The Hong Kong Polytechnic University
Department of Computing

Enriching User and Item Profiles for Collaborative Filtering: From Concept Hierarchies to User-Generated Reviews

---

LEUNG Wing Ki, Cane

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy

September, 2008

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____

LEUNG Wing Ki, Cane

_____

To Grandpa

Abstract

Collaborative Filtering (CF) is a recommender systems technique that generates personalized recommendations for users based on user preferences. Such preferences are usually expressed in the form of numerical ratings, or binary votes such as purchase data. Despite its considerable success and popularity in both research and practice, CF suffers from the problems of *data sparseness* and *cold-start recommendation*, which is an extreme form of data sparseness. Specifically, CF algorithms have difficulty with generating reliable recommendations when data are sparse, and they cannot recommend items that have not received any ratings from users.

This thesis addresses the problems of data sparseness and cold-start recommendation of CF along two dimensions. Firstly, we developed two novel recommendation algorithms based on association rule mining techniques. The proposed algorithms, namely FARAMS and CLARE, exploit the relationships between items that are encoded in the concept hierarchies of the items when users' preference data are too limited for generating recommendations. Specifically, FARAMS makes use of interesting associations between item categories to find recommendable items for users having limited known preferences, while CLARE generates recommendations for a given cold-start item by finding other items in the system that are highly correlated with the attributes of the cold-start item. We evaluated both algorithms based on widely adopted benchmarking datasets of CF. Results show that both algorithms outperform related algorithms in addressing data sparseness and the cold-start problem under similar experimental settings.

Secondly, we investigated the use of user-generated reviews for generating personalized recommendations. We made three major contributions in this area. First, we collected and analyzed a set of movie reviews to understand how user opinions are expressed in user-generated reviews, which are free-form texts written in natural language. Based on the results of our analysis, we proposed a novel method for determining the sentimental orientations and strength of user opinions. Second, we proposed a rating inference framework, namely PREF, for augmenting ratings for CF. PREF aims at determining and representing the overall sentiments expressed in reviews as numerical ratings that can readily be used by existing CF algorithms. In other words, PREF enables existing CF algorithms to utilize textual reviews as an additional source of user preferences, thereby lessens the problem of data sparseness. Third, we found that user-generated reviews contain valuable information for constructing the interest profiles of users and domain items based on a real-world dataset of tourist attraction reviews. Using such information for generating personalized recommendations significantly improve the prediction quality and coverage of traditional CF algorithms. While existing CF algorithms operate on numerical ratings or binary votes of items, our research represents an important pioneering step towards a novel CF paradigm based on user-generated reviews.

Publications Arising from the Thesis

Listed in reverse chronological order:

1. C. W. K. Leung, S. C. F. Chan, and F. L. Chung. A probabilistic rating inference framework for mining user preferences from reviews. Submitted to *World Wide Web - Internet and Web Information Systems (under revision)*.

2. C. W. K. Leung, S. C. F. Chan, and F. L. Chung. An empirical study of a cross-level association rule mining approach to cold-start recommendations. *Knowledge-based Systems*, 21(7): 515–529, October 2008.

3. C. W. K. Leung, S. C. F. Chan, and F. L. Chung. Evaluation of a rating inference approach to utilizing textual reviews for collaborative recommendation. In A. T. S. Chan et al. (Eds), *Cooperative Internet Computing*. World Scientific, pages 94–109, 2008.

4. C. W. K. Leung and S. C. F. Chan. Sentiment analysis of product reviews. In J. Wang (Eds.), *Encyclopedia of Data Warehousing and Mining - Second Edition*, Information Science Reference, pages 1794–1799, 2008.

5. C. W. K. Leung, S. C. F. Chan, and F. L. Chung. Applying cross-level association rule mining to cold-start recommendations. In *Proceedings of the IEEE/WIC/ACM WI-IAT Workshop on Web Personalization and Recommender Systems*, pages 133–136, 2007.

6. C. W. K. Leung, S. C. F. Chan, and F. L. Chung. A collaborative filtering framework based on fuzzy association rules and multiple-level similarity. *Knowledge and Information Systems*, 10(3):357–381, 2006.

7. C. W. K. Leung, S. C. F. Chan, and F. L. Chung. Integrating collaborative filtering and sentiment analysis: A rating inference approach. In *Proceedings of The ECAI 2006 Workshop on Recommender Systems*, pages 62–66, 2006.

8. C. W. K. Leung, S. C. F. Chan, and K. F. L. Chung. Towards collaborative travel recommender systems. In *Proceedings of the 4th International Conference on Electronic Business*, pages 445–451, 2004.

Acknowledgements

I would like to express my sincere gratitude to my Ph.D. supervisor, *Dr. Stephen Chan*. Thank you very much for offering me an opportunity to begin my research studies five years ago, an opportunity that has widened my horizon. Thank you for your continuous support, encouragement, guidance and patience throughout my studies, for sharing with me your perspectives of research, teaching and life as a supervisor, a teacher and a friend. I have learned a lot more from you than you realize.

I would like to thank my co-supervisor, *Dr. Korris Chung*, for all interesting discussions and directions he gave me on my studies.

I sincerely thank the chairman and members of my thesis committee. They include *Dr. James Liu*, Department of Computing, The Hong Kong Polytechnic University, *Prof. Nick Cercone*, Faculty of Science and Engineering at the York University, Canada, and *Dr. William Cheung*, Department of Computer Science, Hong Kong Baptist University. Thank you very much for all your invaluable comments and suggestions which helped improve this thesis.

Thanks must be given to the *Technical Team* and *General Office* staff, especially *Ms. Miu Tai*, of the Department of Computing at the Hong Kong Polytechnic University. Thanks for all the technical and administrative support throughout the years.

I also appreciate *Karen Tso*, from the University of Hildesheim, and *Stefan Hauger*, from the University of Freigburg, for research collaboration during my studies.

Most of all, I would like to express my deepest appreciation to my family

and friends. Thank you *Grandpa*, for your love, for your encouraging words, for everything. To my *parents* and *sisters*, thank you for supporting me, even when you are not sure what I have been doing all these years. I also thank *Bo the Cat*, *Big Head the Cat* and *Nat the Cat*, for keeping me company night after night when I was working hard writing up this thesis. *Ada*, thank you for always being there for me during my ups and downs. Thank you *Askin* and *Siu Man*, who believe in me more than I do in myself. Finally, I would like to thank my dear friends, especially *Bo*, *Chan Man Yee*, *Din*, *Elaine* and *Venus*, for always asking me "what do you actually study?" and "(when) will you get it done?". Once again, my heartfelt thanks to all of you.

Contents

Abstract i

Publications Arising from the Thesis iii

Acknowledgements v

Table of Contents vii

List of Figures xii

List of Tables xiv

1 Introduction 1

1.1 Recommender Systems and Collaborative Filtering (CF) 1

1.2 Research Challenges 3

1.2.1 Scalability 3

1.2.2 Data Sparseness 4

1.2.3 Cold-start Problem 5

1.3 Objectives 6

1.4 Contributions 6

1.5 Organization of the Thesis 9

2 Literature Review 10

2.1 Recommender Systems Overview 10

2.1.1 CF-based Recommender Systems 11

2.1.2 Content-based Recommender Systems 11

2.1.3 Knowledge-based Recommender Systems 13

2.1.4 Strengths and Weaknesses of Recommender Systems . . 14

2.2 Collaborative Filtering (CF) 17

2.2.1 Terminology and Notations 17

2.2.2 User Preferences in CF 18
2.2.3 Tasks of CF 19
2.2.4 User- and Item-based CF 19
2.2.5 Evaluating CF Algorithms 21
2.3 Association Rule Mining (ARM) 26
2.3.1 Association Rules and Their Interestingness 27
2.3.2 Adaptive-Support ARM for CF 31
2.3.3 Fuzzy Association Rule (FAR) Mining 32
2.3.4 Cross-level Association Rule (CAR) Mining 37
2.4 Text Data Mining Basics 37
2.4.1 Analyzing Text Documents 39
2.4.2 Representing Text Documents 40
2.4.3 Determining Term Weights 40
2.4.4 The Text Classification Task 43
2.5 Sentiment Analysis of User-Generated Reviews 44
2.5.1 Sentiment Analysis 45
2.5.2 Extracting Interesting Features 47
2.5.3 Determining SO and Strength of Opinions 48
2.5.4 Classifying Reviews 51

3 Alleviating Data Sparseness by Fuzzy Association Rule Mining and Item Taxonomies 54
3.1 Introduction 54
3.2 Related work on ARM-based CF 56
3.3 The FARAMS Framework 56
3.3.1 Data Preprocessing 57
3.3.2 Mining User Preferences 60
3.3.3 Predicting Scores of Recommendable Items 65
3.3.4 Generating Recommendations 66
3.4 Experimental Results 68
3.4.1 Datasets 68
3.4.2 Experimental Settings 69
3.4.3 Results and Discussions 70
3.5 Summary 76

4 Cold-start Recommendations by Cross-level Association Rule Mining **77**
4.1 Introduction . . . 77
4.2 Existing Approaches to Cold-start Recommendations . . . 78
4.2.1 The Aspect Model . . . 78
4.2.2 The Naive Filterbot Algorithm . . . 79
4.3 Problem Description . . . 80
4.3.1 Data Representation . . . 83
4.4 CLARE: Cold-start Recommendations by CAR Mining . . . 83
4.4.1 Data Preprocessing . . . 84
4.4.2 Mining Association Rules . . . 84
4.4.3 Generating Recommendations . . . 87
4.5 Experimental Results . . . 88
4.5.1 Dataset . . . 88
4.5.2 Method and Evaluation Metrics . . . 89
4.5.3 Parameters . . . 90
4.5.4 Evaluation of CLARE . . . 90
4.5.5 Comparisons with Related Work . . . 99
4.6 Summary . . . 103

5 Augmenting Ratings from Reviews for CF by Rating Inference **106**
5.1 Introduction . . . 106
5.2 Analysis of Movie Reviews . . . 108
5.2.1 Data Collection . . . 108
5.2.2 Preliminary Experiments and Observations . . . 109
5.3 PREF: A Probabilistic Rating Inference Framework . . . 114
5.3.1 Data Preparation . . . 114
5.3.2 Feature Extraction . . . 115
5.3.3 Opinion Dictionary Construction . . . 116
5.3.4 Rating Inference . . . 117
5.4 Experimental Results . . . 120
5.4.1 Method . . . 120
5.4.2 Parameters . . . 122
5.4.3 Evaluation of PREF . . . 122
5.4.4 Comparisons with Related Work . . . 127

5.5 Integrating PREF and CF 133
5.6 Summary 136

6 Towards Review-based Recommender Systems: A Case Study on TripAdvisor 138
6.1 Introduction 138
6.2 Relation to Other Work 140
6.3 Travel Reviews on TripAdvisor 141
6.3.1 Data Collection and Filtering 142
6.3.2 Data Model 143
6.3.3 Data Characteristics and Implications 144
6.4 Generating Item Predictions from Reviews 147
6.4.1 Sentiment Analysis of Reviews 147
6.4.2 Building User, Item and Category Profiles 152
6.4.3 Making Predictions 154
6.5 Experimental Study 161
6.5.1 Method 162
6.5.2 Existing Tools Adopted 162
6.5.3 Parameters 163
6.5.4 Results and Discussions 163
6.6 Summary 170

7 Conclusions 173
7.1 Summary of Contributions 173
7.2 Suggestions for Future Research 175

Bibliography 177

List of Figures

Figure 1.1 – Recommendation list associated with a book title on Amazon.com. 2

Figure 2.1 – A conceptual ratings matrix in CF. 18
Figure 2.2 – Sample fuzzy sets and membership functions. 33
Figure 2.3 – Major steps in the typical text mining process. 38

Figure 3.1 – Example of an item taxonomy. 55
Figure 3.2 – Recall rates achieved using different maximum rule lengths. 71
Figure 3.3 – Recall rates achieved using different interestingness measures for predicting the scores of recommendable items. . 72
Figure 3.4 – Recall rates achieved with and without utilizing multiple-level similarity between items. 73
Figure 3.5 – Recall rates achieved with and without using fuzzy association rules on the MovieLens dataset. 73
Figure 3.6 – Recall rates achieved with and without using fuzzy association rules on the Jester dataset. 74
Figure 3.7 – Performance of MAR and FARAMS for the Movielens dataset. 76

Figure 4.1 – Illustration of the proposed preference model comprising user-item and item-item relationships. 80
Figure 4.2 – A motivating example 81
Figure 4.3 – Recommendation quality produced using different interestingness measures (M) for predicting preferences for recommendable items, and *Plot* as attribute for mining CARs. . . 92
Figure 4.4 – Recommendation quality produced using different attributes for mining CARs, and H(*FC, CORR*) for predicting preferences for recommendable items. 95

Figure 4.5 – Coverage of cold-start items achieved using different item attributes for mining. 96
Figure 4.6 – Comparison between recommendation quality achieved for all test users, and that for users having at least 20 known ratings in the training set. CARs were mined using *Director* + *Plot* as attributes. 98
Figure 4.7 – Comparison between CLARE and MS-based recommendation (hybrid), using *Genre* as attribute. 101
Figure 4.8 – Comparison between CLARE and Naive Bayes (NB) recommender (content-based), using *Director* + *Plot* as attributes. 103

Figure 5.1 – Overview of PREF 115
Figure 5.2 – Distribution of ratings in our movie reviews dataset. . 120
Figure 5.3 – Learning curves of PREF and the majority baseline showing how (a) MAE, and (b) MSE change with respect to the size of training set in the 3-point and the 4-point settings. . 126
Figure 5.4 – Mean and standard deviation of PSP of reviews having different ratings in our dataset. 132

Figure 6.1 – A user-generated travel review on TripAdvisor. 142
Figure 6.2 – Distribution of ratings in our attraction reviews dataset. 143
Figure 6.3 – Distribution of review count by user. 144
Figure 6.4 – Distribution of review count by attraction. 145
Figure 6.5 – Tasks in the sentiment analysis of user-generated reviews. 148
Figure 6.6 – Number of distinct features versus number of reviews. 150
Figure 6.7 – Illustration of the contents of the opinion dictionary. . 151
Figure 6.8 – Summary of results achieved using SO and user-specified ratings. 164
Figure 6.9 – Summary of results achieved using various feature weighting schemes and feature selection levels. 166
Figure 6.10 – Summary of prediction quality and coverage of all prediction models. 168

List of Tables

Table 2.1 – A confusion matrix. 23
Table 2.2 – A sample decision table. 31
Table 2.3 – A sample user-item ($U \times I$) ratings matrix in CF. . . . 34
Table 2.4 – Fuzzified ratings. 34
Table 2.5 – Fuzzified ratings with normalization. 35

Table 3.1 – Transforming user preferences for (a) items and (b) item categories into transactions. 58
Table 3.2 – A relation matrix of items (e.g. movies) and their categories (e.g. movie genres). 58
Table 3.3 – Transactions containing ⟨*Item, Fuzzy Set*⟩ pairs and normalized membership degrees. 59
Table 3.4 – Transformed transactions in the vertical TID-list format for efficient support counting. 60
Table 3.5 – TID-lists of $\langle i_1, L \rangle$ and $\langle i_2, D \rangle$. 62
Table 3.6 – Performance of ASARM and FARAMS for the EachMovie dataset. 75

Table 4.1 – Important notations used to represent user preference data in CLARE. 84
Table 4.2 – Averaged statistics about training and test sets. 89
Table 4.3 – Statistics about the various attributes. 94

Table 5.1 – Top 15 opinion words with relative frequencies. 112
Table 5.2 – Top 1 opinion words with relative frequencies. 112
Table 5.3 – Summary of experimental results on PREF and the baseline algorithms. Each training set contained 1500 reviews with uniform class distribution. 123
Table 5.4 – Seed adjectives used for pruning the opinion dictionary. 125

Table 5.5 – Summary of comparisons between the majority baseline, PREF, NB classifier, a method based on Dave et al. [25], and SVR. Each training set contains 1,500 reviews with uniform class distribution. 128
Table 5.6 – Comparison with SVR: Each training set contains 4,800 reviews with uniform class distribution. 130
Table 5.7 – Comparison with SVR: Each training set contains 1,500 reviews, and retained the original class distribution of the dataset. 131
Table 5.8 – Comparison with the graph-based approach [41] to rating inference: Each training set contains 1,500 reviews, and retained the original class distribution of the dataset. 133
Table 5.9 – Performance achieved using different datasets for performing CF. 135

Table 6.1 – Brief descriptions of prediction models. 154
Table 6.2 – Relative improvements in MAE and MSE achieved using SO. 165

Chapter 1

Introduction

1.1 Recommender Systems and Collaborative Filtering (CF)

Recommender systems are automated recommendation engines designed to help users in alleviating the well-known problem of information overload. They intend to efficiently and automatically search through vast information spaces, and then recommend to users only information that are relevant to their interests [18, 139, 4, 63]. Recommender systems are also valuable tools that help e-commerce applications engage visitors and enhance online sales. For instances, they have the potential to turn browsers into customers, and may improve cross sales by suggesting additional products that are of interest to customers [139]. One of the most well-known and successful e-commerce applications employing recommender system technologies is *Amazon.com*[1], which has the following long-term vision according to Jeff Bezos:

> *"... if we have 25 million customers, we should have 25 million stores ... building a place where people can find anything they might want to buy online."*
>
> *– Jeff Bezos, CEO of Amazon.com* [169]

Amazon.com has been fulfilling its mission by providing *personalized recommendations* to its users and customers. One of the most noticeable recommendation features Amazon.com provides might be that when a user browses the

[1] Amazon.com: http://www.amazon.com

Figure 1.1: Recommendation list associated with a book title on Amazon.com.

information page of a book title, (s)he is presented with a recommendation list showing what other book titles have been bought with the selected book title as Figure 1.1 shows. The underlying technique for generating this recommendation list is known as *Collaborative Filtering* (CF).

CF has been well-acknowledged to be the most promising recommender systems technique. It has achieved great success in both research and practice since its introduction in the early 90's [42, 127, 144, 97]. CF provides personalized recommendations to users based on user preferences, usually in the form of user-specified ratings or previous interactions between the users and the system (e.g. purchase or browsing histories). Traditionally, CF exploits the similarities between users for generating recommendations. Such *user-based CF* systems compare the known preferences of a given user, known as the *active user*, with the known preferences of other users to find the k most similar users, known as the k-nearest neighbors (k-nn), of the active user. They then predict the preference of the active user for a particular item based on the preferences of his/her neighbors for that item. Alternatively, CF-based systems may suggest to the active user a list of N items that were of interest to his/her neighbors.

As CF-based systems make recommendations based on user preferences, they are particularly useful for recommending taste-based items. For examples,

they have been successfully applied to the recommendation of books [90], movies (e.g. MovieLens[2]), jokes [43], music [144, 7] and television programs [148].

1.2 Research Challenges

CF-based systems suffer from several challenges despite their success and popularity. In what follows, we describe, among others, the three most well-known challenges of CF. They include scalability, data sparseness and the cold-start problem. We also outline general approaches for addressing each of those challenges.

1.2.1 Scalability

Traditional user-based CF lacks *scalability* due to its *memory-based* nature. It determines the k-nn of the active user by making statistical computations over the entire database of user preferences. Such a similarity computation step is done in an online phase before recommendations can be made [136]. This leads to severe latency for generating recommendations, and the problem may get worse as the number of users in the system grows over time.

Researchers have proposed a class of *model-based CF* algorithms in view of the poor scalability of user-based CF. Model-based CF algorithms construct *compact models* about users or items using various data mining techniques. They then generate recommendations based on the compact models rather than the entire database. Note that such models are constructed offline to ensure the scalability of model-based algorithms, and they serve as a reduced information space for generating recommendations. For instance, Ungar and Foster [158] proposed to group similar users or items into clusters based on training data, and generate recommendations for an active user or a given item from the cluster to which the user or item belongs.

The *item-based CF* model proposed by Sarwar et al. [136] can be considered a model-based variant of user-based CF. Item-based CF exploits the similarities between items, and recommends to the active user items that are similar to those that the user has previously shown a preference for. Sarwar et al. observed

[2]MovieLens: http://movielens.umn.edu

that neighbors of items in a recommender system are relatively more static than those of users. Similarities between items can therefore be pre-computed offline to reduce the online computations required for generating recommendations.

1.2.2 Data Sparseness

Data sparseness poses a challenge to CF because the set of items examined by a particular user is usually very small in a given system (imagine the number of book titles available on Amazon.com and the number of books that an active user could have purchased from it). CF generates recommendations based on similarities between users or items as noted. Such similarities are derived from overlapping items, or *co-rated items*, in users' known preferences. In a system with large user and item spaces, the number of co-rated items between an active user and his/her neighbors might be very limited. This may result in less reliable computations of similarity or correlation between users, which may in turn result in less accurate recommendations [14, 138, 109].

There are three major strategies for addressing data sparseness in CF. The first strategy attempts to improve the density of a ratings dataset by injecting ratings into it. Breese et al. [14] described a simple default voting scheme that assigns a default vote to items that have neither been rated by the active user nor by his/her neighbors. Some researchers made use of rating robots that rate new items based on their quality as reflected by their content-based information, such as the number of misspelled words in an article [137, 44]. The second strategy is to apply dimensionality reduction methods to sparse datasets, thereby increases the density of the datasets and improves the results of similarity computations between users [135, 138]. The third strategy is to incorporate content-related features about domain items into collaborative filters, resulting in *hybrid content- and CF-based recommendations*.

