized item-based *top-N* suggest method; (4) **TopN-user** the centralized user-based *top-N* suggest method; and (5) **Item-average** a reference recommendation method based on a average ranking.

relevance links stored in the distributed item buddy tables. For the centralized approaches these transactions were used to build a user-item rating matrix. For the test users (from the test set), again 50% of the items were put into their download list and the other 50% items acted as the ground truth.

Since the precision and coverage vary according to the number of recommended items, we adopted a normalized precision to compare the methods. To this end we normalized the precision according to the precision of a random recommendation of the same number of elements. The normalized precision denoted as p_{norm} then becomes:

$$p_{norm} = \left(\sum_{i=1}^{N_R} p(i) \right) / \left(\sum_{i=1}^{N_R} p_{rand}(i) \right) \quad (7.24)$$

where i denotes number of items returned from the recommendation. $p(i)$ denotes the precision when i items are returned from the recommendation while $p_{rand}(i)$ denotes the precision from a random recommendation. We set $N_R = 50$ in our experiment.

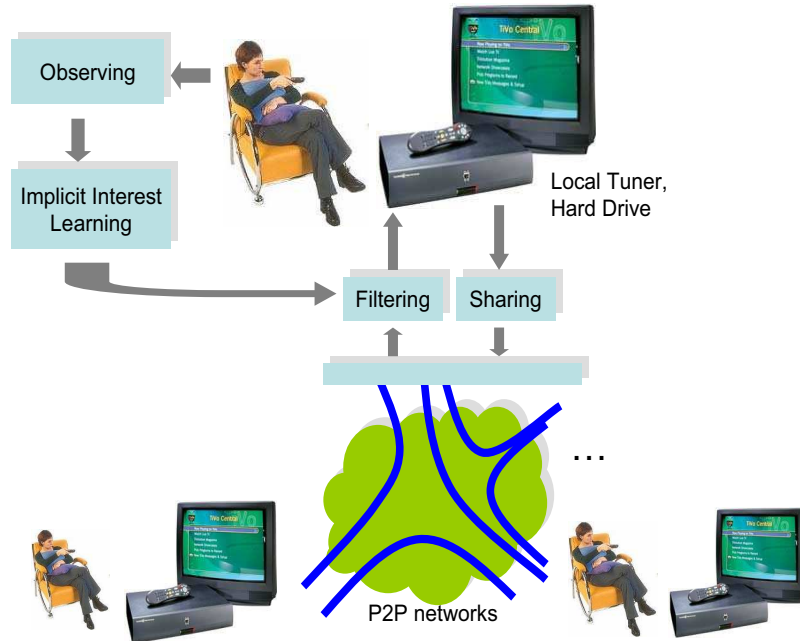


Figure 7.13: An Illustration of Tribler, a Personalized P2P File Sharing and Television System.

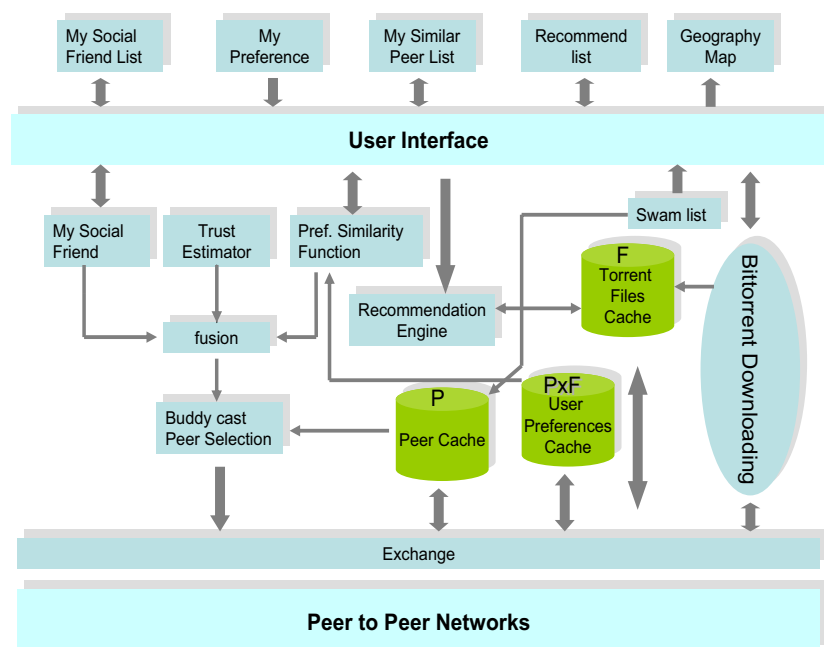
The normalized precision of the five methods is shown in Fig. 7.12. It shows that the performance of our distributed recommendation is comparable to that of the centralized methods. Our approach using prior relevance outperforms even the centralized *top-N* user-based recommendation method and approximates the best *top-N* item-based method.

7.5 Applications

7.5.1 The Tribler System⁵

Television signals have been broadcast around the world for many decades. More flexibility was introduced with the arrival of the VCR. PVR (personal video recorder) devices such as the TiVo further enhanced the television experience. A PVR enables people to watch television programs they like without

⁵This section is based on the publication: “Personalization on a peer-to-peer television system”, J. Wang, J. Pouwelse, J. Fokker, A. P. de Vries, and M. J. Reinders, Special Issue on Multimedia Tools and Applications, 2006.

Figure 7.14: The system architecture of *Tribler*.

the restrictions of broadcast schedules. However, a PVR has limited recording capacity and can only record programs that are available on the local cable system or satellite receiver.

We presents a prototype system that goes beyond the existing VCR, PVR, and VoD (Video on Demand) solutions. We believe that amongst others broadband, P2P, and recommendation technology will drastically change the television broadcasting as it exists today. Our operational prototype system called *Tribler* (Tribler.org) gives people access to all television stations in the world. By exploiting P2P technology, we have created a distribution system for live television as well as sharing of programs recorded days or months ago.

The Tribler system is illustrated in Fig 7.13. The basic idea is that each user will have a small low-cost set-top box attached to his or her TV to record the local programs from the local tuner. This content is stored on a hard disk and shared with other users (friends) through the Tribler P2P software. Each user is then both a program consumer as well as a program provider. Tribler implicitly learns the interests of users in TV programs by analyzing their zapping behavior. The system automatically recommends, records, or even downloads programs based on the learned user interest. Connecting millions

of set-top boxes in a P2P network will unbolt a wealth of programs, television channels and their archives to people. We believe this will tremendously change the way people watch TV.

The basic architecture of the Tribler system is shown in Fig 7.14 and a detailed description can be found in [79]. The key idea behind the Tribler system is that it exploits the prime social phenomenon "kinship fosters cooperation" [79]. In other words, similar taste for content can form a foundation for an online community with altruistic behavior. This is partly realized by building social groups of users that have similar taste captured in user interest profiles.

The user interest profiles within the social groups can also facilitate the prioritization of content for a user by exploiting recommendation technology. With this information, the available content in the peer-to-peer community can be explored using novel personalized tag-based navigation.

Our research focuses on the personalization aspects of the Tribler system. We use the zapping behavior of a user to learn the user interest in the watched TV programs. The zapping behavior of all users is recorded and coupled with the EPG (Electronic Program Guide) data to generate program IDs. In the Tribler system different TV programs have different IDs. TV series that consists of a set of episodes, like "Friends" or a general "news" program, get one ID (all episodes get the same ID) to bring more relevance among programs.

7.5.2 Wi-Fi Walkman⁶

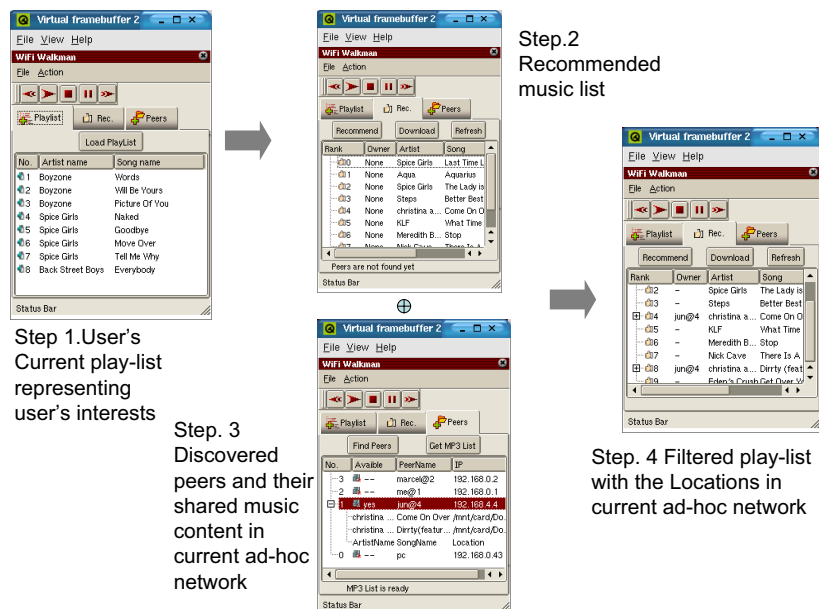
The Wi-Fi walkman that we developed is a case study that investigates the technological and usability aspects of human-computer interaction with personalized, intelligent and context-aware wearable devices in ad-hoc wireless environments such as the future home, office, or university campuses. It is a small handheld device with a wireless link that contains music content in the environment or from the user. Users carry their own Wi-Fi walkman around and listen to the music content. All this music content can be shared using mobile ad-hoc networking. The Wi-Fi walkman is situated in a peer-to-peer environment and naturally interacts with the users. Without annoying interactions with users, it can learn the users' music taste and consequently provide personalized music resources to fit the user's interest according to the user's current situated context.