There exist various methods for generating hybrid recommendations. A typical method is to combine content- and CF-based methods sequentially by computing similarities between users or items based on content analysis first, and then generating collaborative recommendations using the resulting similarities (e.g. [9]). Another method computes content-based and CF-based recommendations simultaneously. It then determines the final recommendation results by combing both types of recommendations based on some weighting schemes (e.g.

[23, 103]). A similar method described in [109] computes similarities between items based on their semantic attributes, which are content-based information, as defined in an domain-specified ontology. It then uses such semantic similarities along with item similarities computed from ratings data for performing item-based CF.

1.2.3 Cold-start Problem

The cold-start problem, also known as the early-rater problem and the ramp-up problem [137], is a crucial shortcoming of CF. As pure CF generates recommendations solely based on user preferences, it cannot recommend new items that have not yet been observed or rated by users in the system. Similarly, CF cannot provide personalized recommendations to new users who have not yet expressed any preferences. Note that the cold-start problem does not only apply to new items and users. It also happens when no neighbor can be found for items or users due to the lack of overlapping preferences.

The cold-start problem is an extreme form of data sparseness. Thus, it has been addressed using methods similar to those for addressing data sparseness. For examples, several researchers explored the use of aspect models for generating cold-start recommendations [121, 140]. Such models are actually hybrid content- and CF-based recommendation models. They generate recommendations for a given cold-start item by estimating the probability that an active user would like the attributes it possesses. Park et al. [117] described the use of the naive filterbot algorithm for addressing the cold-start problem. The algorithm injects pseudo users, or bots, into a recommender system. The bots then generate user ratings according to the attributes of the items or the users in the system as an attempt to increase the density of the ratings matrix.

Solutions for the cold-start problem in the literature mostly focus on the new item problem. As Schein et al. [140] suggest, however, the new user problem is actually symmetric if user attributes (demographic data) are available. There are also studies on acquiring preferences of new users and on generating recommendations for new users [105, 125, 76]. For example, Middleton et al. [105] made use of an ontology to construct interest profiles of new users for research article recommendation. They defined an ontology that models people, projects, papers (articles), events and research interests, which are considered

important concepts related to research articles. They then populated such an ontology from a personnel database and a publication database as initial user profiles. Specifically, their work makes use of the research publication list of a new user, who has not interacted with their system before, to establish his/her initial interest profile, based on which recommendations are generated.

1.3 Objectives

The primary objectives of this thesis is to alleviate two crucial challenges of CF, namely data sparseness and the recommendations of cold-start items. Pure CF algorithms that operate solely on user preferences cannot recommend cold-start items as noted. In order to solve the cold-start problem, the proposed methods are hybrid methods that exploit information sources other than user preferences for enriching the interest profiles of users and items. Further, the proposed methods are model-based due to the scalability issue of memory-based (user-based) CF, although we do not explicitly discuss the scalability issue of CF in this thesis. For the proposed methods to be meaningful, we base our studies on real world datasets that can simulate the problems of data sparseness and cold-start recommendations in CF.

1.4 Contributions

We address the problem of data sparseness and cold-start recommendations along two dimensions. Firstly, we proposed novel methods for generating hybrid recommendations based on concept hierarchies of items. This area of work focuses on *how* recommendations are generated. Secondly, we proposed and investigated into the use of user-generated reviews for generating personalized recommendations. From a CF perspective, our work in this area is concerned with *what* information about user preferences and domain items to consider in the recommendation process. However, our work is not only related to CF, but also to text mining and computational linguistics. This is because our work involves understanding user preferences expressed in user-generated reviews, which are free-form texts written in natural language.

The major technical contributions made in this thesis are summarized as follows:

Hybrid recommendation techniques: We proposed in Chapters 3 and 4 two hybrid recommendation techniques based on Association Rule Mining (ARM). ARM allows for the flexibility to integrate concept hierarchies into the rule mining process. By taking advantage of this, and by utilizing taxonomies as well as attributes of domain items, we addressed the problems of data sparseness and cold-start recommendations in CF. We validated our proposed techniques against related techniques based on standard benchmarking datasets for CF.

Understanding user preferences in user-generated reviews: In Chapter 5, we presented an experimental study on the use of opinion words in a real world dataset of movie reviews. Based on the results of the study, we identified a shortcoming of a class of existing methods that estimates the *sentimental orientation* of opinion words based on the semantic similarity between the words, and proposed a novel method for determining the sentimental orientation and strength of opinion words.

Using user-generated reviews for personalized recommendations: In Chapter 5, we proposed a probabilistic *rating inference* approach for estimating the overall sentiments expressed in reviews and representing those sentiments as numerical ratings. Such ratings can be fed into existing CF algorithms for generating personalized recommendations. We demonstrated experimentally the benefit of using the rating inference approach for augmenting ratings for CF.

In Chapter 6, we investigated further into the use of user-generated reviews for generating personalized recommendations. Our work attempts to construct user profiles based on their preferences expressed in reviews, and makes use of review contents to derive similarities between items and those between item categories. We performed an experimental evaluation of eight prediction models, including rating-based, review-based and non-personalized prediction models. Results suggest that user-generated reviews contain valuable user preferences that can be utilized for making personalized recommendations. Further, review-based models improve greatly the coverage of traditional rating-based CF models, with comparable or significantly better prediction accuracies. This indicates that deriving similarities between items and those between item categories based on feature terms extracted from reviews effectively addresses data sparseness and the cold-start problem with no loss of accuracy.

A recent survey of the state-of-the-art and possible extensions of recommender systems suggests that recommender systems that utilize textual reviews

by text mining techniques are yet to be developed [4]. However, we report the use of sentiment analysis, which is a text mining task, for understanding and extracting user sentiments in textual reviews. We also studied two different approaches for using such sentiments for generating personalized recommendations. As aforementioned, the rating inference approach we proposed allows existing CF algorithms to utilize user-generated reviews as an additional source of user preferences. The review-based prediction models we presented in Chapter 6 are developed along the idea of item-based CF, which is a classical CF model that is generally acknowledged to be promising in the literature. More importantly, we demonstrated the utilization of user-generated reviews for generating personalized recommendations in the tourist attraction domain, in which tremendous collections of textual reviews are available but most existing recommender systems are built upon knowledge-based techniques. The significance of what we discussed in this paragraph is that our research represents an important pioneering step on review-based recommendations. Further, our work opens up interesting opportunities for researchers to extend CF to domains where rating-based recommendations are not common.

We noticed that some researchers have previously raised the possibility of using user opinions in reviews for generating personalized recommendations. Pang and Lee [114] mentioned that inferring fine-grained ratings from reviews would be helpful for performing CF. Several researchers confirmed, by means of user studies, the usefulness of user-generated reviews in helping users making decisions about online purchases [131, 78, 46]. Further, there exist e-commerce websites that provide recommendations to users based on user-generated reviews. For instance, BooRah[3] provides restaurant recommendations using their patent-pending technologies. Unfortunately, such technologies are not known to the public. To the best of our knowledge, our work is the first on reporting empirical research that systematically studies the use of user-generated reviews for generating personalized recommendations. We hope our work can compel further research into this area of work, which we believe is very interesting and useful given the current developments and popularity of Web 2.0 technologies.

[3]BooRah restaurant search: http://www.boorah.com/restaurants/

1.5 Organization of the Thesis

The remainder of this thesis is organized as follows:

Chapter 2 is a literature review that provides background information for our discussions in later chapters. It also reviews previous and related studies on CF and sentiment analysis.

Chapter 3 presents a CF framework that addresses the problem of data sparseness by the use of fuzzy association rule mining and item taxonomies.

Chapter 4 discusses our effort on addressing the problem of cold-start recommendations in CF. Our approach is based on a preference model comprising user-item and item-item relationships, as well as the cross-level association mining technique.

Chapter 5 investigates the use of sentiment analysis for eliciting user preferences from user-generated reviews. It describes our experimental study on the use of opinion words in a movie reviews dataset. It also presents our initial effort on integrating sentiment analysis and CF by rating inference, which aims at mapping the overall sentiments expressed in textual reviews onto a numerical rating scale.

Chapter 6 looks further into the use of user-generated reviews for modeling user preferences in CF, and for deriving similarities between domain items and their categories. It presents an experimental evaluation of prediction models that make predictions for users based on different forms of user preference information.

Chapter 7 concludes with a summary of our research findings and suggestions for future research.

Chapter 2

Literature Review

2.1 Recommender Systems Overview

Recommender systems are automated recommendation engines designed to address the well-known problem of information overload [18, 139, 4, 63]. Specifically, they receive information from users about their needs, and then help users navigate through a large information space by recommending to them information that they may be interested in. Such recommended information may be of different forms, such as books to purchase, movies to watch, and articles to read. All recommender systems make use of information about users to generate recommendations. Such information about users are usually referred to as user profiles, user models, or user preferences. Recommender systems that exploit content information about domain items also maintain item profiles, which are content descriptors of items. Examples include the topics of books [178] and the cast of movies [103].

Recommender systems can be classified into three major types based on the underlying recommendation techniques they employ: social-filtering- or CF-based, content-based, and knowledge-based [18, 4]. Moreover, there are hybrid systems that combine some of these recommendation techniques to avoid the pitfalls of individual techniques [9, 10, 19, 103]. The following subsections introduce the three major types of recommender systems and comment on their strengths and weaknesses.

2.1.1 CF-based Recommender Systems

The term "*Collaborative Filtering*" (CF) was coined by Goldberg et al. [42], for an information filtering technology in which "people collaborate to help one another perform filtering by recording their reactions...". CF-based recommender systems are sometimes referred to as social-filtering-based recommender systems in the literature. Pure CF-based systems generate personalized recommendations solely based on user preferences, which are subjective evaluations of users [42, 127, 144, 97]. As such, they are particularly useful for recommending taste-based items, such as movies, audio CDs, and jokes [139, 43]. The underlying philosophy of CF is that each individual user belongs to a larger group of like-minded users. CF-based systems therefore maintain preference data about users' purchasing habits or interests, and use such data to identify groups of similar users. They then recommend to a given active user items liked by other, similar users.

Tapestry [42] is the first CF-based system. It allows users to annotate electronic documents they have read, for example, as "interesting" or "uninteresting". Such annotations can be accessed by other users, to help them decide which documents to (or not to) read. CF-based systems have later become fully automated by tracking user preferences through users' interactions with the systems. GroupLens [127, 75], a CF-based system of Usenet articles, gathers ratings on articles from users. It then makes numerical predictions about how much a user would like an article that (s)he has not read before, based on the ratings given by similar users who have read that article. Most other CF-based systems work in a similar way. Examples are MovieLens, which gathers user ratings on movies for performing CF, and Amazon.com, which generates recommendations based on the purchase histories of previous customers.

2.1.2 Content-based Recommender Systems

Content-based recommender systems have their roots in Information Retrieval (IR) and Information Filtering (IF) systems [35, 11, 8]. They model information needs of users and generate recommendations based on the contents or semantic knowledge, such as keywords, taxonomies and ontologies, of domain items [110, 104, 4]. Specifically, a content-based recommender system analyzes the features of items preferred by the active user. It then compares the features of those items

to the features of other items that have not been observed by the active user in the system. Finally, it recommends the more relevant items, as defined by some similarity measures, to the active user.

Content-based techniques are more commonly used in textual application domains, such as books [110] and research articles [104, 105], in which content information about domain items is rich and easy-to-obtain. The Quickstep system proposed by Middleton et al. [104], for example, is a research article recommender system. It uses an ontology that captures the is-a hierarchy of research article topics to represent domain knowledge and users' interest profiles. Specifically, it monitors the research articles browsed by a certain active user, and classifies those articles based on the ontology to determine the topics that the user is interested in. It then recommends to the user other articles whose topics are related to his/her interests. While Middleton et al. considered Quickstep to be a hybrid content- and CF-based recommender system, we point out that the recommendation technique Quickstep employs is actually content-based. It is a collaborative system in the sense that it allows an active user to provide feedbacks, which are then made available to other users of the system. Such feedbacks include the suggestion of new research articles, and the correct topics of the recommended research articles that the user deemed to be wrong. In this light, the way that the Quickstep system supports CF is different from the automated CF process this thesis and most recent studies on CF are concerned with.

Hybrid content- and CF-based recommender systems

Content-based techniques are commonly combined with CF techniques to build hybrid recommender systems, because these two kinds of techniques complement the shortcomings of each other in general. For instance, content-based techniques are able to generate recommendations for all domain items, including *new* or *cold-start* items, as long as content information about the items are available. In contrast, CF can generate recommendations for items only if they have been rated by users.

The Fab system [9], for example, combines both content- and CF-based techniques for web page recommendation. Fab employs content-based techniques to analyze important keywords of web pages and to build interest profiles of users. At the same time, it collects users' ratings on web pages in order to

identify users with similar interests, so that collaborative recommendations can be provided to users. Another example is the *naive filterbot* algorithm described in [117], which injects *pseudo users*, or *bots*, into a movie recommender system. The bots augment user ratings for CF based on the attributes of movies, such as movie genres and cast. Such ratings are then combined with user-specified ratings to generate collaborative recommendations. The work of Mobasher et al. [109] exploits semantic similarities, which are content-based, between items with the aid of an ontology of domain items. Specifically, it computes semantic similarities between items based on the semantic attributes their share. It then uses such similarities along with the similarities between user ratings received by the items for performing CF.

As noted, content- and hybrid content- and CF-based recommender systems exploit semantic knowledge, such as ontologies or is-a hierarchies, about domain items for generating recommendations. Empirical studies in the area of recommender systems, including ours, often make use of existing semantic knowledge collected from prominent information sources of the domains concerned. For examples, our work and several others on movie recommendation (e.g. [103, 109, 141]) collect semantic knowledge about movies from MovieLens and the Internet Movie Database (IMDb)[4], while the work of Mooney [110] on book recommendation extracts information about books from Amazon.com. We nonetheless point out that there exist studies that employ sophisticated Information Extraction (IE), Natural Language Processing (NLP) and/or machine learning techniques for the acquisition and learning of such semantic knowledge. Examples include [96, 160, 22].

2.1.3 Knowledge-based Recommender Systems

Knowledge-based systems are complex systems that make use of *functional knowledge* about users and domain items to generate recommendations [18]. They maintain knowledge about how a particular item meets a particular user need, and based on which they use a reasoning process to generate possible recommendations that best fit a given user need. Entree [19], for example, is a knowledge-based restaurant recommender system. It generates recommendations to users by Case-based Reasoning (CBR), a problem solving skill that

[4]The Internet Movie Database (IMDb): http://www.imdb.com

attempts to solve the current problems by adapting solutions for previous, similar problems [1, 163]. Entree models each restaurant and its attributes, such as its type of cuisine, as a case. It also maintains similarity relationships between the cases, for example, whether one case is *more expensive* than the other, determined based on users' perception on the cases. As such, when the active user suggests that the current recommended case (restaurant) is too expensive, Entree is able to retrieve less expensive cases from its knowledge base.

While content-based and CF-based techniques are popular in textual domains and taste-based domains respectively, knowledge-based techniques are commonly used in domains where the decision making process of users is more complicated and constrained. A typical example of such domains is travel and tourism, in which most recommender systems are built with extensive knowledge contributed by domain experts [128, 94, 129]. DieToRecs [33] and NutKing [130] are examples of knowledge-based travel recommender systems. They maintain a knowledge base of travel products, including tourist spots, events, activities, and accommodations. These products are recommended by domain experts with respect to different travel settings, such as the time of travel and duration of a trip. Given the travel needs of an active user, both DieToRecs and NutKing use the CBR technology to match the specified needs with the knowledge base they maintain to generate recommendations.

A complete review of knowledge-based recommender systems and the CBR technology is beyond the scope of this thesis. Related discussions and surveys can be found in [18, 19, 95].

2.1.4 Strengths and Weaknesses of Recommender Systems

Each of the three major types of recommender systems has its strengths and weaknesses, as summarized below.

CF-based recommender systems

CF-based recommender systems are best known for their ability to make taste-based, personalized recommendations. They offer three major advantages over the other two types of recommender systems [42, 127, 97, 144]. Firstly, as they do not take into account content information about items, they can make recommendations for items that are not computer-parsable. Secondly, ignoring

content information allows CF systems to generate recommendations based on user tastes rather than the objective properties of domain items themselves. This means that CF-based systems can recommend items very different (content-wise) from those that the active users had previously shown a preference for, thereby overcomes a major shortcoming of content-based recommender systems [144]. Lastly, CF algorithms are much simpler and easier to implement than knowledge-based ones. For instance, building knowledge-based recommender systems requires an extensive domain knowledge engineering process, whereas CF systems can be fully automated. Consequently, CF can easily be applied to domains where a database of user preferences is available. For these reasons, CF has been the most popular and successful type of recommender systems to date. It has already been successfully applied to various application domains, for example, electronic documents [42], Usenet articles [127, 75, 137], movies (e.g. MovieLens), jokes [43], books [90], music [144, 7], and web pages [146].

CF-based recommender systems, however, are not without problems as described in the Introduction of this thesis (Chapter 1.2). Firstly, user-based CF suffers from poor scalability due to their *memory-based* nature. It makes statistical computations over the entire ratings matrix in order to make predictions for the active user. Specifically, it involves a similarity weighting step that computes the similarity between the preferences of the active user and those of *all other users* in the system. As the numbers of users and items are usually very large and ever increasing in a system, this similarity weighting step, if done in real-time, becomes a performance bottleneck of user-based CF.

Secondly, CF suffers from the problem of data sparseness. Data sparseness arises because in a given database, the set of items rated or observed by a particular user is usually very small. This implies that a complete set of ratings across all items in the system may not exist. Making predictions for users and items having limited ratings data can be difficult. Despite the success of CF and the considerable amount of CF research in the past years, data sparseness stills remains as an open and crucial challenge to researchers.

Thirdly, CF suffers from the cold-start problem, also known as the early-rater problem and the ramp-up problem [137]. The cold-start problem is an extreme form of data sparseness. It arises when no recommendations can be generated for items with no or very few ratings data. The cold-start problem applies not only to items, but also to users with no or limited known preferences.

Lastly, pure CF-based techniques cannot derive correlation between two similar but not identical items if they have never been rated by the same user. Similarly, such techniques cannot relate two users who have never co-rated the same item. These are known as the non-transitive association problems [69], which arise because the similarity and relatedness between users and items are solely derived from ratings data in CF.

Content-based recommender systems

Content-based recommendation techniques analyze the feature similarity between domain items, and recommend items that are the most similar to those that the active user has previously liked. The main advantage of content-based techniques is that they can generate recommendations for all domain items as long as content information about them are available. Pure content-based techniques, however, have three main drawbacks. Firstly, defining meaningful content descriptors for items in non-textual and taste-based domains, such as audio and jokes, can be difficult. Utilizing meta-data (item attributes), such as music genres, about domain items can improve recommendations, but this does not address the problem of eliciting and analyzing content information about non-textual items [102].

Secondly, content-based techniques suffer from over-specialization, meaning that recommendations they make are restricted to items that are similar (in terms of their contents) to items that the active user has previously seen [9].

Thirdly, content-based techniques cannot analyze the subjective aspects of domain items. A content-based newspaper recommender system, for example, cannot distinguish between high-quality and poor-quality articles on the same topic, but this can be an important issue to consider when generating recommendations for users [23].

Knowledge-based recommender systems

Knowledge-based recommender systems are knowledge-rich. They operate on knowledge about domain items and users, as well as functional knowledge about what and how items can meet a specific user requirement. They are therefore able to generate rather complex recommendations that maximize users' satisfaction [19]. While CF- and content-based systems usually deal with one single type of

domain items, knowledge-based systems are able to recommend more complex solutions, such as a travel plan that bundles heterogeneous but related products in a travel recommender system (e.g. [33, 130]). Further, they do not suffer from the cold-start problem because their knowledge bases are pre-constructed, usually with the help of domain experts. These, however, imply that knowledge-based systems require an extensive domain engineering process for building knowledge bases. This is the main limitation of knowledge-based systems, as compared to CF-based systems that can be fully automated. Another drawback of knowledge-based systems is that the recommendations they generate are rather static [19]. Further, they require more feedbacks and involvements from an active user than the other two types of recommender systems in order to arrive at an appropriate solution for the user.

2.2 Collaborative Filtering (CF)

CF is the core technique related to this thesis. In the following subsections, we first introduce important terminology and notations related to CF. We then describe user preferences that are used, or can be used, for performing CF. Next, we introduce the two common tasks of CF, followed by descriptions of the classical user- and item-based CF models, as well as evaluation metrics for measuring the performance of CF algorithms.

2.2.1 Terminology and Notations

We first define the terminology and notations related to CF. In a CF-based recommender system, there exist a set of *users* and a set of *items*. Preferences data of users in a pure CF-based system are represented as a user-item *ratings matrix*, depicted in Figure 2.1. The ratings matrix encodes the relationships between user and items in the system. The person who seeks a recommendation is called the *active user*. The item for which a prediction (e.g. predicted rating) is to be made is known as the *target item*.