The Wi-Fi walkman is implemented on the Sharp Zaurus PDA (see Fig 7.15

⁶This section is based on the publication: "Wi-Fi walkman: a wireless handheld that shares and recommends music on peer-to-peer networks", J. Wang, M. J. Reinders, J. Pouwelse, and R. L. Lagendijk, in Proc. of Embedded Processors for Multimedia and Communications II, part of the SPIE Symposium on Electronic Imaging 2005.



(a) The Implementation on a Sharp Zaurus PDA



(b) The Snap-shots of the music recommendation

Figure 7.15: The *Wi-Fi Walkman* prototype.

(a)), using C++. It is running on an ad-hoc wireless network, featuring audio playback, audio storage, audio recommendation, and ad-hoc wireless connectivity for audio exchange.

The Wi-Fi walkman itself contains an audio agent, a transport agent, and a wireless interface shown. The audio agent is responsible for the communication with the recommendation services, manages the MP3 files on the storage devices

(e.g. a flash memory card), and selects which MP3 to play. The transport agent uses the wireless ad-hoc network to communicate with other transport agents and enables the sharing of the music files. Due to the dynamic nature of an ad-hoc network, the transport agents must keep track of the other walkmans around them. The enhanced ad-hoc wireless interface also informs the transport agent of new walkmans and walkmans that can no longer be reached.

To test the performance of our P2P music recommendation, we utilize a data set of the AudioScrobbler community (**Last.FM**) as our play-list data set. Currently this data set has 857.020 tracks and 4.175.146 playback actions. The recommendation procedure is illustrated in Fig 7.15 (b).

7.6 Conclusions

This chapter studied one of the practical issues of recommender system, the decentralization problem. We proposed two overlay networks to facilitate P2P recommendation. For the user-oriented overlay network, we took user proximity into account, deriving a novel user profile exchange algorithm. For the item-oriented overlay work, we introduced the item buddy tables, which are attached to items that are distributed throughout a P2P network. As a result, the computation on the item similarity that is needed for the item-based ranking model is decentralized. Our experiments with TV and music play-list data showed that our approaches are promising techniques for recommendation in a P2P network.

Our item ranking model of collaborative filtering has the same root as other more general text retrieval models, for instance the language modeling of information retrieval, the classic probabilistic relevance models (the binary independent model and the BM25 variants), etc (see Chapter 3). Hence, it is worthwhile seeking the possible usage of the proposed overlay networks to decentralize these text retrieval models [63, 107].

Chapter 8

Discussions

This thesis has introduced several (relevance) models for collaborative filtering. Theoretically, we aimed at formulating the correspondence between user interests and information items; the probabilistic-modelling assumptions behind our proposed relevance models gave us an in-depth understanding of the underlying assumptions and limitations of existing collaborative filtering approaches. Practically, we have classified collaborative filtering into two types of practical problems, namely rating prediction and item ranking, tackling them independently. Some major practical issues such as data sparsity and distributed recommendation were also studied. To conclude the thesis with this final chapter, we will discuss the main points made in the thesis and point out future directions.

8.1 Scenarios

In practice, recommender systems and collaborative filtering exist in various forms with their input data potentially varying depending on the targeted applications. This thesis has studied two main scenarios, distinguishing between item ranking and rating prediction problems. We have seen in the study that such distinction is necessary, because it allows us to specifically target our algorithms and evaluations to the particular recommendation problem at hand. Chapters 2, 3, and 4 studied the item ranking problem, aiming at generating the top-N of the target user's most favorite items, while Chapters 5 and 6 looked at rating prediction, aiming to predict a rating of a given item for the target user. The item ranking problem is relatively close to the text retrieval problem, as both problems aim at generating a top-N relevance ranking list.

We have shown that such a collaborative filtering problem can be cast as a retrieval problem if the query is based on the user profile. Once we properly establish the link between text retrieval and collaborative filtering, the models in text retrieval, such as the language modelling of information retrieval and the classic probabilistic models (the binary independence relevance model and its BM25 variants), can be naturally reformulated and extended to the item ranking problem in collaborative filtering.

8.2 Relevance

As has been seen in the thesis, using the concept of relevance is a natural way to explain the correspondence between user interests and information items. Introducing relevance into collaborative filtering has resulted in several formal frameworks for modelling collaborative filtering. In these frameworks, several relevance models are independently derived, targeting various data types and application scenarios. In contrast to previous studies, these probabilistic relevance models have not only solved some specific problems of collaborative filtering but, most importantly, have also provided an insight into collaborative filtering problems. Our relevance models in Chapters 2 and 4 depicted a statistical ranking mechanism that lies at the heart of item ranking for collaborative filtering. For instance, our proposed ranking models imply that the relevance ranking of a target item should consider two aspects, namely the *personalized* and *generalized relevance*. For example, in the item-generation models, the generalized relevance arises from the popularity of the item while the personalized relevance equals the co-occurrence of such an item with other items that are represented in the profile of the target user.

8.2.1 Two views

Previous studies on collaborative filtering make a distinction between user-based and item-based approaches. Our probabilistic relevance models, for both item ranking (Chapter 1) and rating prediction (Chapter 6), were derived with an information retrieval view to collaborative filtering. They demonstrated that the user-based and item-based models are equivalent from a probabilistic point of view, since they have actually been derived from the same generative relevance model. The only difference corresponds to the choice of independence assumptions in the derivations, leading to the two distinct factorizations.