Formally, in a CF-based recommender system:

- $U = \{u_1, u_2, u_3...u_m\}$ is a set of users.
- $I = \{i_1, i_2, i_3...i_n\}$ is a set of items.

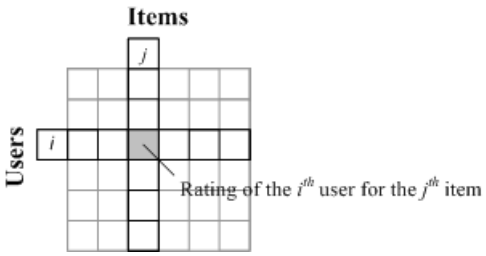

Figure 2.1: A conceptual ratings matrix in CF.

- $R = U \times I$ is the user-item ratings matrix.
- $a \in U$ is the active user seeking recommendations.
- $t \in I$ is the target item for which prediction is to be made.
- I_a is the set of items user a has examined.
- $r_{u,i}$ is the rating of user u for item i, if any.
- $p_{a,t}$ is the predicted rating of item t for user a.

These notations are used throughout this thesis.

2.2.2 User Preferences in CF

A pure CF algorithm operates on a database of user preferences, represented as a $U \times I$ ratings matrix R as aforementioned. Elements in R may represent *actual ratings* users have given items. Such ratings are usually collected explicitly by asking users to give scalar ratings on items they have examined. In MovieLens, for example, users are asked to rate movies based on a five-point rating scale.

Elements in R may also be users' *binary votes* for items. In such case, $r_{u,i}$ = 1 if user u has examined item i, and $r_{u,i}$ = 0 otherwise. Examples of binary votes include purchase histories and clickstream data, which are captured implicitly based on users' interactions with the system. In Amazon.com, for example, when a user purchases a book, (s)he is considered to be giving a positive vote on the book.

With the advent of Web 2.0, user-generated reviews are now a popular means for users to express their comments or preferences on items that they

have examined. Review hubs such as Epinions.com[5] and IMDb, as well as e-commerce web sites such as Amazon.com, allow end-users to provide reviews in free-text format in addition to numerical ratings. Such reviews can also be considered a type of "ratings", although they are natural language texts. To the best of our knowledge, however, no existing CF-based system in the research community utilizes user-generated reviews for personalization purpose.

2.2.3 Tasks of CF

Generally speaking, CF aims at recommending items that the active user a may be interested in but has not yet observed. Such a task can be of two forms, namely *prediction* and *recommendation* [136].

The task of prediction is to compute $p_{a,t}$, which is a predicted rating indicating how much the active user a may like the target item t. Note that $t \notin I_a$. The task of recommendation is to generate a list of N items, I_r, that user a may like. Note that $I_r \in I$ and $I_r \cap I_a = \emptyset$. I_r is usually a ranked list, with more interesting items ranked higher on the list. This task is commonly referred to as *Top-N recommendation*.

2.2.4 User- and Item-based CF

User-based and item-based CF are classical CF techniques that are well-known for their simplicity and prediction accuracy, which tends to improve as the active user rates more items in the system [127, 14, 54]. User-based CF first finds the k most similar users (k-nn) of the active user. It then predicts how much the active user would like a target item based on his/her ratings data and the preferences of his/her neighbors. Item-based CF, in contrast, exploits similarities between items based on the ratings they received from users. It makes predictions for the active user based on how (s)he has rated items that are similar to the target item.

We first detail the idea of **user-based CF**. Given an active user a and a target item t, user-based CF predicts $p_{a,t}$ in three steps, namely similarity weighting, neighbor selection, and prediction computation. The similarity weighting step computes the weight of *each user* in $\{u|u \in U \ and \ u \neq a\}$ with respect to his/her similarity with the active user. Similarities are reflected in the ratings that users have given items. For two particular users to be comparable, only co-rated

[5]Epinions.com: http://www.epinions.com/

items, which are items that both users have rated, are counted. The most widely used method for computing the similarity weight between two users a and u, or simply $w(a,u)$ is the *Pearson correlation coefficient*. It reflects the correlation between users a and u, and is defined as follows [127, 14]:

$$w(a,u) = \frac{\sum_{j \in I_c}(r_{a,j} - \bar{r}_a)(r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{j \in I_c}(r_{a,j} - \bar{r}_a)^2 \sum_{j \in I_c}(r_{u,j} - \bar{r}_j)^2}} \tag{2.1}$$

where I_c denotes the set of items co-rated by both users a and u. That is, $I_c = \{I_a \cap I_u\}$, and j is an item in I_c. $\bar{r}_a$ is the average of all ratings user a have given items in the training set, $r_{a,j}$ is the rating of user a for item j, and similar for $\bar{r}_u$ and $r_{u,j}$.

The neighbor selection step takes place after computing similarity weights between the active user and all other users in U. This step simply selects the set of k users having the highest similarity weights with user a. Such users, who are the k-nn of user a, are used as predictors for the target item t in the next step, the prediction computation step.

The prediction computation step estimates $p_{a,t}$ based on $\overline{r}_a$, and the weighted sum of the ratings on item t given by the k-nn of user a:

$$p_{a,t} = \overline{r}_a + \alpha \sum_{u \in knn(a)} w(a,u)(r_{u,t} - \overline{r}_u) \tag{2.2}$$

where $u \in knn(a)$ denotes the k-nn of user a determined in the previous neighbor selection step. α is a normalizing factor such that the absolute values of similarity weights w(a,u) sum to unity.

The drawback of user-based CF is that it suffers from poor scalability as aforementioned. This is due to the performance bottleneck of user-based CF in the similarity weighting step, which computes $w(a,u)$ between user a and all other users in the system for determining $knn(a)$ in real time.

Sarwar et al. [136] proposed the **item-based CF** paradigm in view of the scalability issue of user-based CF. Item-based CF exploits the similarities between items instead of those between users for making predictions. Sarwar et al. observed that the item space in a recommender system is relatively static as compared to the user space. They therefore proposed to compute similarities between items offline. When user a seeks a prediction for item t, item-based CF determines the set of k-nn for item t from pre-computed similarity weights. It

then estimates $p_{a,t}$ based on how user a has rated the k-nn of t, which are items that are similar to item t.

The similarity weight between two items t and i can also be computed using Pearson correlation coefficient as in user-based CF. In item-based CF, however, this is computed based on the ratings t and i received from users who have rated both items:

$$w(t,i) = \frac{\sum_{u \in U_c}(r_{u,t} - \bar{r}_t)(r_{u,i} - \bar{r}_i)}{\sqrt{\sum_{u \in U_c}(r_{u,t} - \bar{r}_t)^2 \sum_{u \in U_c}(r_{u,i} - \bar{r}_i)^2}} \tag{2.3}$$

where U_c denotes the set of users who have rated both items t and i, and $\bar{r}_i$ is the mean rating received by item i.

In item-based CF, $p_{a,t}$ is computed as the weighted sum of the ratings user a has given the k-nn of item t:

$$p_{a,t} = \alpha \sum_{i \in knn(t)} w(t,i)(r_{a,i}) \tag{2.4}$$

where $i \in knn(t)$ denotes the set of k-nn of item t, $r_{a,i}$ denotes the active user's rating on item i, and α is a normalizing factor such that the absolute values of similarity weights $w(t,i)$ sum to unity.

2.2.5 Evaluating CF Algorithms

CF algorithms can be systematically evaluated along three major dimensions: *statistical accuracy*, *decision support accuracy* and *coverage*. The following subsections describe the commonly used evaluation metrics under each of these dimensions.

Statistical accuracy

Statistical accuracy metrics are suitable for evaluating algorithms that generate numerical predictions given a target item for an active user. They compare the difference between the predicted ratings generated by an algorithm against the user-specified ratings. Two commonly adopted statistical accuracy metrics are Mean Absolute Error (MAE) and Mean Squared Error (MSE) [14, 55].

- **Mean Absolute Error (MAE)**, as its name implies, is the mean of the absolute errors made by an algorithm, using the actual user-specified

ratings as ground truth. MAE is usually computed on a per-user basis, meaning that one MAE is computed for each active user in the test set. The resulting MAE scores are then averaged over all the users in the test set to obtain the final MAE of an algorithm. The MAE of a user u, denoted by MAE_u, is defined as:

$$MAE_u = \frac{1}{|I_u|} \sum_{i \in I_u} |p_{u,i} - r_{u,i}| \tag{2.5}$$

where I_u is the set of withheld ratings of user u in the test set, $p_{u,i}$ and $r_{u,i}$ respectively represent the predicted and the actual rating of item i for user u.

- **Mean Squared Error (MSE)** emphasizes large errors made by an algorithm. The MSE of user u, denoted by MSE_u, is computed as follows:

$$MSE_u = \frac{1}{|I_u|} \sum_{i \in I_u} (p_{u,i} - r_{u,i})^2 \tag{2.6}$$

  Similar to MAE, the overall MSE produced by an algorithm is given by the average of the set of per-user MSE scores.

We used MAE and MSE in our experimental studies to evaluate algorithms that make numerical predictions. The lower the values of MAE and MSE of an algorithm, the more accurate the predictions are.

Decision support accuracy

Decision support accuracy metrics are mainly used for evaluating Top-N recommendation algorithms [55]. Commonly used decision support accuracy metrics include precision, recall, F-measure (or F_1 measure), classification accuracy (or simply accuracy), Receiver Operator Characteristics (ROC), and rank score. Employing these metrics basically requires casting the recommendation task as a binary classification task. An example is to classify an unseen movie of an active user as *Like* or *Does Not Like* [13, 55, 141].

We first introduce Table 2.1, a *2 × 2* confusion matrix for computing some of aforementioned metrics. Columns and rows in the confusion matrix respectively describe the actual (**Relevant = Y/N**) and the predicted (**Recommended = Y/N**)

Table 2.1: A confusion matrix.

	Recommended = Y	**Recommended = N**
Relevant = Y	TP	FN
Relevant = N	FP	TN

classification of items generated by a Top-N recommendation algorithm. In the matrix, TP (True Positive) is the number of recommended items (**Recommended = Y**) that are relevant to the active users' interests (**Relevant = Y**), FP (False Positive) is the number of recommended items (**Recommended = Y**) that are in fact irrelevant (**Relevant = N**), and similar for FN (False Negative) and TN (True Negative).

We now define the various aforementioned decision support metrics.

- **Precision** is the portion of recommended items that are in fact relevant. Referring to Table 2.1, precision is defined as:

$$Precision = \frac{TP}{TP + FP} \tag{2.7}$$

  The sum of TP and FP in the equation corresponds to the total number of recommendations provided to the active users.

- **Recall** is the portion of relevant items (in the withheld test set) that were recommended to the active users. It is defined as:

$$Recall = \frac{TP}{TP + FN} \tag{2.8}$$

- **F-measure**, also known as the $\mathbf{F}_1$ measure, combines precision and recall into a single metric. This metric is desirable because precision and recall are inversely related, and it is easy to optimize either one separately [13]. For instance, using a larger value of N, which tends to increase the chance of recommending relevant items to the active users, would result in higher recall but lower precision. The most widely-adopted method for computing the F_1 measure is to take the harmonic mean of precision and recall:

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.9}$$

Such a definition gives equal importance to precision and recall, and the resulting value tends strongly towards the smaller value of the two.

- **Classification accuracy**, or simply **accuracy**, is another metric that is computed based on the confusion matrix in Table 2.1. It is defined as:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.10}$$

The major difference between accuracy and precision, as well as that between accuracy and recall is that accuracy takes TN (the number of true negatives) into account.

- **Receiver Operator Characteristics (ROC)** plots the true-positive rate (*tpr*) against the false-positive rate (*fpr*) produced by a classifer under different classification thresholds, for instance, thresholds applied to the predicted preferences for recommendable items. *tpr* is equivalent to recall in Eq. (2.8), while *fpr* is computed based on Table 2.1 as follows:

$$fpr = \frac{FP}{FP + TN} \tag{2.11}$$

A ROC curve is usually used with an associated metric, known as the Area under the ROC curve (AUC). AUC is a single metric that summarizes the performance of a classifier across all possible thresholds. The larger the AUC of a classifier, the more successful it can distinguish between positives and negatives.

Details about ROC analysis, including how ROC curves are plotted, and the characteristics about the ROC curves of perfect, random, good and bad classifiers are available in [123, 31, 34]. These are not discussed in this thesis because, as described in the later part of this subsection, ROC is not an appropriate metric for evaluating our tasks at hand.

- **Customer Receiver Operator Characteristics (CROC)** is a variant of ROC. It was proposed specifically for evaluating recommender systems that provide the same number of recommendations (N) to each active user [141]. A CROC curve is plotted by varying the value of N.

Detailed discussions on ROC and CROC in the context of recommender systems are available in [55, 141].

- Rank Score measures the utility of a ranked list of recommendations [14, 63, 82]. The rank score of a user u, denoted as RS_u, is computed as follows:

$$RS_u = \sum_{j=1}^{|I_r|} \frac{\delta(u, i_j)}{2^{(j-1)/(h-1)}} \tag{2.12}$$

where I_r denotes the list of items recommended to the user, j is the index of an item on the recommendation list, and h is the *viewing halflife*, which is the rank of an item on the list such that there is a 50% chance that user u will like that item. $\delta(u, i_j)$ is a number in the range [0, 1] that indicates the contribution of a correct recommendation to the overall utility of the ranked recommendation list [82]. Its value is 0 if i_j is not in user u's testset.

The aggregated rank score (RS) over the set of test users U is then computed as follows:

$$RS = 100 * \frac{\sum_{u \in U} RS_u}{\sum_{u \in U} RS_u^{max}} \tag{2.13}$$

where RS_u^{max} is the maximum achievable RS_u for user u if all of the items (s)he liked in the test set had been at the top of the ranked list. A higher rank score indicates that correct recommendations were ranked higher on the recommendation lists.

Note that we did not adopt all of the above metrics for evaluating Top-N recommendation algorithms. As noted, classification accuracy takes the number of true negatives (TN) into account. TN includes not only items that the users had rated negatively in the test set, but also items that the users *did not observe at all*. In recommender systems, data is normally extremely skewed towards the TN category as in IR systems (Chapter 8 in [99]). Specifically, *most items are unobserved by a particular user in practice* in a given recommender system. Consider the number of book titles a user could have purchased and the total number of titles available on Amazon.com as an example. Taking TN into consideration when evaluating recommender systems might not be meaningful.

We also chose not to adopt ROC and CROC in our studies, partially because they capture the TN of recommendation algorithms. In fact, both ROC

and CROC curves offer the advantage of summarizing the performance of an algorithm across different numbers of recommendations by means of AUC. In recommender systems, however, researchers are more concerned about an algorithm's performance when only a small number of items are recommended to users ([13, 63]), because the goal of a recommender system is to help users avoid information overload. For instance, users would be more interested in the quality of the top 10 items than the quality of the top 100 items on a recommendation list. This is the main reason why our studies analyzed the performance of Top-N recommendation algorithms based on precision and recall at small values of N, but not ROC and CROC.

Herlocker et al. [55] noted that precision and recall are biased and shall not be interpreted as absolute measures. However, they are still useful for comparing different algorithms and are easy to understand. Further, they do not take into consideration the value of TN, which is likely to be extremely large in reality in a recommender system. For these reasons, we adopted precision and recall in our experimental studies on Top-N recommendation algorithms.

Coverage

Coverage is generally defined as the percentage of items for which recommendations or predictions can be provided. There are two ways to compute the coverage of an algorithm as noted in [106]. The first way takes into account the universe of items in a recommender system, and computes coverage as the percentage of items for which an algorithm can make recommendations or predictions. The second way only considers percentage of successful predictions given the withheld user-item pairs in the test set.

We chose the definition of coverage based on the specific purposes of our experimental studies, as described in corresponding experimental setup sections.

2.3 Association Rule Mining (ARM)

Association Rule Mining (ARM) is one of the most well-studied data mining tasks since its introduction in the early 90's [5]. It aims at discovering interesting relationships among a set of items (I) by finding items that frequently appeared together in a transactional database (DB). This section describes the idea of

ARM and three of its variants that are directly related to our work.

2.3.1 Association Rules and Their Interestingness

An association rule is in the form of "$\Phi \rightarrow \Psi$". Φ, where $\Phi \subset I$, is called the *antecedent* or the *body* of the rule, while Ψ, where $(\Psi \subset I) \wedge (\Psi \cap \Phi = \emptyset)$, is called the *consequent* or the *head* of the rule [5].

Agrawal et al. [5, 6] proposed an ARM approach and the well-known *Apriori* algorithm based on the *support-confidence* framework. Their approach decomposes the ARM problem into two sub-problems:

1. Find all combinations of items, known as *large itemsets* or *frequent itemsets*, having *support* values above the predefined minimum (*minSupp*). Note that if a certain itemset Φ is frequent, all subsets of items in Φ are also frequent. This is known as the *downward closure property* of support values.

2. Generate association rules from the frequent itemsets. A rule holds and is considered *interesting* if it satisfies the predefined minimum *confidence* (*minConf*). For example, if the itemset $\{\Phi, \Psi\}$ is frequent, we shall check if the two rules "$\Phi \rightarrow \Psi$" and "$\Psi \rightarrow \Phi$" are interesting.

The remainder of this subsection introduces interestingness measures for evaluating association rules and popular ARM algorithms, followed by brief discussions on applying ARM to CF.

Interestingness measures of association rules

The most widely-adopted interestingness measures of association rules are the aforementioned *support* and *confidence*. The support of a rule "$\Phi \rightarrow \Psi$" is the percentage of transactions in DB containing $\{\Phi \cap \Psi\}$, or simply P($\Phi \cap \Psi$). The confidence of the rule is the percentage of transactions in DB containing Φ that also contain Ψ. It can be written as P($\Phi \cap \Psi$)/P(Φ) or P($\Psi|\Phi$). For example, the association rule "*milk* $\rightarrow$ *butter*" [20%, 50%] indicates that 50% of users who bought *milk* also bought *butter* in the same transaction, and that 20% of all users bought both *milk* and *butter* in the same transaction.

Note that the computation of confidence ignores P(Ψ). Brin et al. [16] pointed out that this is problematic, because the confidence of the rule may still

be high enough to satisfy *minConf* even if the occurrence of Ψ is unrelated to that of Φ (e.g. when $P(\Phi \cap \Psi)/P(\Phi) = P(\Psi)$, meaning that Φ and Ψ are statistically independent). Various alternative interestingness measures have been proposed to address the flaw of confidence values (e.g. [16]). A well-known example of these is the *interest* measure, also known as *lift ratio*. It indicates the correlation between the itemsets in the body and the head of a rule, and is defined as $P(\Phi \cap \Psi)/P(\Phi)P(\Psi)$. Φ and Ψ are positively correlated if the interest value of "$\Phi \rightarrow \Psi$" is above 1. In fact, there exist a large variety of interestingness measures, evaluating association rules from different perspectives. A comparative study and comprehensive survey of various interesting measures are available in [152, 40].

Klemettinen et al. [73] provide a non-statistical definition to interesting association rules. Their work applies *rule templates* to look for interesting rules from a collection of *already discovered* association rules. A rule template defines the structure of interesting rules in a form similar to regular expressions. For instance, the rule template "*Any item*$*\rightarrow$ *butter*" considers rules containing the item *butter* in their consequent part to be interesting. Li et al. [88] later adopted the idea of rule templates to a recommendation task. Their work first discovered a set of association rules based on the support-confidence framework, and then applied rule templates to the discovered rules to look for the rules that are of interest. They experimented their approach with a subset of the EachMovie dataset [101]. Results showed that the adoption of rule templates in their work slightly improved recommendation accuracy.

ARM algorithms

The Apriori algorithm [6] is a classical and the most well-known ARM mining algorithm in the literature. It consists of the two major steps described earlier, namely the generation of frequent itemsets and the generation of interesting association rules from the frequent itemsets.

The most noticeable characteristic of the Apriori algorithm is that it takes a breath-first approach to frequent itemsets generation. It iteratively joins two frequent itemsets containing κ-1 items, known as (κ-1)-itemsets, to generate a *candidate κ-itemset*. It then makes use of the aforementioned downward closure property of support values to prune the candidate κ-itemsets. More specifically, this pruning step removes candidate κ-itemsets containing *infrequent item sub-*

sets, with respect to a given *minSupp* value, from further consideration. The remaining frequent κ-itemsets are used in the next iterations for generating larger candidate itemsets as well as association rules.

The frequent itemsets generation approach of Apriori may become inefficient when the transactional database (DB) contains a large collection of items. Further, the algorithm makes multiple passes over DB, which may be a concern for some applications [175]. Some researchers therefore proposed ARM paradigms that adopt different data structures for efficiently generating frequent itemsets for ARM. A well-known example is the CLOSET algorithm described in [119], which proposes the use of the *frequent pattern tree* (FP-tree) structure for efficient discovery of frequent patterns from large databases. Other examples include the DHP (Direct Hashing and Pruning) algorithm [116], which improves efficiency by approximating the support values of itemsets, and the work of Zaki et al. [175], which only scans DB once and clusters itemsets to obtain *potential maximal frequent itemsets*.