Most importantly, the combination of the two models (partial views) is of particular interest, because such a merger could compensate for the independent assumptions embedded in either of the models and further alleviate the data

sparsity problem. Chapter 6 has presented a unified relevance model, providing the complete and unified view of the problem. In this model, we do not fix the two variables, user and item. Instead, we construct a unified model that relies on both the user representation and the item representation. The proposed combination of Parzen-window kernel density estimation with the relevance models provides a general framework for collaborative filtering, showing how the final prediction is expressed by summations over rating influences from user neighbors, item neighbors, and both user and item neighbors. In addition, Chapter 5 showed that the cosine distance is indeed equivalent to a Euclidean distance measure, but in the projected space. As a result, the classic vector space model was covered in our proposed probabilistic framework.

8.3 Data Sparsity

Data sparsity seriously hampers the performance and practical usage of collaborative filtering. The study in this thesis revealed that it can be solved in two ways. 1) Using the statistics of whole collection. As in text retrieval, unreliable predictions due to inadequate observations can be smoothed by a properly introduced background model and this background model can be estimated from the statistics of the whole collection (Chapters 1, 2 and 3). 2) Fusing two representations. As we have demonstrated, user-based approaches or item-based approaches of collaborative filtering use only partial information. The final prediction can be smoothed by fusing both the user representation and the item representation (Chapters 5 and 6).

8.4 Future Research

8.4.1 Discovering More from Information Retrieval Models

This thesis has setup a close relationship between the probabilistic models of text retrieval and those of collaborative filtering, facilitating a flexible approach to integrating other techniques from text retrieval (e.g. query expansion, or relevance feedback) whenever necessary.

In one of our formulations, a user interest is represented by a set of items that the user has either rated or visited (played, listened etc). We treat it as a query to rank unseen items. One of the limitations for such a representation is that recommender systems need a target user to rate or visit a certain number of items to be able to generate a good recommendation for that user. In practice, however, particularly in the initial stage, we have few observed items available,

which usually results in an unsatisfactory recommendation. It becomes even worse in a situation where the query items are less popular. The difficulty, often called the “cold-start” problem [94], can be reduced if we extend our models by integrating query expansion from information retrieval [122]. That is, we consider a user profile for a set of items as our initial guess of the user interest (query), rather than directly representing it as we did in our current models. The cold start problem can therefore be alleviated by expanding the initial guess using extra items that are similar or have some other statistical relation to the set of specified items in the query.

It was shown in the introduction of this thesis (Fig. 1.1) that generating the top-N recommendation list or predicting the rating for a certain item does not necessarily mean the end of the task. User feedback on the initial rankings or prediction outputs is a valuable resource with which to further validate and refine the recommendation results. Using the mechanism of relevance feedback has been proven to be a powerful feature in information retrieval systems [16, 88, 89]. In this regard, we are particularly interested in introducing a relevance feedback loop into the relevance models in the future.

8.4.2 Beyond Collaborative Filtering

The proposed models in Chapters 2 and 4 are generative models for co-occurrence data. An interesting idea would be to explore further the usage on related applications beyond collaborative filtering. For example, our ranking models can be easily extended and applied to the expert search task, which aims at ranking people (experts) with respect to a user’s specific expertise request on a topic [3]. It would also be of great interest to apply the unified relevance model for the unification of document and query generation in text retrieval [7, 82, 85].

Measuring relevance between information items (documents) and user needs is a fundamental problem of information retrieval. In the classic probabilistic text retrieval models, relevance is implicitly modelled to be dependent on the query and the document only. However, an issued query is not the only way to represent a particular user information need. This can be further clarified if we take user profiles into account. Thus, the relevance of a document is dependent on both the user’s specified query as well as on his or her user profile. Thanks to the models developed in this thesis, current developments in the two distinct domains (i.e. text retrieval and collaborative filtering) can be integrated to achieve user-centric information retrieval.

Bibliography

- [1] G. Adomavicius and A. Tuzhilin. 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, 2005.
- [2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, May 1999.
- [3] K. Balog, L. Azzopardi, and M. de Rijke. Formal models for expert finding in enterprise corpora. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 43–50, New York, NY, USA, 2006. ACM Press.
- [4] G. Begelman, P. Keller, and F. Smadja. Automated tag clustering: Improving search and exploration in the tag space. In *WWW2006: Proceedings of the Collaborative Web Tagging Workshop*, Edinburgh, Scotland, 2006.
- [5] N. J. Belkin and W. B. Croft. Information filtering and information retrieval: two sides of the same coin? *Commun. ACM*, 35(12):29–38, 1992.
- [6] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [7] D. Bodoff. A re-unification of two competing models for document retrieval. *J. Am. Soc. Inf. Sci.*, 50(1):49–64, 1999.
- [8] D. Bodoff and S. Robertson. A new unified probabilistic model. *J. Am. Soc. Inf. Sci. Technol.*, 55(6):471–487, 2004.