Li and colleagues combined classical ARM with the rough sets theory ([118], as cited in [86, 85, 87]) and rule templates [73] to mine interesting and important association rules. To explain their work, we first introduce important concepts in the rough sets theory following the notations used in [85]. Given is a decision table $\mathcal{T} = \{\mathcal{U}, \mathcal{C}, \mathcal{D}\}$, where $\mathcal{U}$ is a set of records in the table, $\mathcal{C}$ is a set of *condition attributes*, and $\mathcal{D}$ is a set of *decision attributes*. A *reduct* of a decision table is a set of condition attributes that is sufficient to represent the decision attributes. In other words, generating reducts essentially means finding important attributes that can characterize the knowledge in the original data. There can be multiple reducts for a decision table, and the intersection of reducts forms the *core* of the decision table.

Li and Cercone [86] described a knowledge discovery framework that first generates reducts from a decision table $\mathcal{T}$, and then applied the classical ARM technique to find a set of interesting association rules for each resulting reduct. They also proposed a novel Rule Importance Measure (RIM) to evaluate such rules mined from reducts:

$$RIM = \frac{no.\ of\ rule\ sets\ containing\ the\ rule}{no.\ of\ rule\ sets} \tag{2.14}$$

where "rule sets" refer to the sets of association rules generated from the reducts (one rule set per reduct).

In another related study [85], Li and Cercone utilized the Apriori algorithm with the support-confidence framework to mine association rules from a decision table $\mathcal{T}$. $\mathcal{T}$ contains a single decision attribute, and rule templates were used to mine association rules containing the decision attribute in the consequent part of the rules. The resulting association rules were then used to create a new decision table $\mathcal{A}$. The objects represented as rows and the decision attribute in $\mathcal{A}$ come from the original decision table, while each condition attribute in $\mathcal{A}$ represents one association rule. An attribute value $\mathcal{A}_{mn}$ in the new table indicates whether the m^{th} rule is applicable to the n^{th} record in the original table $\mathcal{T}$. Li and Cercone then generated reducts from the new decision table. A reduct therefore represents a set of association rules, called a *reduct rule set*, mined from the original data. Li and Cercone adopted RIM to rank the importance of rules in the reduct rule sets. They reported experimental results on a synthetic dataset and a real dataset to illustrate the intuitions of their work.

Applying ARM to CF

ARM techniques can be applied to CF by finding interesting associations between items (item-based CF), or those between users (user-based CF). Suppose in an online supermarket system, a rule "*milk* $\rightarrow$ *butter*" is interesting, and a certain active user has selected *milk* in his/her "shopping cart". The system can then recommend *butter* to the user with some confidence.

Both the classical Apriori based and the rough sets based ARM paradigms can be applied to CF, or recommender systems in general. Recall that the rough sets theory operates on a decision table $\mathcal{T}$ containing some condition attributes and decision attributes. A reduct of $\mathcal{T}$ contains a set of condition attributes that can represent the decision attributes. For the geriatric dataset used in [85], for example, the corresponding decision table contains patient records as rows, symptoms as condition attributes, and the survival status of patients as the decision attribute. Applying the rough sets based approach to a general item recommender system is practically feasible by treating a given target item as a decision attribute, and other items as condition attributes. We illustrate this with Table 2.2, a sample decision table $\mathcal{T}$. Each row in $\mathcal{T}$ represents a user, while each column represents an item. A value $\mathcal{T}_{mn}$ is 1 if user u_m has rated item i_n in a certain system, and 0 otherwise.

Suppose item i_5 in Table 2.2 is the decision attribute, thus the task at hand is

Table 2.2: A sample decision table.

	i_1	i_2	i_3	i_4	i_5
u_1	1	1	0	0	1
u_2	0	0	0	1	0
u_3	1	1	1	0	1
u_4	1	1	0	0	0

to find the important condition attributes (among items i_1 to i_4) for representing i_5. The same task can be defined for every item in the item space. Note that the rough sets theory has the notion of *data inconsistency*. Consider the items rated (and not rated) by u_1 and u_4 as an example. Specifically, both users have rated items i_1 and i_2, but have not rated items i_3 and i_4. In other words, u_1 and u_4 have exactly the same values for all condition attributes, but their values for the decision attribute i_5 do not agree with each other. The rough sets based approach considers this data to be inconsistent.

Our work reported in Chapters 3 and 4 in this thesis adopted an Apriori-like rule mining process. Our work is also closely related to three variants of the classical ARM task described in the following subsections. They include Adaptive-Support Association Rule Mining (ASARM), Fuzzy Association Rule (FAR) mining, and Cross-level Association Rule (CAR) mining. More background information and in-depth discussions on ARM can be found in [51] (Chapter 6). Hipp et al. [56] also provide a brief survey of ARM algorithms, including several variants of the Apriori algorithm.

2.3.2 Adaptive-Support ARM for CF

The traditional ARM problem mines interesting association rules for all items in I from DB. Lin et al. [89] pointed out that this problem definition of ARM is inefficient for mining rules for collaboration recommendations due to two reasons. Firstly, many rules mined from the database would not be relevant for a given user. Secondly, the support and confidence constraints for mining rules must be specified in advance. Due to the variations in user tastes and the popularity of items in the database, this could either lead to too many or too few rules mined for a particular item. Related to this, rules involving less popular items may be difficult to discover, meaning that it is difficult to recommend less

popular items to users in an ARM-based CF system.

Lin et al. therefore proposed the Adaptive-Support ARM (ASARM) algorithm for CF. ASARM mines rules for one target item (t, where $t \in I$) at a time. This idea is similar to that of applying rule templates to find interesting association rules in [73]. Instead of applying rule templates to a collection of already discovered rules, however, the ASARM algorithm makes use of a rule template in the form of "$\Phi \rightarrow t$" in the rule mining process.

ASARM allows users to specify a desired range [*minNumRules, maxNumRules*] for the number of rules to be mined, and automatically adjusts the value of *minSupp* so that the number of resulting rules is in the given range, unless fewer than *minNumRules* rules exist. More specifically, it increases (resp. decreases) the value of *minSupp* when there exist too many (resp. few) rules that satisfied the user-specified interestingness constraints. The rule mining process of ASARM makes multiple passes over the data as it mines rules for one single target item at a time. This process, however, is done offline and therefore does not affect the response time of the recommendation process.

2.3.3 Fuzzy Association Rule (FAR) Mining

ARM was designed for mining rules from categorical attributes. Applying it to quantitative attributes, such as numerical ratings users have given items in the CF scenario, requires an additional data discretization step. Such discretization is usually boolean. For instance, given the rating scale of -10 to 10 in the Jester recommender [43] and a discretization threshold of 5, one may transform ratings below 5 into *Dislike* or 0, and those equal to or greater than 5 into *Like* or 1. Traditional ARM techniques can then be applied to the transformed data. This boolean discretization process, however, suffers from the *sharp boundary problem*, in which a very small difference between two values can cause them fall into two totally different classes. In the previous example, a rating of 4.9 will be transformed into *Dislike*, whereas that of 5 will be transformed into *Like*.

One solution to address the sharp boundary problem, in the context of ARM, is the use of Fuzzy Association Rule (FAR) mining. FAR mining is an extension of ARM that mines rules from quantitative attributes. A FAR is in the form of "$\langle \Phi, \mathcal{X} \rangle \rightarrow \langle \Psi, \mathcal{Y} \rangle$", where $\mathcal{X}$ and $\mathcal{Y}$ are fuzzy sets that characterize attributes Φ and Ψ respectively. The main feature of FAR mining is that each attribute can be

a member of a fuzzy set to a certain degree in [0,1], assigned by the membership function (MF) associated with the corresponding fuzzy set. Figure 2.2 shows three simple examples of fuzzy sets and MFs. In each example, numerical ratings are fuzzified into three fuzzy sets, *Like*, *Neutral* and *Dislike*, respectively denoted by *L*, *N* and *D* in the figure and in the subsequent discussions.

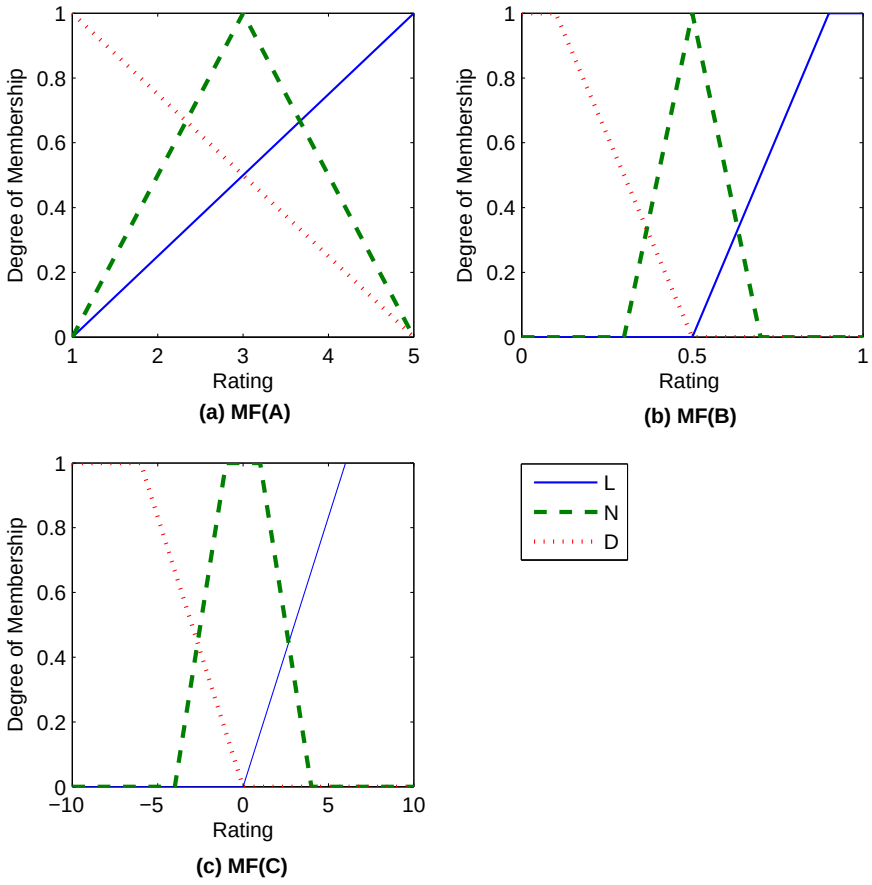

Figure 2.2: Sample fuzzy sets and membership functions.

We use an example to illustrate the use of fuzzy sets for transforming user-specified ratings. Table 2.3 shows a sample $U \times I$ ratings matrix (R), in which rows and columns represent users and items respectively. Given the sample ratings in R and MF(A) in Figure 2.2(a), Table 2.4 shows the transformed ratings of the two items i_1 and i_2. Each column, which was the identifier (ID) of an item, is expanded into a $\langle$*Item ID, Fuzzy Set*$\rangle$ pair, and each element in R, which was the user rating of a particular item, is transformed into its membership degree with respect to the specified fuzzy set. Note that the term "attribute" refers to a

Table 2.3: A sample user-item ($U \times I$) ratings matrix in CF.

	i_1	i_2	i_3	i_4	i_5
u_1	1	4			2
u_2				5	
u_3	5	2	1		5
u_4	5	1			

Table 2.4: Fuzzified ratings.

	$\langle i_1, L\rangle$	$\langle i_1, N\rangle$	$\langle i_1, D\rangle$	$\langle i_2, L\rangle$	$\langle i_2, N\rangle$	$\langle i_2, D\rangle$
u_1	0	0	1	0.8	0.5	0.2
u_2	0	0	0	0	0	0
u_3	1	0	0	0.2	0.5	0.8
u_4	1	0	0	0	0	1

⟨*Item ID, Fuzzy Set*⟩ pair in this section.

Some items have a larger total fuzzified rating than the others in Table 2.4. An example is the item i_2 in the first transaction with UserID u_1. The fuzzified ratings of $\langle i_2, L\rangle$, $\langle i_2, N\rangle$ and $\langle i_2, D\rangle$ are respectively 0.8, 0.5 and 0.2. In other words, the item i_2 in the transaction of u_1 would contribute more than 1, which is the actual support count it received from u_1, to the interestingness measures of itemsets containing it. To avoid this, fuzzified ratings are normalized so that their contributions with respect to all fuzzy sets for a given item sum to 1 if rated [48]. The normalization is done as follows:

$$m_{x_j}(DB_n[\phi_j]) = \frac{m\prime_{x_j}(DB_n[\phi_j])}{\sum_{x\in\mathcal{X}} m\prime_x(DB_n[\phi_j])} \tag{2.15}$$

where $\phi_j \in \Phi$ is an attribute, and $\mathcal{X}$ is the fuzzy set that characterizes Φ as noted. $DB_n[\phi_j]$ represents the value of the attribute ϕ_j in the i^{th} record in DB. $m_{x_j}(DB_n[\phi_j])$ and $m\prime_{x_j}(DB_n[\phi_j])$ are respectively the normalized and original membership degrees obtained by the value $DB_n[\phi_j]$ with respect to x_j. According to this equation, the fuzzified ratings in Table 2.4 are normalized to 0.8/1.5, 0.5/1.5 and 0.2/1.5 (i.e. 0.53, 0.33 and 0.14) as Table 2.5 shows.

Similar to classical association rules, there exist various measures for indicating the interestingness of FARs. The three most commonly used measures

Table 2.5: Fuzzified ratings with normalization.

	$\langle i_1, L\rangle$	$\langle i_1, N\rangle$	$\langle i_1, D\rangle$	$\langle i_2, L\rangle$	$\langle i_2, N\rangle$	$\langle i_2, D\rangle$
u_1	0	0	1	0.53	0.33	0.14
u_2	0	0	0	0	0	0
u_3	1	0	0	0.14	0.33	0.53
u_4	1	0	0	0	0	1

are fuzzy support, fuzzy confidence and correlation [49]. We illustrate the computations of these measures with Example 2.1 and the ratings in Table 2.5 in the rest of this subsection.

Example 2.1 *Let* $\langle \Phi, \mathcal{X}\rangle$ = $\langle i_1, L\rangle$, $\langle \Psi, \mathcal{Y}\rangle$ = $\langle i_2, D\rangle$. *Also, let* $\Gamma = \Phi \cup \Psi$, $\mathcal{Z} = \mathcal{X} \cup \mathcal{Y}$, *and therefore* $\langle \Gamma, \mathcal{Z}\rangle$ = $\langle \{i_1, i_2\}, \{L, D\}\rangle$.

We now describe the computations of fuzzy support. A fuzzy support value reflects not only the number of transactions supporting an itemset but also their degree of support. The fuzzy support value of an itemset $\langle \Phi, \mathcal{X}\rangle$, denoted by $FS_{\langle \Phi, \mathcal{X}\rangle}$, is defined as [49]:

$$FS_{\langle \Phi, \mathcal{X}\rangle} = \frac{\sum_{n=1}^{|DB|} \prod_{\phi_j \in \Phi} \{m_{x_j}(DB_n[\phi_j])\}}{|DB|} \tag{2.16}$$

where $m_{x_j}(DB_n[\phi_j])$ is computed using Eq. (2.15), and $|DB|$ denotes the number of records in the transactional database DB.

Example 2.2 *Computation of fuzzy support value:* $FS_{\langle \Phi, \mathcal{X}\rangle}$ *= (0 + 0 + 1 + 1) / 4 = 0.5,* $FS_{\langle \Psi, \mathcal{Y}\rangle}$ *= (0.14 + 0 + 0.53 + 1) / 4 = 0.4175, and* $FS_{\langle \Gamma, \mathcal{Z}\rangle}$ *= (0 * 0.14) + (0 * 0) + (1 * 0.53) + (1 * 1) / 4 = 0.3825.*

The fuzzy confidence of a rule "$\langle \Phi, \mathcal{X}\rangle \rightarrow \langle \Psi, \mathcal{Y}\rangle$", denoted by $FC_{\langle \langle \Phi, \mathcal{X}\rangle \rightarrow \langle \Psi, \mathcal{Y}\rangle\rangle}$, is computed as [49]:

$$\begin{aligned} FC_{\langle \langle \Phi, \mathcal{X}\rangle \rightarrow \langle \Psi, \mathcal{Y}\rangle\rangle} &= \frac{FS_{\langle \Gamma, \mathcal{Z}\rangle}}{FS_{\langle \Phi, \mathcal{X}\rangle}} \\ &= \frac{\frac{\sum_{n=1}^{|DB|} \prod_{\gamma_j \in \Gamma} \{m_{z_j}(DB_n[\gamma_j])\}}{|DB|}}{\frac{\sum_{n=1}^{|DB|} \prod_{\phi_j \in \Phi} \{m_{x_j}(DB_n[\phi_j])\}}{|DB|}} \\ &= \frac{\sum_{n=1}^{|DB|} \prod_{\gamma_j \in \Gamma} \{m_{z_j}(DB_n[\gamma_j])\}}{\sum_{n=1}^{|DB|} \prod_{\phi_j \in \Phi} \{m_{x_j}(DB_n[\phi_j])\}} \end{aligned} \tag{2.17}$$

where $FS_{\langle \Phi, \mathcal{X} \rangle}$ and $FS_{\langle \Gamma, \mathcal{Z} \rangle}$ are computed using Eq. (2.16).

Example 2.3 *Computation of fuzzy confidence value:* $FC_{\langle \langle \Phi, \mathcal{X} \rangle \rightarrow \langle \Psi, \mathcal{Y} \rangle \rangle}$ *= (0.3825 / 0.5) = 0.765.*

The correlation between $\langle \Phi, \mathcal{X} \rangle$ and $\langle \Psi, \mathcal{Y} \rangle$, denoted by $CORR_{\langle \langle \Phi, \mathcal{X} \rangle, \langle \Psi, \mathcal{Y} \rangle \rangle}$, is defined as follows [49]:

$$CORR_{\langle \langle \Phi, \mathcal{X} \rangle, \langle \Psi, \mathcal{Y} \rangle \rangle} = \frac{Cov_{\langle \langle \Phi, \mathcal{X} \rangle, \langle \Psi, \mathcal{Y} \rangle \rangle}}{\sqrt{Var_{\langle \Phi, \mathcal{X} \rangle} * Var_{\langle \Psi, \mathcal{Y} \rangle}}} \tag{2.18}$$

where

$$Cov_{\langle \langle \Phi, \mathcal{X} \rangle, \langle \Psi, \mathcal{Y} \rangle \rangle} = FS_{\langle \Gamma, \mathcal{Z} \rangle} - FS_{\langle \Phi, \mathcal{X} \rangle} * FS_{\langle \Psi, \mathcal{Y} \rangle} \tag{2.19}$$

$$Var_{\langle \Phi, \mathcal{X} \rangle} = FS_{\langle \Phi, \mathcal{X} \rangle^2} - (FS_{\langle \Phi, \mathcal{X} \rangle})^2 \tag{2.20}$$

$$FS_{\langle \Phi, \mathcal{X} \rangle^2} = \frac{\sum_{n=1}^{|DB|} (\prod_{\phi_j \in \Phi} m_{x_j}(DB_n[\phi_j]))^2}{|DB|} \tag{2.21}$$

and similar for $\langle \Psi, \mathcal{Y} \rangle$.

The definitions in Eqs. (2.18)-(2.21) are extensions of the basic formulas of variance and covariance in statistics [49]. The value of correlation ranges from -1 to 1. Only a positive value tells that the body and the head of a rule are positively correlated. The closer the value is to 1, the more correlated they are.

Example 2.4 *Computation of correlation:* $Cov_{\langle \langle \Phi, \mathcal{X} \rangle, \langle \Psi, \mathcal{Y} \rangle \rangle}$ *= 0.3825 - (0.5 * 0.4175) = 0.17375,* $FS_{\langle \Phi, \mathcal{X} \rangle^2}$ *= (*0^2 + 0^2 + 1^2 + 1^2*) / 4 = 0.5,* $(FS_{\langle \Phi, \mathcal{X} \rangle})^2 = (0.5)^2$ *= 0.25,* $Var_{\langle \Phi, \mathcal{X} \rangle}$ *= (0.5 - 0.25) = 0.25,* $FS_{\langle \Psi, \mathcal{Y} \rangle^2}$ *= (*0.14^2 + 0^2 + 0.53^2 + 1^2*) / 4* $\approx$ *0.3251,* $(FS_{\langle \Psi, \mathcal{Y} \rangle})^2 = (0.4175)^2 \approx$ *0.1743,* $Var_{\langle \Psi, \mathcal{Y} \rangle}$ *= (0.3251 - 0.1743) = 0.1508, and therefore* $CORR_{\langle \langle \Phi, \mathcal{X} \rangle, \langle \Psi, \mathcal{Y} \rangle \rangle} = \frac{0.17375}{\sqrt{0.25 * 0.1508}} \approx 0.895$.

We adopted the three commonly-used measures described above as indicators of interestingness in Chapters 3 and 4.

2.3.4 Cross-level Association Rule (CAR) Mining

Cross-level Association Rule (CAR) mining operates on DB and a concept hierarchy of the items in DB. It aims at discovering associations among the concepts from different levels of the hierarchy. Several studies, such as [71, 21, 111], reported the use of *is-a hierarchies* or *item taxonomy* as concept hierarchies for capturing similarities between items.

Higher-level items in a is-a hierarchy represent more general concepts, and are likely to have more support than lower-level items [150, 50]. This property of is-a hierarchies has two implications for mining. The first implication is the need to apply different *minSupp* values to items at different levels, and a typical case of which is to progressively reduce *minSupp* at lower levels of abstraction. The second implication is that descendents of infrequent items can be pruned. Detailed discussions on the two issues are available in [50].

2.4 Text Data Mining Basics

This section briefly introduces the essential processes and tasks in text data mining, so as to provide background information about our discussions on sentiment analysis research in the next section.

Text data mining, or simply text mining, is the application of data mining techniques to automated knowledge discovery from unstructured or semi-structured electronic text documents [32, 53, 51, 126, 30]. Text mining has received a considerable amount of research attention due to the vast, yet increasing, amount of textual information available on the Web. It also has a business value in supporting business intelligence because, in reality, unstructured data in the form of text documents typically account for 85% of an organization's knowledge store [134, 39, 77].

Text documents are usually in the form of natural language text. Examples include e-mails, HTML pages and online newspaper articles, as well as Web 2.0 contents including user-generated reviews and blog (weblog) posts. As traditional data mining techniques are designed to handle structured data from databases, a text mining application requires a text analysis process to transform text documents into structured data, so that standard data mining techniques can be applied to them. Figure 2.3 depicts the major steps in a typical text mining

application.

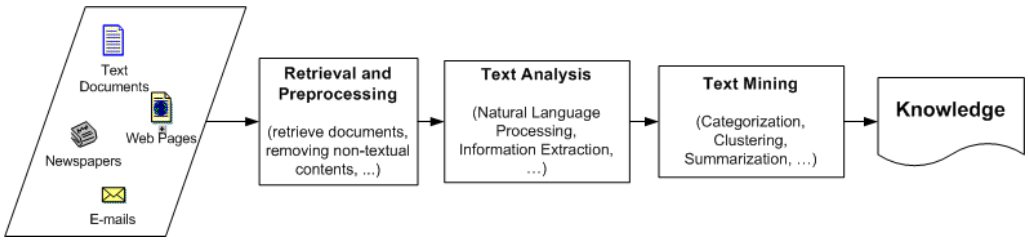

Figure 2.3: Major steps in the typical text mining process.

As shown in Figure 2.3, the first step in a text mining system is to retrieve from a document collection the relevant documents to be analyzed [30]. Examples of sources of document collections include the World Wide Web (e.g. [110]) and internal file systems as well as application data (e.g. [170]). Some documents, such as HTML pages, can contain non-textual contents including multimedia objects, HTML tags, and so on. Our work assumes that a collection of relevant documents is given, and that the non-textual contents of the documents are removed. Thus, our discussions focus on the processing and mining of the textual portions of those documents only.

The second step in a text mining system is a text analysis process that preprocesses the documents and extracts the relevant pieces of data from them. This step usually utilizes various techniques borrowed from other related disciplines, such as NLP and IE, which is a subfield of NLP [47, 64]. The processed text documents are then transformed into structured models, such as the vector space model that is commonly used in IR algorithms (pages 428–435 in [51]).

The third step in a text mining system involves the use of different data mining techniques to perform the various tasks that the system is about to solve, and produces high-level knowledge that is required by the users of the system. Common tasks of text mining include text classification, also known as text categorization [143], clustering [98] and summarization [124].

In the rest of this section, we introduce commonly used tasks for analyzing text documents, followed by a description of data representation of text documents. We then describe several weighting schemes for reflecting the importance of features. Finally, we introduce text classification, a text mining task that is relevant to our work on sentiment analysis and review-based recommendations.

2.4.1 Analyzing Text Documents

As aforementioned, the text analysis process in a text mining system preprocesses text documents and extracts the relevant data from them for mining. This process usually relies on various techniques borrowed from other related disciplines, including NLP and IE, for analyzing natural language texts. The output produced by this process is a structured representation of useful data in the original text documents. As the focus of this thesis is not on NLP and IE, we only describe two tasks that have been adopted in our work. They include stemming and part-of-speech (POS) tagging.

Stemming identifies and reduces syntactic variants of the same word into their root form [51, 64, 164]. In the stemming process, for example, "do", "does", "doing", "done" and "did" would be reduced to the stem "do". This type of stemming is inflectional. It analyzes variations in word forms that express grammatical features, such as singular and plural, or past and present tenses. Another type of stemming is derivational. It analyzes words that are created from other words. Derivation usually involves changing the grammatical category of a word and may even modify its meaning. For example, the word "disallow" comes from the word "allow" but has the opposite meaning.

Stemming results in a smaller number of distinct words in a document while increasing the frequency counts of some words. In the previous paragraph, for example, the underlined words "reduces" and "reduced", both occurred once, belong to the same stem "reduce", which will have a total frequency count of 2 after stemming. Stemming can make a difference for algorithms that take word frequency counts into account [164].

Part-of-speech (POS) tagging assigns POS tags to tokens in a document. There are two major types of POS taggers: rule-based and stochastic. Rule-based taggers apply linguistic knowledge to assign POS tags to unknown or ambiguous words, whereas stochastic taggers rely on training corpora to determine the probability that a word occurs with a particular tag [64]. POS tagging is the basis for extracting useful information from documents in various IE and text mining tasks. For example, Named Entity Recognition (NER) is a typical IE task that aims at locating and classifying proper names in texts into named entities classes (e.g. people and places) [12]. Another example is sentiment analysis, which involves extracting user opinions, which are usually verbs or adjectives

[156, 172, 61], from documents.

2.4.2 Representing Text Documents

The previous subsection described two tasks that help identifying in text documents their major features. These features are then represented in a structured way for mining. The vector space model is a widely adopted model for representing text documents [8]. In this model, a document $d_j \in D$, where D is a document collection, is represented as a n-dimensional vector $\overrightarrow{d_j}$. Each dimension of the vector corresponds to a *term*, or a *feature*, whose definition may vary depending on the application. If a term is a single word appeared in a text document, then the dimension of $\overrightarrow{d_j}$ would be the total number of distinct terms in the document collection D. Such vectorial representation of documents is also known as the *bag-of-words model*, which represents a document as an *unordered* bag of terms.

Each value $w_{i,j}$ in the vector $\overrightarrow{d_j}$ corresponds to the *weight* of the term f_i with respect to d_j:

$$\overrightarrow{d_j} = [w_{1,j}, w_{2,j}, ..., w_{n,j}]^T$$

The value of $w_{i,j}$ would be 0 if f_i did not appear in d_j. Otherwise, it can be determined using a variety of term weighting schemes, or feature weighting schemes, to reflect the importance of f_i in d_j. We defer the description of feature weighting schemes to the next subsection.

If the set of terms in a document collection is very large, meaning that the dimensionality of term vectors is very high, operations on the terms vectors may become very expensive and inefficient. Dimensionality reduction, by means of feature selection for example, may be necessary to prune less important terms [8, 51].

2.4.3 Determining Term Weights

Term weights are not only useful for reflecting the importance of terms in a document or a document collection, but also for facilitating feature selection. Feature selection aims at picking only features that are considered important and relevant in a given application, which is *text classification* in our subsequent

discussions. For instance, we may define a minimum threshold on feature weights, and discard features whose weights do not meet the threshold from further consideration. We describe, among others, three feature weighting schemes in this subsection. They include Term Frequency-Inverse Document Frequency (TF-IDF), Information Gain (IG) and Chi-square test (χ^2). Note that we used the words "term" and "feature", as well as "class" and "category" interchangeably.

Term Frequency-Inverse Document Frequency (TF-IDF) is the most well-known term weighting scheme in IR. It measures the importance of a given term f_i, both "locally" with respect to a given document $d_j \in D$, and "globally" in the document collection D (pages 29–30 in [8], Chapter 6 in [99]).

The **Term Frequency** of f_i in d_j, denoted by $tf(f_i, d_j)$, indicates the frequency of f_i in d_j. tf_i is often normalized by the total frequency counts of all terms in d_j. The purpose of doing so is to prevent a bias towards longer documents, in which terms may have high frequency counts regardless of their importance in the documents. Formally, $tf(f_i, d_j)$ is defined as:

$$tf(f_i, d_j) = \frac{N(f_i, d_j)}{\sum_{k=1}^{|F_j|} N(f_k, d_j)} \tag{2.22}$$

where F_j denotes the set of features in d_j, and $N(f_i, d_j)$ returns the number of times f_i appeared in d_j.

The **Inverse Document Frequency** of f_i, denoted by $idf(f_i)$, is defined as:

$$idf(f_i) = log \frac{|D|}{|\{d_j | \exists d_j \in D : N(f_i, d_j) > 0\}|} \tag{2.23}$$

where $|D|$ denotes the number of documents in the document collection D, and $|\{d_j | \exists d_j \in D : N(f_i, d_j) > 0\}|$ represents the number of documents in which f_i appeared. Logarithms may be to base 10 or base 2, but this does not matter if one's purpose is to rank terms rather then measuring the absolute weights of terms (Chapter 6 in [99]). The IDF of a f_i is high if it is a rare term, whereas that of f_i is likely to be very low if it is commonly seen in a large number of documents.

The TF-IDF of f_i with respect to d_j is then computed as follows:

$$tf\text{-}idf(f_i, d_j) = tf(f_i, d_j) \cdot idf(f_i) \tag{2.24}$$

The term f_i will receive a high weight under the TF-IDF scheme if it has a high frequency count in d_i, but a low document frequency in D.

Information Gain (IG) is an information based weighting scheme that measures the number of bits of information obtained for category (class label) prediction by knowing the presence or absence of a term in a document [171, 142]. It has been found to perform well in various studies on text classification [171, 142, 37]. Given a set of terms and a set of classes C, the IG of term f_i with respect to a certain class $c_j \in C$, denoted by $IG(f_i, c_j)$, is defined as [142]:

$$IG(f_i, c_j) = \sum_{c \in \{c_j, \overline{c}_j\}} \sum_{f \in \{f_i, \overline{f}_i\}} P(f, c) \cdot log \frac{P(f, c)}{P(t)P(c)} \tag{2.25}$$

where $P(f_i, c_j)$ represents the probability that the term f_i appears in a certain document in the class c_j, $P(f_i)$ and $P(c_j)$ represent the probabilities of appearance of the term f_i and the class c_j respectively. $P(\overline{f}_i, c_j)$ is the probability that f_i *does not* appear in a document in the class c_j, and similar for $P(f_i, \overline{c}_j)$, $P(\overline{f}_i)$ and $P(\overline{c}_j)$. These are all prior probabilities calculated from training data.

Note that $IG(f_i, c_j)$ is "locally" specified with respect to a single class c_j. One possible way to estimate $IG(f_i)$ as a global measure is to compute the sum of $IG(f_i, c_j)$ over all possible classes [142]:

$$IG(f_i) = IG_{sum}(f_i) = \sum_{j=1}^{|C|} IG(f_i, c_j) \tag{2.26}$$

Other possible methods for determining $IG(f_i)$ include taking the weighted sum $IG_{wsum}(f_i) = \sum_{j=1}^{|C|} P(c_j) IG(f_i, c_j)$, or the maximum $IG_{max}(f_i) = \max_{j=1}^{|C|} IG(f_i, c_j)$ of the set of class-specified IG values of f_i [142].

Chi-square test (χ^2), also **chi-squared test**, is a statistical test that checks whether a null hypothesis (H_0) follows the χ^2 distribution. In the context of text classification, the H_0 is that the occurrence of a term f_i is independent of the class c_j of the document in which f_i occurs. f_i will have a low $\chi^2(f_i, c_j)$ value if its occurrence is independent of c_j. Therefore, only terms that have high χ^2 weights are useful for classifying texts. $\chi^2(f_i, c_j)$ is defined as [142]:

$$\chi^2(f_i, c_j) = |D| \cdot \frac{\left(P(f_i, c_j)P(\overline{f}_i, \overline{c}_j) - P(f_i, \overline{c}_j)P(\overline{f}_i, c_j)\right)^2}{P(f_i)P(\overline{f}_i)P(c_j)P(\overline{c}_j)} \tag{2.27}$$

The above computation of χ^2 statistic of f_i is specific to a given class c_j. Similar to $IG(f_i)$, it is possible to compute $\chi^2(f_i)$ based on χ^2_{sum}, χ^2_{wsum} or χ^2_{max}.

2.4.4 The Text Classification Task

After representing text documents in a structured way, mining knowledge from text data can actually be performed using traditional data mining and machine learning techniques, such as those for classification and clustering. We introduce, among others, a specific task of text mining, namely text classification.

Text classification is also known as text categorization. It is a *supervised learning* technique that involves identifying the main themes of documents and classifying them into a predefined set of categories with different class labels [143, 30]. Some classifiers assign a single class label to a document, while others may assign multiple class labels to it, a task known as multi-class or multi-label classification, with the associated probabilities.

Text classification has a large variety of applications. For instances, it can be used to classify documents based on their subject categories (topics) (e.g. [60]), authorship (e.g. [26]), or genres such as *Editorial*, *Advertisement*, and *Review* [27, 143, 28]. It can also be used for spam detection, with the class labels being *Spam* and *Not Spam* [45]. We call these *topic-based classification* in general. Sentiment analysis, sometimes referred to as sentiment classification, is another application of text classification that deals with the *polarity* or the *sentimental orientation*, referred to as SO hereafter, of the opinions contained in texts. Class labels in such application represent sentiment classes, such as *Positive* or *Negative* [115, 25].

There exist a variety of methods for performing text classification. The multinomial **Naive Bayes (NB)**, or Naive Bayesian, model [100], among others, is a simple model that has been widely used for various text classification tasks. Given a class c_j and a feature $f_i \in F$, the NB classifier estimates the conditional probability that f_i appeared in a document in class $c_j \in C$, denoted by $\mathrm{P}(f_i|c_j)$, from training data. $\mathrm{P}(f_i|c_j)$ is often computed with add-one (Laplace) smoothing to prevent zero probabilities [74]. Formally, $\mathrm{P}(f_i|c_j)$ is defined as:

$$\mathrm{P}(f_i|c_j) = \frac{N(f_i, c_j) + 1}{\sum_{k=1}^{|F|} N(f_k, c_j) + |F|} \tag{2.28}$$

where $N(f_i, c_j)$ is the number of times f_i appeared in c_j, and $|F|$ denotes the size of feature set in the training data.

The NB classifier then estimates the probability that a text document d belongs to a class c_j, denoted by $\mathrm{P}(d|c_j)$, based on the features d contains. Note that NB assumes that the occurrence of each feature f_i is independent of the occurrences of other features in d. It computes $\mathrm{P}(d|c_j)$ as follows:

$$\mathrm{P}(d|c_j) = \mathrm{P}(c_j) \prod_{f_i \in F_d} \mathrm{P}(f_i|c_j) \tag{2.29}$$

where F_d is the set of features appeared in the document d. $\mathrm{P}(c_j)$ is the prior probability of c_j in the training data. The NB classifier attempts to assign the best class label to d after estimating $\forall c_j \in C : \mathrm{P}(d|c_j)$. In other words, it classifies d based on the maximum a posteriori class $c_{map}(d)$:

$$c_{map}(d) = \arg\max_{c_j \in C} \mathrm{P}(d|c_j) \tag{2.30}$$

The NB classifier has been successfully applied to various classification problems, including those that are related to the themes of this thesis, namely collaborative recommendations [110, 141] and sentiment classification [115, 25]. It has been found to perform generally well despite its simplicity.

2.5 Sentiment Analysis of User-Generated Reviews

We now look into *sentiment analysis*, a specific application of text classification that classifies texts based on the *polarity* or the SO (sentimental orientation) of the opinions they contain.

Sentiment analysis is sometimes referred to as opinion mining, opinion extraction, and affect analysis in the literature. Further, the terms **sentiment analysis** and **sentiment classification** have been used interchangeably. It is, however, necessary to distinguish between these two subtly different concepts. In this thesis, hence, sentiment analysis refers to the complete process of analyzing, extracting and understanding the sentiments being expressed in text documents, whereas sentiment classification is the task of assigning class labels to the documents, or segments of the documents, to indicate their polarity.

In what follows, we first describe what sentiment analysis is and how it is different from traditional text classification. We then discuss how precedent

studies addressed the essential tasks in sentiment analysis.

2.5.1 Sentiment Analysis

Sentiment analysis has been viewed as an application of text classification, but in fact, it falls at the crossroads of a few disciplines. They include computational linguistics, information extraction, machine learning and text mining. Sentiment analysis research therefore covers a broad range of topics. Examples include the detection of subjectivity in texts, a research topic in computational linguistics known as *subjectivity analysis* (e.g. [17, 166, 165]), the classification of the polarity of opinion-bearing terms, sentences and/or documents (e.g. [52, 156, 157, 72, 61]), as well as extraction of opinions, product features, and the relationships between them from reviews (e.g. [172, 133, 93, 120, 65]).

From a text classification perspective, sentiment analysis is the classification of texts into *sentiment classes* that reflect the SO of the opinions contained in the texts as noted. Most existing sentiment analysis algorithms perform classification based on bipolar classes, such as *Positive* or *Negative* [156, 115, 25]. More recent work extends binary sentiment classification to classify texts with respect to multi-point rating scales, such as 1 to 5 "stars" or "points", to indicate the *overall* sentiments expressed in texts. This task is known as *rating inference* in the literature [114, 41, 83, 177]. While rating inference aims at determining one overall rating for a particular review, some researchers try to identify the different aspects of a product being mentioned in a review, and infer one rating for each of the identified aspects [149, 176, 145].

Sentiment analysis has been applied to various text genres, including open answers in questionnaires [170], newsgroup articles [24], user-generated reviews (e.g. [156, 115, 25, 61, 114]) and weblog posts (e.g. [20, 108]). It can be applied at various levels as well, including word or phrase level, sentence level, or document level [79]. Our discussions focus on studies that perform document level sentiment analysis of user-generated reviews, which have been referred to as product reviews or user reviews in previous work.

Sentiment analysis versus topic-based text classification

Sentiment analysis is different from topic-based classification in four aspects. Firstly, some sentiment analysis algorithms are only interested in the following

two types of features: evaluative *opinion phrases* carrying users' sentiments, and *item features*[6], which are terms that describe the attributes or characteristics of the subject matters being reviewed (generally referred to as *items* regardless of the domain concerned) [62, 84]. Sentiment analysis may therefore require an additional feature extraction step in order to identify these from reviews.

Secondly, sentiment analysis involves the determination of *opinion strength* [114, 81]. This is different from the determination of feature weights in topic-based classification. For instance, both opinion words "brilliant" and "good" carry positive sentiments, but the preference implied by "brilliant" is "more positive" (i.e. stronger). Consider another pair of opinion words, "best" and "worst". Apparently, they have the same strength but represent opposite polarity.

Thirdly, class labels in sentiment analysis are sentiment classes that encode a specific *order*, and pairwise preference between a given pair of class labels may exist [112, 114, 177]. In the rating inference task, for instance, a review rated as "5 stars" is obviously is more positive than a review rated as "1 star". This is a special property of sentiment analysis that does not exist in topic-based classification.

Lastly, while text classification assigns class labels to documents, the outputs of sentiment analysis algorithms can be of different forms. Some algorithms aim at producing the classifications of terms, sentences or documents based on their polarity, while others produce a summarization of the opinions expressed in documents (e.g. [61, 91, 38]). There are also studies that focus on building lexicons to facilitate sentiment analysis, such as those reported in [29, 67].

Key tasks in sentiment analysis

We use an example to illustrate the key tasks in a typical sentiment analysis or rating inference process. The paragraph below is extracted from a movie review on IMDb.

> "*This movie is ..... quite boring ..... I just felt that they had nothing to tell ..... However, the movie is not all bad ..... the acting is brilliant, especially Massimo Troisi.*"

[6] In this thesis, "item features" refers to the attributes or characteristics describing the subject matters being reviewed, while "features" generally refers to terms, including opinions and item features, that are extracted from the reviews.

In this example, the user stated that the movie being reviewed is "quite boring" but "the acting is brilliant". Suppose we are addressing the rating inference task, which aims at determining the overall sentiment implied by the user given these positive and negative opinions, and mapping such sentiment onto a fine-grained rating scale. This involves three key tasks. Firstly, interesting information, especially opinion words, must be extracted from the review, which is an unstructured, natural language text document. Secondly, the SO of the expressed opinions must be identified. The strength of the SO of the opinions should also be determined. For instance, both "brilliant" and "good" represent positive sentiments, but the preference implied by "brilliant" is obviously much stronger. Finally, a rating is inferred from a review to represent the overall SO implied by the user (the user-specified rating of the above review was 5/10).

A complete review of the sentiment analysis literature is beyond the scope of this thesis due to the very broad coverage of research topics in sentiment analysis and its multidisciplinary nature. We focus on discussing how precedent studies addressed the aforementioned key tasks in sentiment analysis in the next subsections.

2.5.2 Extracting Interesting Features

This task identifies and extracts interesting features from reviews for the subsequent analysis as in standard topic-based classification. Interesting features in sentiment analysis mainly include opinions and item features as noted. In existing sentiment analysis algorithms, identifying opinions is commonly done and can be automated with the help of POS information. For examples, Turney [156] defined a set of POS patterns containing adverbs and adjectives for extracting opinion phrases, while Hu and Liu's work [61, 62] considered adjectives to be opinions. This is supported by subjectivity analysis research which suggests that adjectives usually have significant correlation with subjectivity [17, 165]. Some studies utilized human effort to aid opinion extraction. Das and Chen [24], for example, manually defined a lexicon of opinion-bearing terms for financial news, while Pang et al. [115] made use of human introspection with corpus statistics to decide terms that may appear in user reviews.