- [9] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 43–52, San Francisco, CA, 1998. Morgan Kaufmann.
- [10] J. Canny. Collaborative filtering with privacy via factor analysis. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 238–245, New York, NY, 2002. ACM Press.
- [11] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336, New York, NY, USA, 1998. ACM.
- [12] H. Chen and D. R. Karger. Less is more: probabilistic models for retrieving fewer relevant documents. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 429–436, New York, NY, USA, 2006. ACM.
- [13] K.-W. Cheung and L. F. Tian. Learning user similarity and rating style for collaborative recommendation. *Inf. Retr.*, 7(3-4):395–410, 2004.
- [14] P. A. Chirita, C. S. Firan, and W. Nejdl. Personalized query expansion for the web. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 7–14, New York, NY, USA, 2007. ACM Press.
- [15] M. Claypool, P. Le, M. Wased, and D. Brown. Implicit interest indicators. In *IUI '01: Proceedings of the 6th international conference on Intelligent user interfaces*, pages 33–40, New York, NY, USA, 2001. ACM.
- [16] I. J. Cox, M. L. Miller, S. M. Omohundro, and P. N. Yianilos. Pichunter: Bayesian relevance feedback for image retrieval. In *ICPR '96: Proceedings of the International Conference on Pattern Recognition (ICPR '96) Volume III-Volume 7276*, page 361, Washington, DC, USA, 1996. IEEE Computer Society.
- [17] A. Crespo and H. Garcia-Molina. Semantic overlay networks for p2p systems. Technical report, Comp. Sci. Dept., Stanford University, 2003.
- [18] B. W. Croft and J. Lafferty. *Language Modeling for Information Retrieval*. Springer, 2003.
- [19] DataSet. MovieLens: <http://www.grouplens.org/>.

- [20] DataSet. EachMovie: <http://research.compaq.com/SRC/eachmovie/>.
- [21] DataSet. MovieRating: http://www.cs.usyd.edu.au/~irena/movie_data.zip.
- [22] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.
- [23] M. Deshpande and G. Karypis. Item-based top-N recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.
- [24] Z. Dou, R. Song, and J.-R. Wen. A large-scale evaluation and analysis of personalized search strategies. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 581–590, New York, NY, USA, 2007. ACM Press.
- [25] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley Interscience, Wiley, New York, 2001.
- [26] R. P. W. Duin. On the choice of smoothing parameters for parzen estimators of probability density functions. *IEEE Transactions on Computers*, C-25:1175– 1179, 1976.
- [27] S. Eyheramendy, D. Lewis, and D. Madigan. On the naive bayes model for text categorization. In *Proc. of Artificial Intelligence and Statistics*, 2003.
- [28] J. Fokker, J. Pouwelse, and W. Buntine. Tag-based navigation for peer-to-peer wikipedia. In *WWW2006: Proceedings of the Collaborative Web Tagging Workshop*, Edinburgh, Scotland, 2006.
- [29] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman and Hall, 2003.
- [30] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, 1992.
- [31] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval Journal*, 4(2):133–151, July 2001.
- [32] S. A. Golder and B. A. Huberman. The structure of collaborative tagging systems. Technical report, Information Dynamics Lab, HP Labs, 2005. <http://www.hpl.hp.com/research/idl/papers/tags/tags.pdf>.

- [33] H. Halpin, V. Robu, and H. Shepherd. The complex dynamics of collaborative tagging. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 211–220, New York, NY, USA, 2007. ACM Press.
- [34] A. Hanjalic. *Content-Based Analysis of Digital Video*. Kluwer Academic Publishers, 2004.
- [35] S. Harter. A probabilistic approach to automatic keyword indexing. *Journal of the American Society for Information Science*, 35:197–206 and 280–289, 1975.
- [36] J. L. Herlocker. *Understanding and improving automated collaborative filtering systems*. PhD thesis, University of Minnesota, 2000. Adviser-Joseph A. Konstan.
- [37] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, New York, NY, 1999. ACM Press.
- [38] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.
- [39] D. Hiemstra. *Using language models for information retrieval*. Doctoral thesis, University of Twente, 2001.
- [40] D. Hiemstra. Term-specific smoothing for the language modeling approach to information retrieval: the importance of a query term. In *Proc. of SIGIR*, pages 35–41, 2002.
- [41] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Info. Syst.*, Vol 22(1):89–115, 2004.
- [42] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *Proc. of IJCAI*, 1999.
- [43] R. Hu and Y. Lu. A hybrid user and item-based collaborative filtering with smoothing on sparse data. In *ICAT Workshops*, pages 184–189, 2006.
- [44] Z. Huang, H. Chen, and D. Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):116–142, 2004.