Item features can also be identified based on POS tags, due to the observation that item features are generally nouns or noun phrases [172, 83]. While Hu and

Liu [62] also made use of POS tags for this task, they adapted the idea of frequent itemsets discovery in ARM [5] to item feature extraction. They considered each frequent itemset, which is a set of nouns or noun phrases that appeared together frequently in a sentence, to be an item feature. They also proposed methods for pruning redundant item features and for identifying infrequent item features. Studies with an emphasis on the linguistic perspective of reviews usually employ more sophisticated extraction techniques. Opinion Observer [91], for example, applies a language pattern mining technique to the item feature extraction task. Popescu and Etzioni [120] described the OPINE system that makes use of Web statistics and linguistic rules for finding explicit item features, including the properties and parts, of a given product class.

Apart from studies that explicitly look for opinions and item features in reviews, there are also studies that adopted the simple bag-of-words model for representing reviews (e.g. [114, 112, 149]). Those studies treated all words or phrases in reviews simply as features, and did not distinguish between opinion-bearing terms, product features or other terms that do not express opinions.

2.5.3 Determining SO and Strength of Opinions

Most existing studies addressed the task of SO and/or opinion strength determination as a binary classification problem, in which opinions are classified into two classes, such as *Positive* and *Negative*. We describe and comment on several well-known algorithms for accomplishing this task in the following paragraphs.

Hatzivassiloglou and McKeown [52] proposed a supervised learning algorithm for classifying a set of adjectives into two groups. Their algorithm makes use of conjunctions, such as "and" and "but", and morphological relationships between words, such as "adequate" versus "inadequate", to determine whether a pair of adjectives has the same or has opposite SO. After partitioning the adjectives into two groups, their algorithm computes the average frequency of the adjectives in each group. Based on the observation that positive adjectives are used more frequently than negative adjectives [52], their algorithm labels the group of adjectives with higher average frequency as *Positive*, and the other group as *Negative*. Note that their algorithm cannot determine the strength of an adjective with respect to its estimated SO.

Turney [156] proposed the PMI-IR algorithm for the SO determination task.

PMI-IR is an unsupervised algorithm utilizing Pointwise Mutual Information (PMI) between opinion phrases and IR (more specifically, a search engine). It computes the SO of a phrase as the PMI between the phrase and the words "excellent" and "poor". Such information is determined based on statistics gathered by a general-purpose search engine. Turney considered that a phrase is likely to be positive (resp. negative) if it is strongly associated with the word "excellent" (resp. "poor") [156]. Taboada et al. [151] recently experimented with the PMI-IR algorithm using Google as the search engine for computing SO. They noted that the SO of an opinion word predicted at different times might vary greatly. For example, the SO of the adjective "solid" ranged from -2.75 (*Negative*) to 0.75 (*Positive*). This might suggest that the accuracy of PMI-IR depends greatly on the reliability, in terms of SO determination, of the index of the search engine being used.

Dave et al. [25] proposed a method, which they referred to as their baseline method, that determines the SO and strength of opinion words based on corpus statistics. Given an opinion word v_i and a sentiment class c_j, their method first computes $p(v_i|c_j)$, the normalized frequency of v_i in c_j, as follows [25]:

$$\mathrm{P}(v_i|c_j) = \frac{N(v_i, c_j) + 1}{\sum_{k=1}^{|V_j|} N(v_k, c_j) + |V_j|} \tag{2.31}$$

The above equation is similar to the one that a NB classifier uses to determine $\mathrm{P}(f_i|c_j)$ (Eq. (2.28) on page 43), with the opinion word v_i taking the role of the feature term f_i. In the equation, $N(v_i, c_j)$ is the number of times v_i was observed in c_j, and V_j denotes the set of opinion words in c_j. Note that add-one smoothing is applied to the above computation of normalized frequency, as Dave et al. noted that doing so produced the best classification results in their experiments. Their method then assigns a score to v_i using the following equation [25]:

$$score(v_i) = \frac{\mathrm{P}(v_i|c_j) - \mathrm{P}(v_i|\overline{c}_j)}{\mathrm{P}(v_i|c_j) + \mathrm{P}(v_i|\overline{c}_j)} \tag{2.32}$$

Such a score is a measure of bias ranging from -1 to 1. More specifically, their work was designed for binary outputs, and the set of sentiment classes C used for performing classification only contains two members, c_j and $\overline{c}_j$. For a given word v_i, the closer its score is to 1 (resp. -1), the stronger its opinion strength is with respect to c_j (resp. $\overline{c}_j$). This method is closely related to our proposed method for SO and opinion strength determination, in the sense that

both methods are based on corpus statistics. Our proposed method, however, does not restrict or makes any assumption about the number of classes in C.

The most commonly-used SO determination approach operates on the assumption that *semantic similarity between words implies their sentimental similarity*. Hu and Liu [61, 62] presented a bootstrapping algorithm that makes use of a seed set containing opinion-bearing words having known SO, such as "great" for *Positive* and "bad" for *Negative*. Their algorithm then estimates the SO of a word v_i based on the lexical relationships between v_i and the words in the seed set. Specifically, the SO of v_i is assumed to be the same as its synonyms' and opposite to its antonyms'. Several other algorithms adopted the same assumption. Kim and Hovy [72] described an algorithm capable of determining the SO and strength of opinions. Given are a word v_i, a sentiment class $c_j \in C = \{$*Positive, Negative*$\}$, and a set of words V_j that are members of c_j. Their algorithm computes $\mathrm{P}(v_i|c_j)$, which is the probability that v_i belongs to c_j, based on the occurrence of the *synonyms* of v_i that are in V_j. Kamps et al. [68] determined the SO of adjectives based on distances between words in WordNet [107], in which lexically related words are connected to each other. For instance, the relative distance of v_i to the two adjectives, "good" (*Positive*) and "bad" (*Negative*), provides an estimate of how positive or negative v_i is. Esuli and Sebastian [28] also assumed that lexical relationships define a relation of SO. Their method determines the SO of v_i by classifying the glosses of v_i, rather than simply comparing v_i to the predefined seed set.

To sum up, SO and opinion strength determination is commonly done by utilizing the lexical relationships between words. Existing methods assume that semantic similarity implies sentimental similarity, and we refer to such methods as *semantic-similarity-based* methods. As we discuss in Chapter 5.2, our analysis of a real world movie reviews dataset reveals that this assumption may not hold in sentiment analysis. Further, the SO of an opinion word may not agree with its generally understood SO. The word "frightening", for example, seems to express a negative feeling. When it appears in the sentence "*Movie X* is frightening", where *Movie X* is a horror movie, it may be representing a positive comment towards *Movie X*. In view of the weaknesses of the existing methods, we proposed a *relative-frequency-based* method for determining the SO and strength of opinion words as described in Chapter 5.2.

2.5.4 Classifying Reviews

Various studies on sentiment analysis classify reviews with respect to bipolar classes (e.g. [156, 115, 25, 61]). More recent studies, including ours, extend sentiment analysis to classify reviews based on multi-point rating scales, a task known as rating inference as noted (e.g. [114, 112, 41, 83]). In either group of studies, there exist two major approaches to classifying reviews. The first approach works in two steps. It first determines the SO and/or strength of opinions, for example, using any of the methods described in the previous subsection. It then classifies a review d by aggregating the SO of the opinions d contains. d is, for instance, classified as *Positive* if the dominating SO of its opinions is positive; and *Negative* otherwise. Studies taking this approach include [156, 25, 83]. The second approach is similar to topic-based classification, with the topics being sentiment classes such as *Positive* and *Negative* in binary classification, or scalar ratings for rating inference. This approach mainly works by representing d as a document vector based on the bag-of-words model, and then classifying d using machine learning algorithms [115, 114, 112, 41, 149]. We describe three related studies ([115, 114, 41]), among others, taking this approach in the following paragraphs.

Pang et al. [115] investigated whether binary sentiment classification can be addressed as a topic classification task. They applied three machine learning algorithms, namely NB, Support Vector Machines (SVMs) and Maximum Entropy, to movie reviews. They attempted to incorporate various features of the reviews into the bag-of-words model, including the positions and the POS of words in the reviews, but the performance of the three algorithms was found inferior to that reported for topic classification. Pang et al. therefore concluded that sentiment classification is more difficult than topic classification, and that discourse analysis of reviews is necessary for more accurate sentiment analysis. In view of these, we attempted to identify features representing the overall recommendations of users in reviews as a kind of discourse analysis in our work.

Pang and Lee [114] formulated rating inference as a metric labeling problem. The main idea of their work is to minimize the difference between the predicted rating of a test review and the ratings of the reviews that are similar to the test review. Formally, their metric labeling problem seeks to minimize [114]:

$$\sum_{i \in \Omega} \left[-\pi(d_i, \hat{r_i}) + \alpha \sum_{j \in knn(d_i)} \Delta(\hat{r_i}, r_j) sim(d_i, d_j) \right] \tag{2.33}$$

where Ω denotes the set of unlabeled (test) review indices, d_i and d_j are reviews, $\pi(d_i, \hat{r_i})$ is called an "initial label preference function" that assigned a predicted rating $\hat{r_i}$ to the review d_i, α is a weight coefficient, $knn(d_i)$ is the k-nn of d_i according to the similarity function $sim(d_i, d_j)$, and $\Delta(\hat{r_i}, r_j)$ is a distance metric between the predicted rating of d_i and the actual rating of its neighbor d_j.

Pang and Lee experimented with two n-ary classifiers, namely One-Versus-All (OVA) SVM and linear Support Vector Regression (SVR), as $\pi(d_i, \hat{r_i})$, for determining the predicted rating $\hat{r_i}$ for the test review d_i. Further, they proposed Positive-Sentence Percentage (PSP) as the similarity function $sim(d_i, d_j)$. PSP is defined as the number of positive sentences divided by the number of subjective sentences in a review. Pang and Lee evaluated their work on four *author-specific corpora* of movie reviews using a 3-point and a 4-point rating scale. Results showed that their metric-labeling algorithm based on PSP improved the performance of OVA SVM and SVR in most experimental settings. In the 4-point case, however, the performance of their algorithm was not statistically better than that of SVR.

Pang and Lee noted that formulating rating inference as a metric labeling problem allows for *transductive semi-supervised learning*, meaning that the nearest neighbors of a test review can come from both the training and the test sets [114]. Based on this idea, Goldberg and Zhu [41] proposed a graph-based semi-supervised learning algorithm for rating inference. Given are a set of n reviews $d_1, d_2, ..., d_n$. Without loss of generality, the first $\theta < n$ reviews are labeled with ratings $r_1, r_2, ..., r_\theta \in C$, and the rest are unlabeled. $C = \{c_1, c_2..., c_{|C|}\}$ is the set of numerical sentiment classes, and $c_1 < c_2 < ... < c_{|C|} \in \mathbb{R}$. Goldberg and Zhu's work aims at finding a rating function $f : d_i \rightarrow \mathbb{R}$ that gives a rating $f(d_i)$ to review d_i. They addressed this as an optimization problem that minimizes the following *loss function* $\mathcal{L}(f)$ [41]:

$$\sum_{i \in \Theta} \mathcal{M}(f(d_i) - r_i)^2 + \sum_{i \in \Omega} (f(d_i) - \hat{r_i})^2$$
$$+ \sum_{i \in \Theta} \sum_{j \in knn_\Theta(d_i)} \alpha * sim(d_i, d_j)(f(d_i) - f(d_j))^2$$

$$+\sum_{i\in\Omega}\sum_{j\in k'nn_{\Omega}(d_i)}\beta * sim(d_i,d_j)(f(d_i)-f(d_j))^2 \tag{2.34}$$

where Θ and Ω are labeled and unlabeled review indices respectively, r_i is the observed rating of d_i, $\hat{r_i}$ is the predicted rating assigned to d_i by a separate label preference function $f(d_i)$, which is equivalent to $\pi(d_i, \hat{r_i})$ in Eq. (2.33), $knn_{\Theta}(d_i)$ and $k'nn_{\Omega}(d_i)$ are the nearest k labeled neighbors and k' unlabeled neighbors of d_i respectively, α and β are weight coefficients assigned to labeled and unlabeled neighbors respectively, $sim(d_i, d_j)$ measures the similarity between the two reviews d_i and d_j as in Eq. (2.33), and $\mathcal{M}$ is a weight representing the influence of r_i. If $\mathcal{M} \to \infty$, then $f(d_i) = r_i$ becomes a hard constraint.

Eq. (2.34) allows for transductive semi-supervised learning as the neighbors of a review d_i can be labeled or unlabeled. A small loss thus implies that the rating of a test review is close to the ratings of its labeled neighbors as well as unlabeled neighbors [41]. Goldberg and Zhu described a special case of $\mathcal{L}(f)$ when $\beta = 0$ and $\mathcal{M} \to \infty$, in which unlabeled neighbors of d_i does not contribute to the rating inference process. In such case, for an unlabeled review d_i, the optimization problem becomes:

$$i \in \Omega : \mathcal{L}_{\beta=0,\mathcal{M}\to\infty}(f(d_i)) = (f(d_i)-\hat{r_i})^2 + \sum_{j\in knn_{\Theta}(d_i)} \alpha\cdot sim(d_i,d_j)(f(d_i)-r_j)^2 \tag{2.35}$$

This equation corresponds exactly to the metric labeling formulation of rating inference in [114], except that this equation considered the squared difference between actual and predicted labels, whereas that of Pang and Lee [114] used the absolute difference.

Goldberg and Zhu tested their work on the four author-specific corpora used in [114] with respect to a 4-point rating scale, using SVR as the rating function $f()$. They reported results based on two similarity measures $sim(d_i, d_j)$, namely PSP and Word Vector (WV), a measure computed as the cosine between weighted word vectors of d_i and d_j. They demonstrated that their graph-based semi-supervised learning algorithm with PSP outperformed Pang and Lee's work when the numbers of labeled training reviews were relatively limited, and that unlabeled reviews can help improve classification accuracy.

Chapter 3

Alleviating Data Sparseness by Fuzzy Association Rule Mining and Item Taxonomies

3.1 Introduction

Information overload has led to the popularity of recommender systems, which receive information about users' needs, and recommend to them items that may fit their needs. As discussed in Chapter 2.1, there are three types of recommender systems: content-based, CF-based and knowledge-based. CF is generally acknowledged to be the most promising and successful technique in recommender systems, providing personalized recommendations to users based on their previously expressed preferences and those of other similar users. There has been considerable research into CF, yet the issue of data sparseness, the cold-start problem and the non-transitive association problems remain open challenges.

This chapter presents our work on alleviating data sparseness in CF based on the ARM (association rule mining) paradigm with the use of item taxonomies, and describes the proposed CF framework based on Fuzzy Association Rules And Multiple-level Similarity (FARAMS). FARAMS is a model-based CF framework that exploits the similarities between items and those between item categories for making personalized recommendations. Given is a *taxonomy*, or *is-a hierarchy*, of domain items, in which items are classified into different categories. Figure 3.1 depicts an example of an item taxonomy

having a multiple-level hierarchical structure, with *items* appearing at the lowest level of the hierarchy, and *categories* appearing at higher-levels represent more general concepts. Given such a taxonomy, FARAMS mines association rules between items (hereafter, item-level association rules) and those between their categories (hereafter, category-level association rules) from users' preference data. FARAMS always attempts to recommend items to users based on item-level association rules. However, when the known preferences of the active user are insufficient for generating recommendations, FARAMS utilizes category-level association rules for determining recommendable items for the active user.

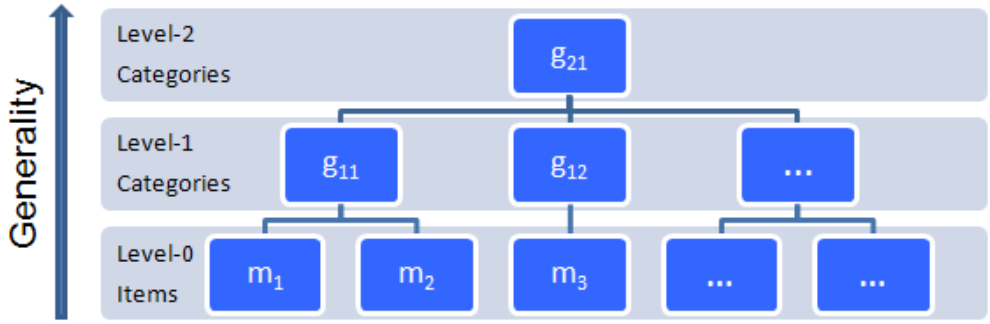

Figure 3.1: Example of an item taxonomy.

The use of ARM-based techniques for CF has a number of desirable outcomes. Firstly, item taxonomies can easily be integrated into the ARM process as shown in [150, 50]. The fact that relationships between items are already implicit in their taxonomies may help reduce the non-transitive association problems [69]. Secondly, multiple- or cross-level ARM applies association between item categories to address data sparseness, resulting in improved recall rates of Top-N recommendations as shown in [71]. Thirdly, the use of item taxonomies in ARM techniques increases the number of recommendable items, which are items that can be recommended by a recommender system, to users for whom only limited known preference data is available. Finally, ARM techniques provide the flexibility to, if necessary, easily discover associations between content-related attributes and user ratings on items. This is not considered in FARAMS, but is used for addressing the cold-start problem in CF as discussed in the next chapter.

The rest of this chapter is organized as follows. The next section briefly describes related work on ARM-based CF. Chapter 3.3 details the design of FARAMS. Chapter 3.4 discusses evaluation results on FARAMS, and finally

Chapter 3.5 summarizes our contributions in this chapter.

3.2 Related work on ARM-based CF

We described in Chapter 2.3.2 that Lin et al. [89] proposed an Adaptive-Support Association Rule Mining (ASARM) algorithm for CF. ASARM mines rules for one target item at a time, and automatically adjusts the minimum support value to mine a user-specified number of rules. Numerical ratings are discretized into two classes, *Like* and *Dislike*, based on some chosen threshold value. This is done to support the creation of a transactional representation of the $U \times I$ ratings matrix in CF to facilitate the rule mining process.

Kim and Kim described another ARM-based CF algorithm in [71]. Their algorithm, known as MAR, applied association rules between categories in multiple-level item taxonomies to address data sparseness. They showed that by taking advantage of item taxonomies, the MAR algorithm increased the number of recommendable items for those users whose known preferences are otherwise so limited that it is not possible to produce recommendations. The rating a user has given an item is classified as *Like* if it is above the average rating given on all items by that user in [71].

In summary, ASARM mines rules for CF based on an adaptive minimum support strategy, but it does not consider item taxonomies in the mining process. On the other hand, the MAR algorithm utilizes relationships between categories in item taxonomies, but it mines rules for all items in the database. Consequently, recommending less popular items will be more difficult according to Lin et al. [89]. Note that both ASARM and MAR may suffer from the sharp boundary problem, which arises due to the boolean discretization of ratings. FARAMS can be considered an integration of the ASARM and MAR algorithms, enhanced with fuzzy logic for modeling numerical ratings.

3.3 The FARAMS Framework

The FARAMS framework adopted some existing ARM techniques, including ASARM [89] and MAR [71], to generate collaborative recommendations. These techniques, when applied to CF, are integrated with classification techniques for handling quantitative ratings data. FAR mining, which is a variation of

the classical ARM techniques, is used to address the resulting sharp boundary problem in existing techniques [48].

FARAMS is carried out in four major steps. The first step is data preprocessing, in which data are prepared in a format that is suitable for the subsequent tasks. The second step mines item-level and category-level association rules from user preferences given a taxonomy of domain items. Association rules that satisfied user-specified interestingness constraints, such as minimum support and confidence values, are stored in the system. They serve as a compact model of user preferences, and are used when users request recommendations. The third step is prediction computation, which determines relevant rules for a user and assigns predicted preferences to items recommended by those relevant rules. The fourth step generates recommendations. If the number of recommendable items is smaller than the desired number of recommendations, FARAMS considers category-level similarities among items to predict user preferences for items that are not covered by item-level association rules. The following sections provide further details about the tasks involved in each step.

3.3.1 Data Preprocessing

The data preprocessing step involves four tasks. They include transforming rating matrixes into transactional databases, computing user preferences for categories of domain items, fuzzifying user preferences for FAR mining and transforming transactions into *TID-lists* to allow for efficient support counting.

Transforming ratings matrixes into transactional databases

CF ratings data are usually represented as a ratings matrix as noted. They have to be transformed into a transactional database for ARM tasks. We use an example to illustrate the transformation process based on the sample ratings matrix R in Table 2.3 (page 34). Table 3.1 (a) shows a transactional representation of R. In the table, each transaction consists of a transaction identifier (TID), which is the User ID of the user to which the transaction belongs, as well as the IDs and ratings of the items that have been rated by that user.

Table 3.1: Transforming user preferences for (a) items and (b) item categories into transactions.

TID	Items
u_1	i_1(1), i_2(4), i_5(2)
u_2	i_4(5)
u_3	i_1(5), i_2(2), i_3(1), i_5(5)
u_4	i_1(5), i_2(1)

(a)

TID	Categories
u_1	[g_1](4), [g_2](2), [g_3](1.5), ...
u_2	[g_5](5), [g_9](5), ...
u_3	[g_1](2), [g_2](5), [g_3](5), ...
u_4	[g_1](1), [g_3](5), [g_4](5), ...