- [45] D. Hull. Using statistical testing in the evaluation of retrieval experiments. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 329–338, New York, NY, 1993. ACM Press.
- [46] T. Jebara. *Machine Learning: Discriminative and Generative (Kluwer International Series in Engineering and Computer Science)*. Kluwer Academic Publishers, Norwell, MA, USA, 2003.
- [47] M. Jelasity and M. van Steen. Large-scale newscast computing on the Internet, Oct. 2002.
- [48] R. Jin, J. Y. Chai, and L. Si. An automatic weighting scheme for collaborative filtering. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 337–344, New York, NY, 2004. ACM Press.
- [49] R. Jin and L. Si. A bayesian approach toward active learning for collaborative filtering. In *UAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 278–285, Arlington, Virginia, United States, 2004.
- [50] R. Jin, L. Si, and C. Zhai. A study of mixture models for collaborative filtering. *Inf. Retr.*, 9(3):357–382, 2006.
- [51] M. Jordan. *Learning in Graphical Models*. MIT Press, 1999.
- [52] G. Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proc. of the tenth international conference on Information and knowledge management*, 2001.
- [53] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(3):226–239, 1998.
- [54] F.-F. Kuo and M.-K. Shan. A personalized music filtering system based on melody style classification. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, page 649, Washington, DC, USA, 2002. IEEE Computer Society.
- [55] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 111–119, New York, NY, 2001. ACM Press.
- [56] J. Lafferty and C. Zhai. Probabilistic relevance models based on document and query generation. *Language Modeling and Information Retrieval, Kluwer International Series on Information Retrieval*, V.13:1–10, 2003.

- [57] V. Lavrenko. *A generative theory of relevance*. PhD thesis, University of Massachusetts Amherst, 2004. Director-W. Bruce Croft and Director-James Allan.
- [58] V. Lavrenko and W. B. Croft. Relevance based language models. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 120–127, New York, NY, 2001. ACM Press.
- [59] D. D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *ECML '98: Proceedings of the 10th European Conference on Machine Learning*, pages 4–15, London, UK, 1998. Springer-Verlag.
- [60] R. Library. SUGGESTLib: <http://www-users.cs.umn.edu/~karypis/suggest/>.
- [61] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, Jan/Feb.:76–80, 2003.
- [62] Q. Liu, H. Lu, and S. Ma. Improving kernel fisher discriminant analysis for face recognition. *IEEE Trans. Circuits Syst. Video Techn.*, 14(1):42–49, 2004.
- [63] J. Lu and J. Callan. Content-based retrieval in hybrid peer-to-peer networks. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 199–206, New York, NY, USA, 2003. ACM Press.
- [64] E. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Survey*, 2004.
- [65] D. Maltz and K. Ehrlich. Pointing the way: active collaborative filtering. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 202–209, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.
- [66] B. Marlin. Collaborative filtering: a machine learning perspective. Master's thesis, Department of Computer Science, University of Toronto, 2004.
- [67] C. Marlow, M. Naaman, D. Boyd, and M. Davis. Position paper, tagging, taxonomy, flickr, article, toread. In *WWW2006: Proceedings of the Collaborative Web Tagging Workshop*, Edinburgh, Scotland, 2006.

- [68] M. E. Maron and J. L. Kuhns. On relevance, probabilistic indexing and information retrieval. *J. ACM*, 7(3):216–244, 1960.
- [69] M. R. McLaughlin and J. L. Herlocker. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 329–336, New York, NY, USA, 2004. ACM Press.
- [70] D. R. Millen and J. Feinberg. Using social tagging to improve social navigation. In *Workshop on the Social Navigation and Community based Adaptation Technologies*, Dublin, Ireland, 2006.
- [71] B. N. Miller, J. A. Konstan, and J. Riedl. Pocketlens: Toward a personal recommender system. *ACM Trans. Inf. Syst.*, 22(3):437–476, 2004.
- [72] S. Mizzaro. Relevance: The whole history. *Journal of the American Society of Information Science*, 48(9):810–832, 1997.
- [73] B. Mobasher. Data mining for personalization. In *the Adaptive Web: Methods and Strategies of Web Personalization*, pages 90–135, Berlin-Heidelberg, 2007. Springer.
- [74] T. Oka, H. Morikawa, and T. Aoayama. Vineyard : A collaborative filtering service platform in distributed environment. In *Proc. of the IEEE/IPSJ Symposium on Applications and the Internet Workshops*, 2004.
- [75] P. Paclik, J. Novovicova, P. Pudil, and P. Somol. Road sign classification using laplace kernel classifier. *Pattern Recogn. Lett.*, 21(13-14):1165–1173, 2000.
- [76] H. Peng, X. Bo, Y. Fan, and S. Ruimin. A scalable P2P recommender system based on distributed collaborative filtering. *Expert systems with applications*, 2004.
- [77] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach. In *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 473–480, San Francisco, CA, 2000. Morgan Kaufmann Publishers Inc.
- [78] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, New York, NY, 1998. ACM Press.

- [79] J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. Epema, M. Reinders, M. van Steen, and H. Sips. Tribler: A social-based based peer to peer system. In *5th Int'l Workshop on Peer-to-Peer Systems (IPTPS)*, Feb 2006.
- [80] J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 713–719, New York, NY, 2005. ACM Press.
- [81] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, 1994. ACM Press.
- [82] S. Robertson. The unified model revisited. In *Mathematical/Formal Methods in IR, Workshop in SIGIR 2003*, New York, NY, 2003. ACM Press.
- [83] S. E. Robertson. The probability ranking principle in IR. *Readings in information retrieval*, pages 281–286, 1997.
- [84] S. E. Robertson. On event spaces and probabilistic models in information retrieval. *Information Retrieval*, 8(2):319 – 329, 2005.
- [85] S. E. Robertson, M. E. Maron, and W. Cooper. Probability of relevance: a unification of two competing models for document retrieval. *Information Technology: Research and Development*, 1(1):1–21, 1982.
- [86] S. E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–46, 1976.
- [87] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 232–241, New York, NY, 1994. Springer-Verlag New York, Inc.
- [88] Y. Rui, T. S. Huang, and S. Mehrotra. Content-based image retrieval with relevance feedback in mars. In *ICIP (2)*, pages 815–818, 1997.
- [89] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Readings in information retrieval*, pages 355–364, 1997.
- [90] G. Salton and M. J. McGill. *Introduction to modern information retrieval*. New York : McGraw-Hill, 1983.

- [91] T. Saracevic. Relevance: A review of and a framework for the thinking on the notion in information science. *Journal of the American Society of Information Science*, 26(6):321–43, 75.
- [92] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, 2001. ACM Press.
- [93] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender system – a case study. In *Proc. of ACM WebKDD Workshop*, New York, NY, 2000. ACM Press.
- [94] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proc. of SIGIR*, 2002.
- [95] B. Schölkopf. The kernel trick for distances. In *NIPS*, pages 301–307, Cambridge, MA, 2000. MIT Press.
- [96] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.
- [97] U. Shardanand and P. Maes. Social information filtering: algorithms for automating "word of mouth". In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217, New York, NY, 1995. ACM Press/Addison-Wesley Publishing Co.
- [98] X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 43–50, New York, NY, USA, 2005. ACM Press.
- [99] L. Si and R. Jin. Flexible mixture model for collaborative filtering. In *ICML'03: Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, pages 704–711, DC, 2003. AAAI Press.
- [100] M. Skurichina. Effect of the kernel function form on the quality of nonparametric parzen window classifier. *Statistical Problems of Control*, 95:216–244, 1990.
- [101] A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE PAMI*, December 2000.

- [102] B. Smyth and E. Balfe. Anonymous personalization in collaborative web search. *Inf. Retr.*, 9(2):165–190, 2006.
- [103] K. Sparck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments, part1. *Information Processing and Management*, V. 36(6):779–808, November 2000.
- [104] K. Sparck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments, part2. *Information Processing and Management*, V. 36(6):809–840, November 2000.
- [105] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *Proc. of Infocom*, 2003.
- [106] I. Stoica, D. K. R. Morris, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of SIG-COMM*, Aug. 2001.
- [107] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *Proc. of SIGCOMM '03*, 2003.
- [108] M. J. Taylor, H. Zaragoza, and S. E. Robertson. Ranking classes: Finding similar authors. Technical report, Microsoft Research, Cambridge, 2003.
- [109] C. Tomasi. Estimating gaussian mixture densities with EM : A tutorial. Technical report, Duke University, 2004. <http://www.cs.duke.edu/courses/spring04/cps196.1/handouts/EM/tomasiEM.pdf>.
- [110] Y.-H. Tseng. Content-based retrieval for music collections. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 176–182, New York, NY, USA, 1999. ACM.
- [111] A. Tveit. peer-to-paper based recommendation for mobile commerce. In *Proc. of the First International Mobile Commerce Workshop*, pages 26–29, 2001.
- [112] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, London, UK, 1979.
- [113] S. Voulgaris, A.-M. Kermarrec, L. Massoulie, and M. van Steen. Exploiting semantic proximity in peer-to-peer content searching. In *Proc. of the 10th IEEE Int'l Workshop on Future Trends in Distributed Computing Systems*, 2004.

- [114] J. Wang, A. P. de Vries, and M. J. Reinders. A user-item relevance model for log-based collaborative filtering. In *Proc. of ECIR06, London, UK*, pages 37–48, Berlin, Germany, 2006. Springer Berlin / Heidelberg.
- [115] J. Wang, A. P. de Vries, and M. J. Reinders. Unified relevance models for rating prediction in collaborative filtering. *To appear in ACM Trans. on Information System (TOIS)*, 2008.
- [116] J. Wang, A. P. de Vries, and M. J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508, New York, NY, 2006. ACM Press.
- [117] J. Wang, J. Pouwelse, J. Fokker, A. P. de Vries, and M. J. Reinders. Personalization on a peer-to-peer television system. *Special Issue on Multimedia Tools and Applications*, 2006.
- [118] J. Wang, J. Pouwelse, R. Lagendijk, and M. R. J. Reinders. Distributed collaborative filtering for peer-to-peer file sharing systems. In *Proc. of the 21st Annual ACM Symposium on Applied Computing*, 2006.
- [119] J. Wang, M. J. Reinders, J. Pouwelse, and R. L. Lagendijk. Wi-fi walkman: a wireless handheld that shares and recommends music on peer-to-peer networks. In *Proc. of Embedded Processors for Multimedia and Communications II, part of the SPIE Symposium on Electronic Imaging 2005.*, 2005.
- [120] J. Wang, S. E. Roberston, A. P. de Vries, and M. J. T. Reinders. Probabilistic relevance models for collaborative filtering. *To appear in Journal of Information Retrieval*, 2008.
- [121] J. Wang, J. Yang, M. Clements, A. P. de Vries, and M. J. T. Reinders. Personalized collaborative tagging. *Technical Report, University College London*, 2007.
- [122] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 4–11, New York, NY, USA, 1996. ACM Press.
- [123] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 114–121, New York, NY, 2005. ACM Press.