(b)

Table 3.2: A relation matrix of items (e.g. movies) and their categories (e.g. movie genres).

	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
i_1	0	0	1	1	0	1	1	0	0	0
i_2	1	0	0	0	1	0	0	0	1	1
i_3	0	0	0	0	1	0	0	1	0	1
i_4	0	0	0	0	1	0	0	0	1	0
i_5	0	1	1	1	0	1	1	1	1	0

Computing user preferences for item categories

FARAMS makes use of multiple-level similarities (hereafter referred to as MS), which are implicit in item taxonomies, among items to generate recommendations for users whose known preferences are so limited such that sufficient recommendations cannot be produced. Such MS among items are encoded in the relationships between items and item categories in a given taxonomy or is-a hierarchy of domain items. Note that user preferences for item categories are not readily available in CF datasets as users in general gave ratings on items rather than item categories. We therefore need to compute user preferences for categories from the original transactions containing user preferences for items, so that rules involving categories can be mined.

Suppose items in the sample rating matrix in Table 2.3 are organized into a set of 10 categories, $G = \{g_1, g_2, ..., g_{10}\}$. Table 3.2 is a sample relation matrix A of items, such as movies, and their categories, such as movie genres. In the matrix, rows and columns represent items and categories respectively. A value A_{mn} in the matrix is 1 if the m^{th} item belongs to the n^{th} category.

Table 3.3: Transactions containing ⟨*Item, Fuzzy Set*⟩ pairs and normalized membership degrees.

TID	Items
u_1	$\langle i_1, D\rangle$(1), $\langle i_2, L\rangle$(0.5), $\langle i_2, N\rangle$(0.33), $\langle i_2, D\rangle$(0.17), $\langle i_5, L\rangle$(0.17), $\langle i_5, N\rangle$(0.33), $\langle i_5, D\rangle$(0.5)
u_2	$\langle i_4, L\rangle$(1)
u_3	$\langle i_1, L\rangle$(1), $\langle i_2, L\rangle$(0.17), $\langle i_2, N\rangle$(0.33), $\langle i_2, D\rangle$(0.5), $\langle i_3, D\rangle$(1), $\langle i_5, L\rangle$(1)
u_4	$\langle i_1, L\rangle$(1), $\langle i_2, D\rangle$(1)

A user may have preferences for multiple categories, and ratings of items in the same category can be different. A user's preference for a category is estimated as the average rating (s)he gave to items in that category. In the sample transactions shown in Table 3.1(b), a category ID is enclosed in square brackets, followed by the average rating the user has given the items that belong to it.

Fuzzifying ratings

Ratings are fuzzified in four steps. These steps are the same for both items and categories. First, the fuzzy sets and membership functions for ratings are determined. Second, items in transactions are expanded into ⟨*Item, Fuzzy Set*⟩ or ⟨*Category, Fuzzy Set*⟩ pairs. Third, the degree of membership of each rating is determined with respect to each fuzzy set. Finally, the fuzzified ratings are normalized so that each transaction makes the same contribution, which is 1 (Chapter 2.3.3 on page 32).

Table 3.3 extends Table 3.1(a) to show the resulting fuzzified transactions given the fuzzy sets and membership functions in Figure 2.2(a). The table does not show ⟨*Item, Fuzzy Set*⟩ pairs with membership degrees of 0 for better readability.

Transforming transactions for efficient support counting

FARAMS obtains the support counts of itemsets by making multiple passes over the data as in [89]. A major optimization we made in our approach is that the transactional representation of ratings matrix described in the previous subsections is further transformed into a *vertical TID-list* format [174]. In

such format, an ⟨*Item, Fuzzy Set*⟩ pair takes the role of a user in the original transactional representation. Each pair is associated with a list of users who rated the pair, and the corresponding fuzzified ratings it received from the users. Table 3.4 shows an illustration of the vertical TID-list representation of user preferences for items. This transformation enables efficient support counting of itemsets as described in the next subsection.

TID	**Items**
$\langle i_1, L\rangle$	u_3(1), u_4(1)
$\langle i_1, D\rangle$	u_1(1)
$\langle i_2, L\rangle$	u_1(0.5), u_3(0.17)
$\langle i_2, N\rangle$	u_1(0.33), u_3(0.33)
$\langle i_2, D\rangle$	u_1(0.17), u_3(0.5), u_4(1)
...	...

Table 3.4: Transformed transactions in the vertical TID-list format for efficient support counting.

3.3.2 Mining User Preferences

The overall structure and flow of our algorithms for adjusting the minimum support value and for mining user preferences are similar to those described in [89]. This section therefore focuses on the adaptations and extensions made in the association rule miner of FARAMS. We first describe several issues we considered when designing the mining algorithm, followed by the description of the mining algorithm of FARAMS.

Association mode used

ARM-based CF frameworks may employ two *association modes*, namely user association and item association. The ideas underlying these are equivalent to the ideas of user- and item-based CF. User association identifies similarities between users and recommends items preferred by users similar to the active user. Mining user associations produces rules that are in the form of "$\langle u, L\rangle \rightarrow \langle a, L\rangle$"(recall that L represents the fuzzy set $Like$), where u and a are users. For a target item

t that has not been rated by the active user a, this rule *fires* if t has been liked by user u [89]. Item t will then be considered a recommendable item for user a.

Item association identifies similarities between items, and is used in [71] and [90]. It mines rules that are in the form of "$\langle i, L\rangle \rightarrow \langle t, L\rangle$", where i and t are items. This rule fires if the active user a has liked item i but has not rated t previously. Item t in the head of the rule will then be considered a recommendable item for user a. This idea can be applied to categories by treating a category as an item. FARAMS exploits item associations to facilitate the use of item taxonomies in the recommendation process. As such, FARAMS is an item-based recommendation algorithm.

Defining candidate 1-itemsets

As FARAMS mines rules for one target item (t) at a time, association rules are generated using only itemsets containing t. Candidate 1-itemsets can therefore be limited to the "related items", defined as the union of all items that appeared in transactions containing t. FARAMS mines rules in an apriori-like fashion, which iteratively generates κ-itemsets by joining two (κ-1)-itemsets. Once the related items are determined in the first iteration, their associated TID-lists serve as a reduced database for support counting in the subsequent iterations.

It is expected that taking into account only "related items" will not affect the results of the rule mining process. This is because the *downward closure* property of support values states that all subsets of a frequent itemset must be frequent [6]. If an item did not appear with item t, and as rules are generated using only itemsets containing item t, any κ-itemset, where $\kappa > 1$, containing that item and item t must be infrequent. The item can therefore be excluded from consideration in the first place.

Computing fuzzy support values

Traditional support counting methods obtain the fuzzy support count of an itemset by scanning the entire transactional database. In our work, the vertical TID-list representation of user preferences allows FARAMS to scan the database more efficiently. FARAMS only needs to perform simple joining operations and inspect κ records in the database in order to compute the fuzzy support of a κ-itemset [174]. We illustrate this with the following examples based on Table 3.5.

Example 3.1 *Computation of $FS_{\langle i_1,L\rangle}$, the fuzzy support of the 1-itemset $\langle i_1, L\rangle$: $FS_{\langle i_1,L\rangle}$ can be found efficiently by inspecting only one record (the first row of the table), and the result is (1 + 1) = 2.*

Example 3.2 *Computation of $FS_{\{\langle i_1,L\rangle\langle i_2,D\rangle\}}$, the fuzzy support of the 2-itemset $\{\langle i_1, L\rangle\langle i_2, D\rangle\}$: before computing $FS_{\{\langle i_1,L\rangle\langle i_2,D\rangle\}}$,* FARAMS *inspects the transactions of both items and performs a simple join operation. As shown in Table 3.5, u_3 and u_4 appeared in the transactions of both items. The fuzzy support count of the 2-itemset can then be determined, and is found to be equal to (1 * 0.5) + (1 * 1) = 1.5.*

TID	**Items**
$\langle i_1, L\rangle$	u_3**(1)**, u_4**(1)**
$\langle i_2, D\rangle$	u_1(0.17), u_3**(0.5)**, u_4**(1)**

Table 3.5: TID-lists of $\langle i_1, L\rangle$ and $\langle i_2, D\rangle$.

The mining algorithm

Algorithms 3.3.1 and 3.3.2 describe our mining algorithm and one of its subroutines, *find_frequent_1_itemsets*. As noted, our algorithm mines rules for a target item t at a time. Given t and the TID-list representation of all items (TID), the algorithm first determines the list of related items that are frequent (I_1) and their associated TID-lists (TID_t) using Algorithm 3.3.2. It then proceeds to find frequent κ-itemsets (I_κ), where $\kappa \geq 2$, in an iterative manner. In each iteration, association rules (R_κ) are generated from the frequent κ-itemsets, and at most *maxNumRules* association rules having the highest support values (R_t) are returned. If the total number of rules mined in the κ iterations is larger than *maxNumRules*, the *aboveMaxNumRulesFlag* is raised, which will then cause the rule mining process to terminate. After all the iterations, the *belowMinRulenumFlag* is raised if the total number of rules mined is smaller than *minNumRules*.

The subroutines in Algorithm 3.3.1 are outlined as follows:

1. **find_frequent_1_itemsets**(t, TID): This subroutine, as described in Algorithm 3.3.2, determines the set of frequent 1-itemsets (I_1) and their

Algorithm 3.3.1 *The mining algorithm.*

Input: TID, t, *minSupp* (minimum support), *minConf* (minimum confidence), [*minNumRules, maxNumRules*], *maxRuleLength* (maximum number of items in a rule's body).

Output: Set of association rules (R_t), so that each rule in R_t: (1) has t in its head, (2) with no more than *maxRuleLength* items in its body, and (3) satisfies the *minSupp* and *minConf* constraints. The number of rules in R_t is at most *maxNumRules*. If the number of rules in R_t is above *maxNumRules* (resp. below *minNumRules*), raise the *aboveMaxNumRulesFlag* (resp. *belowMinNumRulesFlag*).

Steps:

1. $(I_1, TID_t) \leftarrow$ **find_frequent_1_itemsets** (t, TID);
2. $\kappa \leftarrow 2$;
3. **for** ($\kappa \leq$*maxRuleLength* + 1) and ($I_{\kappa-1} \neq \emptyset$) and (not R_t.*aboveMaxNumRulesFlag*) **do**
4. $cand_\kappa$ = **gen_candidate** ($I_{\kappa-1}$);
5. **for** each $c \in cand_\kappa$ **do**
6. c.*fuzzySupport* = **compute_fuzzy_support** (c, TID_t);
7. **if** (c.*fuzzySupport* $\geq$ *minSupp*) **then**
8. add c to I_κ;
9. **end if**
10. **end for**
11. R_κ = **gen_rules** (I_κ, t, *minConf*);
12. **if** ($|R_t| + |R_\kappa| >$*maxNumRules*) **then**
13. set R_t.*aboveMaxNumRulesFlag*;
14. **end if**
15. R_t = *maxNumRules* rules with highest support from $R_t.rules \cup R_\kappa.rules$;
16. $\kappa \leftarrow \kappa + 1$;
17. **end for**
18. **if** ($|R_t| <$*minNumRules*) **then**
19. set R_t.*belowMinNumRulesFlag*;
20. **end if**
21. **return** (R_t);

Algorithm 3.3.2 *The **find_frequent_1_itemsets** subroutine.*

Input: t, TID.

Output: Set of frequent 1-itemsets (I_1) that are related to t, and their associated TID-lists (TID_t).

Steps:

1. $TID_t \leftarrow= \emptyset$
2. $U_t \leftarrow \{$users who had rated $t\}$;
3. **for** each $u \in U_t$ **do**
4. $I_u \leftarrow \{$items rated by $u\}$;
5. $cand_1 \leftarrow cand_1 \cup I_u$;
   //$cand_1$ denotes the candidate 1-itemsets that are related to t
6. **end for**
7. **for** each $c \in cand_1$ **do**
8. c.*fuzzySupport* = **compute_fuzzy_support** (c, TID);
9. **if** (c.*fuzzySupport* $\geq$ *minSupp*) **then**
10. add c to I_1;
11. add TID-list of c to TID_t;
12. **end if**
13. **end for**
14. **return** (I_1, TID_t);

---

associated TID-lists (TID_t) given the target item t and the TID-list of all items (TID). TID_t serves as a reduced database for support counting in the subsequent iterations.

2. **gen_candidate**($I_{\kappa-1}$): This subroutine generates the candidate κ-itemsets ($cand_\kappa$) based on $I_{\kappa-1}$ using the Apriori-gen function proposed in [6].

3. **compute_fuzzy_support**(c, TID_t): This subroutine determines the fuzzy support value of the candidate itemset c. An itemset is added to I_κ if its fuzzy support value is above *minSupp*.

4. **gen_rules**($I_\kappa, t,$ *minConf*): This subroutine produces association rules having t as their heads using I_κ. The subroutine returns R_κ, which is the set of rules having confidence values above the predefined minimum (*minConf*).

The subroutine in Algorithm 3.3.2 is outlined as follows:

1. **compute_fuzzy_support**(c, TID): This subroutine determines the fuzzy support value of the candidate 1-itemset c. It is the same as the **compute_fuzzy_support**(c, TID_t) subroutine in Algorithm 3.3.1, except that the set of all TID-lists (TID) is used for support counting since TID_t is not yet determined at this stage.

The same procedures are used for mining association rules from both product-level items and categories. The association rules mined in this step are stored in the system. They will be used when users request recommendations.

3.3.3 Predicting Scores of Recommendable Items

When the active user a requests recommendations, the system finds the list of items user a has previously rated, based on which the relevant rules, which are rules that fire for him or her, are determined. Items in the heads of the relevant rules are considered recommendable items for user a, and are assigned predicted scores based on some *interestingness measures* of the relevant rules [89]. An item may appear in the heads of more than one rule. After all relevant rules are determined, the interestingness measures of the rules that recommend the same item are summed up to obtain the item score predictions.

As noted, indicators of the interestingness of a rule, such as support and confidence values, are used as the basis for item score predictions. Lin et al. [89], for example, made use of the product of a relevant rule's support and confidence, while Kim and Kim [71] only considered a rule's confidence. The correlation value (CORR), which measures the correlation between the body and the head of a rule, may also be used as an interestingness indicator. The choice of interestingness indicators for predicting scores of recommendable items is application- and dataset-specific. We therefore decided the appropriate interestingness indicators to use empirically in Chapter 3.4.3.

3.3.4 Generating Recommendations

After predicted scores are assigned to recommendable items, the recommendation process proceeds to determine which of the items are actually recommended to the active user a. This section describes the how FARAMS generates recommendations, as well as when and how MS (multiple-level similarity as noted) is utilized. Then, it describes the recommendation algorithm used in FARAMS.

Recommendation strategy

There exist two main strategies for generating recommendations. The first strategy recommends items with predicted scores above a score threshold. The threshold applied in [89], for example, is a linear function of the number of rules, but the reason for choosing such a value is unclear. The second strategy recommends a fixed number of items and is known as Top-N recommendations. This strategy is used in [71]. Both strategies rank items in descending order of their predicted scores, so that items with higher scores are recommended first. There are two reasons why FARAMS uses the Top-N approach to recommend items. First, the number of recommended items is fixed and therefore controllable. Secondly, Top-N helps determine when MS should be used.

Item-level association rules are usually more specific and relevant to a user's preferences than the preferences predicted using more general, category-level association rules [71]. Therefore, when computing the predicted scores of recommendable items, we assign a weight w_l to the interestingness score of the relevant rules, where l is equal to the level of the association rules used to reflect the generality of the rules. Specifically, level-0 means the item-level, level-1

means the first level categories, and so on (Figure 3.1).

Recommendations using multiple-level similarity (MS)

MS is utilized to produce recommendations when the known preferences of the active user are very limited and, consequently, the number of recommendable items may be very limited. This is done as follows:

1. Find the preferred categories G_a of the active user.
2. Determine association rules that fire for G_a (rules containing G_a in their bodies).
3. Find rules containing items in those associated categories and assign predicted scores to items in their heads. The predicted scores are weighted by w_l and user a's preferences for those categories. The predicted preference for an item that belongs to more than one category is estimated as the average of the preferences user a has on its categories.

The recommendation algorithm

Algorithm 3.3.3 describes the recommendation algorithm, or recommender, of FARAMS. Given user a and his/her previously rated items (I_a), the algorithm determines the set of relevant association rules, which are rules that fire for user a, from the rule base. The items in the heads of the relevant rules are assigned predicted scores if they are not in I_a. If the number of recommendable items is smaller than the predefined Top-N value (N), MS between items are used for determining more recommendable items for the active user. Finally, the recommendable items (I_r) are sorted in descending order of their predicted scores, and the N items having the highest scores are recommended to user a.

The subroutines in the algorithm are outlined as follows:

1. **recommend**(a, I_a): This subroutine returns a list of recommendable items (I_r) and their predicted scores for the active user (a). I_a denotes the set of items rated by user a. A rule in the rule base is considered relevant if I_a contains all items in the rule's body but not its head.
2. **recommend_by_MS**(a, I_a, I_r): This subroutine determines the preferred categories of user a and uses them to generate recommendations. The

Algorithm 3.3.3 *The recommender of* FARAMS.

Input: The active user a, the set of items rated by user a (I_a), and the maximum number of recommendations (N).

Output: The set of recommended items (I_r) for user a.

```
1  I_r = recommend(a, I_a);
2  if (|I_r| < N) then
3     I_r = recommend_by_MS(a, I_a, I_r);
4  end if
5  sort(I_r); // in descending order of their scores;
6  return (top N items in I_r);
```

---

procedures for determining relevant rules and recommendable items are similar to those described in (1).

3. **sort**(I_r): This subroutine ranks the recommendable items in descending order of their scores.

3.4 Experimental Results

We carried out several experiments to evaluate the performance of FARAMS, and to compare it with some related work. In this section, we first describe the datasets used and the settings of the experiments and then provide the experimental results.

3.4.1 Datasets

We used three CF datasets, or their subsets, as test-beds, depending on the purposes of our experiments. The datasets include:

1. **MovieLens:** The MovieLens 100k dataset[7] contains 100,000 ratings of 1,682 movies from 943 users. Ratings are discrete values from 1 to 5. Movies in the dataset are categorized into a two-level hierarchical

[7]MovieLens 100k Ratings Data Set: http://www.grouplens.org/node/73

structure, with movies as items and movie genres as categories. This dataset is used in several experiments as described in the subsequent subsections.

2. **Jester:** The Jester dataset contains 4.1 million ratings of 100 jokes from 73,421 users. Ratings are real values ranging from -10 to 10 [43]. A subset containing 49,502 ratings of 100 jokes from 2,000 users were used to evaluate the effect of fuzzified ratings on recommendation quality.

3. **EachMovie:** The EachMovie data set contains 2,811,983 ratings of 1,628 movies entered by 72,916 users [101]. Ratings were recorded on a six-point numerical scale (0.0, 0.2, 0.4, 0.6, 0.8, 1.0). Movies are categorized into a two-level hierarchical structure. A small and dense subset of this dataset, containing ratings from 1,100 users who have rated more than 100 movies, was used for comparison with the ASARM algorithm [89].

Ratings in the above datasets were fuzzified for performing various experiments. It is possible to learn fuzzy sets and membership functions from training data [36, 59]. Since the focus of this work is not on modeling user preferences, we used the simple fuzzy sets and membership functions shown in Figure 2.2 (page 33), as described later in the individual results and discussions. In fact, we performed a set of experiments using different membership functions and found little sensitivity of results, especially for the MovieLens and EachMovie datasets which have small rating scales.

3.4.2 Experimental Settings

Parameters

We now describe the values of three parameters we used in the experiments. The first parameter is the desired range of the number of mining rules [*minNumrules*, *maxNumrules*]. We adopted the range [10,100] as suggested in [89] for mining item-level association rules. For categories, we adopted the range [1, $|G|$-1], where $|G|$ equals to the number of categories in the corresponding dataset. We did not impose any confidence threshold due to the limited number of desired rules to be mined. Given the automatically determined *minSupp*, we considered the rules having the highest confidence values and positive correlation values to be interesting.

The second parameter is the consideration of positive and negative preferences data. In FARAMS, it is possible to obtain *Like*, *Neutral* and *Dislike* associations: for example, "$\langle i_1, L\rangle, \langle i_2, N\rangle \rightarrow \langle i_3, D\rangle$". Lin et al. [89] found that employing both *Like* and *Dislike* associations does not outperform employing *Like* associations alone. We therefore adopted the same strategy.

The third parameter is the weight assigned to level-l association rules. Items in the MovieLens and EachMovie datasets are organized into similar 2-level taxonomies. The weights assigned to level-0 (item-level) and level-1 (category-level) association rules are 0.9 and 0.1 respectively, as suggested in [71].

Method

We employed the all-but-1 protocol in this study, following the setting in [71]. Specifically, for each active user a in our dataset, all except 1 ratings data given by a are used as the training set, based on which association rules are mined for generating recommendations. We created five training-test splits randomly, repeated all experiments on the five splits, and reported the average results obtained. In each trial of each experiment, a list of Top-N recommendations was provided to each active user a. If the hidden item in the test set was on the recommendation list, it is called a *hit*. The recall rate of an algorithm is defined as the number of hits over the number of hidden items in the test set.