- [124] J. Yang, J. Wang, M. Clements, J. A. Pouwelse, A. P. de Vries, and M. Reinders. An epidemic-based P2P recommender system. In *Workshop on Large Scale Distributed Systems for Information Retrieval (LSDS-IR) in SIGIR07*, 2007.
- [125] H. Zaragoza, D. Hiemstra, and M. Tipping. Bayesian extension to the language model for ad hoc information retrieval. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 4–9, New York, NY, USA, 2003. ACM Press.
- [126] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342, New York, NY, 2001. ACM Press.
- [127] C. X. Zhai, W. W. Cohen, and J. Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 10–17, New York, NY, USA, 2003. ACM.
- [128] Y. Zhang, J. Callan, and T. Minka. Novelty and redundancy detection in adaptive filtering. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 81–88, New York, NY, USA, 2002. ACM.
- [129] Y. Zhang and J. Koren. Efficient bayesian hierarchical user modeling for recommendation system. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 47–54, New York, NY, USA, 2007. ACM Press.
- [130] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 22–32, New York, NY, USA, 2005. ACM.

Acknowledgements

This thesis would not have been possible without the support of many people.

I am deeply indebted to my supervisor Professor Marcel Reinders who offered invaluable guidance, support and assistance. His expertise in machine learning was highly beneficial to my research. I would also like to express my gratitude to my co-supervisor, Dr. Arjen de Vries, for his broad knowledge of information retrieval and his assistance in writing research articles. It was under their tutelage that I developed a focus and became interested in information retrieval.

A very special thanks goes out to Professor Inald Lagendijk for his financial support as well as his guidance in the first year of my study. I must also acknowledge Professor Stephen Robertson and Dr. Michael Taylor of Microsoft Research Cambridge for their supervision during my internship at Microsoft. The initial idea behind the work in Chapter 3 originated from some discussions we had.

Appreciation also goes out to the other members in the ICT group and the two projects in which I was involved: Pavel, David, and Bob for teaching me machine learning; Jacco for his Linux expertise; Yunlei for throwing parties; Jeroen for the Condensation algorithm; Umute and Alan for interesting Multimedia problems; Maarten for the vivid illustrations of collaborative tagging; Jenneke for a joint work on P2P recommender systems; Robbert and Ben for their technical assistance; Anja for her secretarial support; Johan for his enthusiastic attitude towards research and his expertise in P2P networking; and Jie for his programming skills.

Finally, I would also like to thank my father, mother, and brother for their patience, support, and impeccable understanding and in particular, I must acknowledge my wife, Huiyan, without whose love and encouragement I would not have finished this thesis.

Curriculum Vitae

Jun Wang was born in Jiangsu, China. In 1993 he graduated from Jinling high school in Nanjing, China. The same year, he started his studies in electrical engineering at Southeast University; four years later, he obtained his BE degree. In 2000, after spending three years in industry, he returned to academia and worked as a research assistant in the computer science department, the National University of Singapore, Singapore; in 2003, he obtained his MSc degree in computer science, with the title of his masters thesis “Detecting and Tracking Human Faces in the Compressed Domain from Content-based Video Indexing”.

From July 2003 to June 2007, he was a PhD student with the Information and Communication Theory Group, Delft University of Technology, the Netherlands. His PhD study was carried out within the CACTUS (Context Aware Communication, Terminal and UserS) and I-Share projects. His research focused on information retrieval, mainly personalization. In 2006, he spent three months working at Microsoft Research, Cambridge, UK.

As of July 2007, Jun has been working as a lecturer at University College London, UK. His research interests include collaborative filtering (recommender systems), information retrieval, intelligent multimedia information systems (content analysis, retrieval, and personalization). Jun has published over 30 research papers in journals and conference proceedings including IEEE Trans. on Multimedia, ACM Trans. on Information Systems, ACM Multimedia System Journal, ACM SIGMM, and ACM SIGIR. He received the best Doctoral Consortium award in ACM SIGIR 2006; He was also one of the recipients of the “Beyond Search - Semantic Computing and Internet Economics” award sponsored by Microsoft Research, USA in 2007.