3.4.3 Results and Discussions

We now discuss our results, which demonstrate the effects of various factors on the recommendation quality as measured by the algorithms' recall rates.

Rule length

Rule length refers to the number of items contained in the body of a rule. In [89], a rule's body can contain multiple items, while in [71], it can only contain a single item. Figure 3.2 shows the recall rates achieved using both approaches in FARAMS as applied to the MovieLens dataset. SGL represents the recall rates of the algorithm having a maximum rule length of 1. MUL-8 represents that of the algorithm having a maximum rule length of 8. Such a setting has been found to perform well in [89].

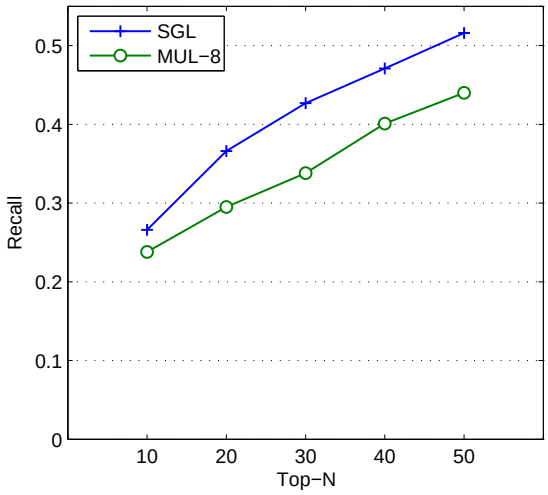

Figure 3.2: Recall rates achieved using different maximum rule lengths.

Figure 3.2 shows that the performance of SGL is better than that of MUL-8 for all values of N. This indicates that rules containing a single body item produce better recommendations. One reason for this may be that rules are easier to fire when they are shorter, so that a larger number of applicable rules as well as recommendable items can be found for the active user, resulted in a lower demand for the use of MS to produce recommendations. As stated in Chapter 3.3.4, item-level association rules are usually more relevant to user preferences, and therefore produce better predictions. Given these results, in the remaining experiments we use 1 as the maximum rule length.

Scores of recommendable items

Chapter 3.3.3 describes the various interestingness measures of association rules that can be used for predicting the scores of recommendable items. This experiment helps us determine the appropriate interestingness measures that should be adopted. The measures we evaluated include the fuzzy confidence (C), the product of fuzzy support and fuzzy confidence (SC), and the correlation (CORR) of association rules. Figure 3.3 shows the results of this experiment.

As shown in Figure 3.3, the best results are produced by computing predicted scores using fuzzy confidence (C). While support indicates a rule's statistical significance in the entire database, confidence indicates the interestingness of the rule head with respect to the rule body. For an association rule that fired for

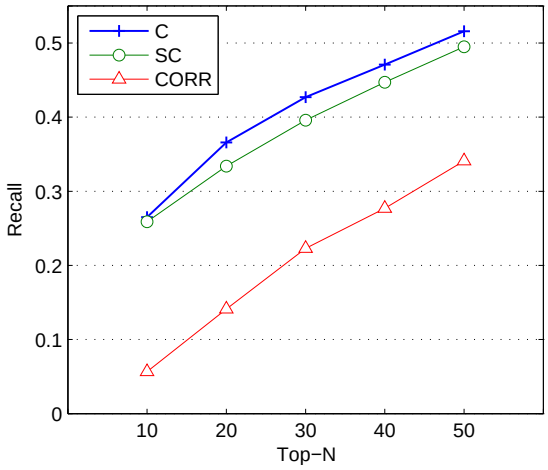

Figure 3.3: Recall rates achieved using different interestingness measures for predicting the scores of recommendable items.

a given user, since we already know that the user has liked the items in its body, it is reasonable to choose to use confidence to measure the probability that the user will like the head. We adopt C for predicting the scores of recommendable items in the subsequent experiments.

Effects of multiple-level similarity

Figure 3.4 compares the performance of FAR (that is, without using MS) and FARAMS. When N is equal to 10 and 50, FAR and FARAMS produce very similar recall rates. When N is equal to 20, 30 and 40, FARAMS outperforms FAR by around 5.5%, 3.2% and 3.1% respectively. Such results suggest that the use of MAR helps alleviate data sparseness in CF dataset by increasing the number of recommendable items to users, but the improvement is small in the MovieLens dataset. This demonstrates that category-level association rules are less relevant to users' preferences.

Effects of fuzzy association rules

Two sets of experiments were performed to evaluate the effects of fuzzy association rules on recommendation quality using the MovieLens and the Jester datasets. Recall that a fuzzy association rule is in the form of "$\langle i_1, L\rangle \rightarrow \langle i_2, L\rangle$", or "$\langle g_1, L\rangle \rightarrow \langle g_2, L\rangle$", where i_1 and i_2 are items, g_1 and g_2 are categories, and L represents the *Like* fuzzy set in these examples. As the MovieLens dataset

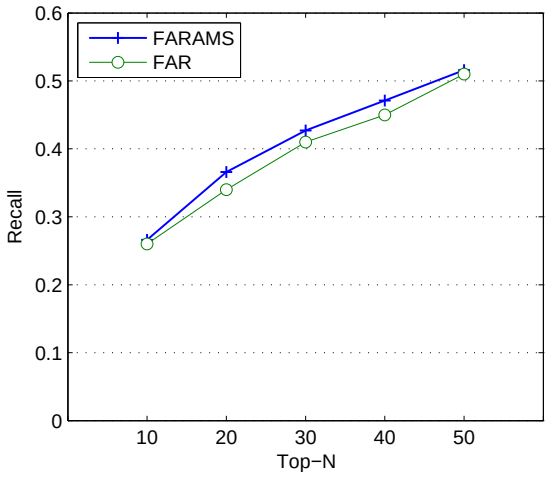

Figure 3.4: Recall rates achieved with and without utilizing multiple-level similarity between items.

contains discrete ratings in a small rating scale, it is expected that the effect of the sharp boundary problem would be small and, as a result, using fuzzified ratings may not improve recommendation results. The Jester dataset, in contrast, contains continuous ratings in a larger rating scale. The sharp boundary problem is believed to be more significant than in the MovieLens dataset.

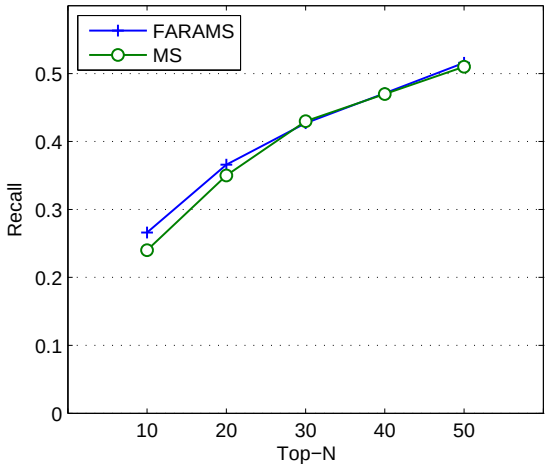

Figure 3.5: Recall rates achieved with and without using fuzzy association rules on the MovieLens dataset.

For the MovieLens dataset, we used the membership functions in MF(A) (Figure 2.2(a)). In Figure 3.5, MS indicates the results of the experiments using MS only, while FARAMS indicates the performance of our framework

using fuzzified ratings. As shown in the figure, FARAMS outperforms MS by 7.5% and 2.0% when N is equal to 10 and 20 respectively. This shows that hits appeared in higher ranks in our approach. As recommendable items are ranked according to the confidence values of their related rules, the results reveal that fuzzy confidence provides a better indicator of a rule's interestingness than does the classical confidence measure. For larger values of N, however, MS outperforms FARAMS by 1.3%, 1.8% and 0.1% respectively. This is in line with our expectation that the sharp boundary problem is small in the discrete ratings with a small rating scale, which therefore do not have much fuzziness.

The second set of experiments on the Jester dataset used the membership functions in MF(B) (Figure 2.2(b)). Figure 3.6 shows the results. As items in this dataset do not have any hierarchical category structure, recommendations were generated using only item-level association rules, denoted by AR in the figure. The performance of AR recorded for different Top-N values is compared to that of FAR, which uses fuzzy association rules without MS.

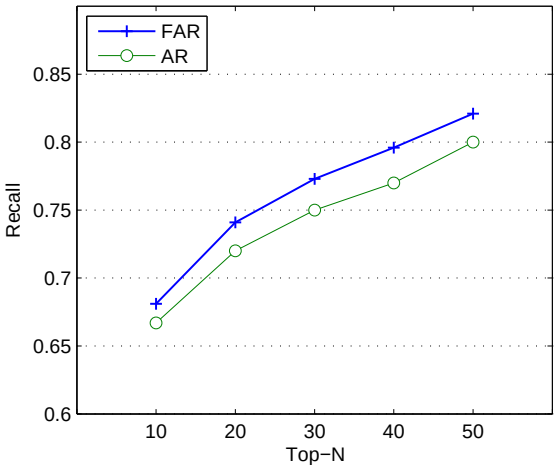

Figure 3.6: Recall rates achieved with and without using fuzzy association rules on the Jester dataset.

As can be seen in Figure 3.6, FAR outperforms AR for all Top-N values. The improvements in recall rates achieved are approximately 2.0%, 1.8%, 2.4%, 2.5% and 2.1% respectively for the Top-N values. This indicates that the sharp boundary problem is more obvious in datasets recorded in a finer-grained rating scale as compared to the previous experiment on the MovieLens dataset, and that the use of FAR helps address the sharp boundary problem.

Comparisons with related work

Lin et al. [89] tested their ASARM algorithm on the EachMovie dataset. As collaborative users (the training set), they used the first 1,000 users who have rated more than 100 movies. As target users (the test set), they used the first 100 users whose user IDs are greater than 70,000 (such that users in the training set and those in the test set do not overlap) and who have rated more than 100 movies. Although this experimental setup is small-scale and neglected the adverse effect of data sparseness on prediction quality, to ensure the comparability of their work and ours, we nonetheless tested our algorithm under similar conditions. Ratings in the dataset were fuzzified using the membership functions in MF(C) (Figure 2.2(c)). Table 3.6 shows the results.

Table 3.6: Performance of ASARM and FARAMS for the EachMovie dataset.

Algorithm	**Recall**	**Confidence Threshold**
ASARM - Item Association	0.226	0.9
FARAMS	0.269	0.65

As shown in Table 3.6, FARAMS produces results similar to those of ASARM, but with a lower confidence threshold. This is because our approach takes all transactions into account when computing the support and confidence values (in percentages) of rules. The ASARM algorithm, in contrast, counts only transactions containing the target item. This approach underestimates the support values of some items other than the target item, which in turns over-emphasizes the confidence values of rules, especially of those of less popular items in the dataset. In other words, the confidence value assigned to the same association rule may be much higher in ASARM than in our approach. This difference in the number of transactions used to determine support and confidence values of rules is why our approach uses a lower minimum confidence.

The MAR algorithm was tested on the MovieLens dataset as well as on the KDD dataset in [71]. As the KDD dataset does not contain ratings data, the comparison uses only the MovieLens dataset. Figure 3.7 shows the experimental results for the five Top-N values, with FARAMS outperforming MAR by 61.0%, 46.3%, 33.4%, 25.7% and 23.0% respectively. This shows that our approach is effective. Furthermore, the significant improvement achieved in Top-10 recommendation suggests that fuzzy confidence produces better rankings of

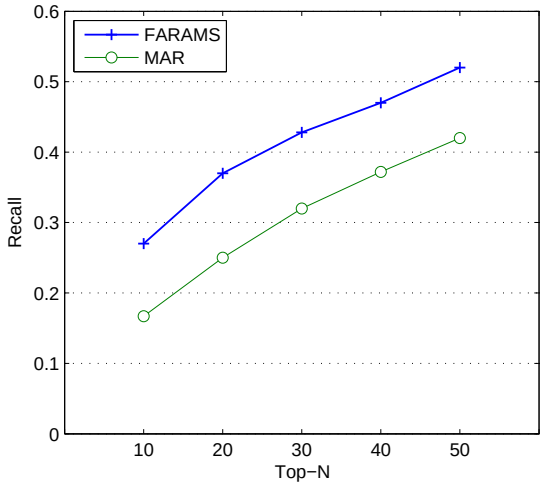

Figure 3.7: Performance of MAR and FARAMS for the Movielens dataset.

recommended items.

3.5 Summary

This section describes the proposed FARAMS framework for CF and its application of fuzzy association rule mining to address the sharp boundary problem in existing techniques. We also approached the problem of data sparseness in CF by taking advantage of multiple-level similarities that are implicit in the taxonomies of items. We presented and discussed the results of an evaluation of FARAMS. Results show that the use of FAR is more effective on datasets containing continuous ratings, and that FARAMS outperforms existing techniques in similar experimental settings.

As we mentioned in the Introduction section of this chapter, ARM techniques provide the flexibility to mine associations between content-related attributes (i.e. categories information in this work) and user ratings on items. The rule mining process of FARAMS, however, does not consider this and only mines rules from the same level of the given item taxonomy. We further explore this issue in the next chapter.

Chapter 4

Cold-start Recommendations by Cross-level Association Rule Mining

4.1 Introduction

The cold-start problem is a crucial shortcoming of CF, which generates personalized recommendations to users solely based on ratings data. Recall that the cold-start problem is an extreme form of data sparseness. It arises when no recommendations can be generated for items with no or very few ratings data. It is also known as the early-rater problem and the new item problem.

This chapter describes our proposed hybrid recommendation approach, developed based on FARAMS (Chapter 3), for addressing the cold-start problem. Our approach makes use of Cross-Level Association RulEs (CLARE)[8] to flexibly integrate content features of items and user ratings. CLARE operates on a preference model comprising both user-item and item-item relationships, and infers user preferences for items from the attributes they possess. The major feature of CLARE is the use of associations between a given item's attributes and other domain items, when no recommendations for that item can be generated using CF. We use an example to illustrate our idea. The CAR "*Movie A* $\rightarrow$ *Director:Woody Allen*" indicates that "*users who liked Movie A also liked movies directed by Woody Allen*". If there exists a new (cold-start) movie, *Movie Z*, directed by *Woody Allen*, we may recommend it to users who had liked *Movie A* previously.

[8]We use the acronym CAR to represent cross-level association rule, and CLARE as the name of our proposed recommendation approach.

The rest of this chapter is organized as follows. Chapter 4.2 describes related work on addressing the cold-start recommendation problem. Chapter 4.3 introduces our proposed preference model, based on which the motivation of our work is explained. Chapter 4.4 describes CLARE, the proposed cross-level association rule mining approach to cold-start recommendations. Chapter 4.5 discusses experimental results on CLARE and on comparisons with related work. Chapter 4.6 summarizes our contributions and findings, as well as outlines some limitations of CLARE for future work.

4.2 Existing Approaches to Cold-start Recommendations

We describe in this section existing approaches to cold-start recommendations. They include the aspect model [57, 58, 121, 140] and the naive filterbot algorithm [117].

4.2.1 The Aspect Model

Schein et al. [140] described the use of the aspect model for generating cold-start recommendations. The aspect model is a statistical latent class model, originally proposed for document indexing by Hofmann [57]. It associates word-document co-occurrence data with a set of latent variables, and was applied to user-item co-occurrence data for CF by Hofmann and Puzicha [58]. The overall idea of the aspect model is as follows. Given a set of users $u \in U$ and a set of items $i \in I$, an *observation* (u, i) corresponds to the co-occurrence of u and i. A latent class variable $z \in Z = \{z_1, z_2, ...z_m\}$ is associated with each (u, i) in the aspect model, which assumes that u and i are independent, conditioned on z [57, 121]. The probability $\mathrm{P}(u, i)$ is therefore defined as [58]:

$$\mathrm{P}(u, i) = \sum_{z \in Z} \mathrm{P}(z)\mathrm{P}(u|z)\mathrm{P}(i|z) \tag{4.1}$$

The model parameters are estimated from training data using the Expectation Maximization (EM) algorithm or its variants [57]. Recommendations to user u are made according to $\mathrm{P}(i|u) \propto \mathrm{P}(u, i)$. The higher the value of $\mathrm{P}(i|u)$, the more likely u will observe i.

Popescul et al. [121] proposed two extensions of the aspect model for hybrid content- and CF-based recommendations. The first extension is known as the *three-way aspect model*. It includes three-way co-occurrence data among users, items and item attributes $f \in F$. An observation in this aspect model, denoted by (u, i, f), corresponds to the event of user u observing item i with attribute f. The second extension, known as the *user-words aspect model*, discards the concept of items, and an observation (u, f) corresponds to the event of user u observing attribute f.

Schein et al. [140] applied the user-words aspect model [121] to cold-start recommendations. Their idea is to regard item attributes (actors of movies) as surrogates of items (movies). Specifically, they estimate P(u, f) from training data, and then "fold-in" a new movie, using the folding-in algorithm described in [57], out of the set of attributes of that movie. This is a hybrid recommendation approach capable of cold-start recommendations, because new items are recommended to user u based on P(u, f) estimated from training data.

4.2.2 The Naive Filterbot Algorithm

Park et al. [117] described the use of the *naive filterbot* algorithm for addressing the cold-start problem. The naive filterbot algorithm injects *pseudo users*, or *bots*, into a recommender system. The bots generate user ratings according to the attributes of the items or the users in the system. Such ratings were injected into the original user-item rating matrix in the system along with actual user ratings, thereby increases the density of the ratings matrix.

Note that the naive filterbot algorithm itself is not a recommendation algorithm. Instead, it injects ratings into the ratings matrix, on which CF algorithms are applied to generate recommendations. For instance, Park et al. used bots to augment ratings for the standard user-based and item-based algorithms [117]. They evaluated the effects of the bots on the cold-start item problem when the number of ratings on *relatively new* items increased gradually (from 2 to 40 ratings in their experiments). Our work, in contrast, is able to recommend items with *no ratings at all*.

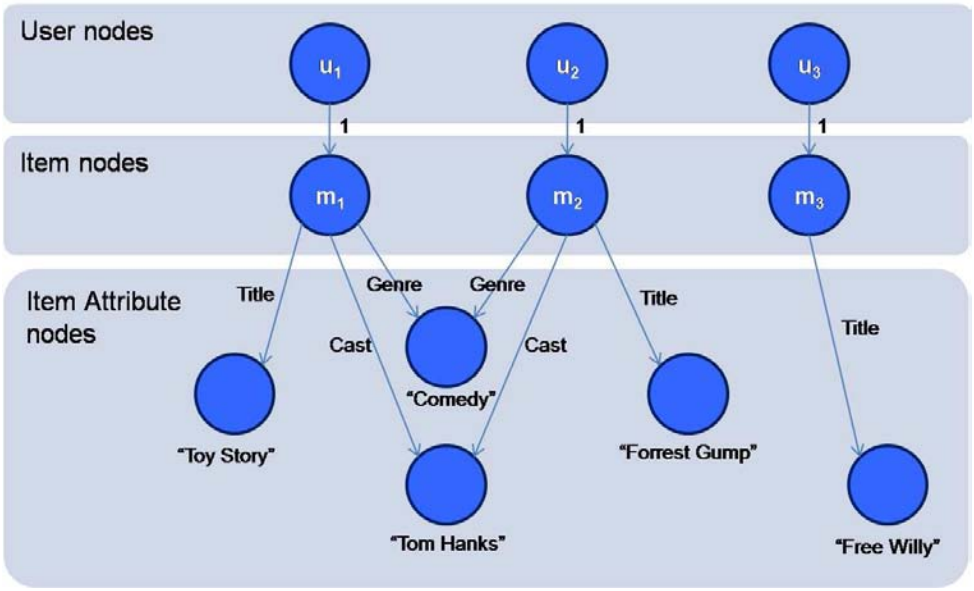

Figure 4.1: Illustration of the proposed preference model comprising user-item and item-item relationships.

4.3 Problem Description

This section introduces the proposed preference model comprising both user-item and item-item relationships for mining CARs, and explains the motivation of our work and some properties of the proposed model.

Figure 4.1 shows a simple example of the proposed preference model in a movie recommender system. It consists of three layers, namely the `User` layer, the `Item` layer, and the `Item Attribute` layer. An edge connecting a `User` node and an `Item` node means that the user has rated the item, and is labeled with a fuzzified rating in the range [0, 1] to reflect the degree to which the user had *liked* the item. An edge connecting an `Item` node and an `Item Attribute` node represents a particular attribute of the item. The label of such an edge describes the name of the attribute while the value of an `Item Attribute` node describes the value of the attribute.

There are two points to note on the preference model. The first point is that both positive and negative user preferences for items are available in an ARM-based recommendation framework, but the model only captures membership degrees obtained by user ratings with respect to the *Like* fuzzy set. This is because the goal of CF is to recommend to users items they may like. Besides,

Item	TID-list	Total Support
i_1	u_1	1
i_2	u_2, u_3	2
i_3	u_2, u_3	2

(a) Support of i_1, i_2 and i_3

Item	TID-list	Total Support
i_3	u_2, u_3	2
c_1	u_1, u_2, u_3	3

(b) Support of c_1 and i_3

Item	TID-list	Total Support
$\{i_3, c_1\}$	u_2, u_3	2

(c) Frequent itemset (*minSupp* = 1)

Association Rule	Support	Confidence
$i_3 \rightarrow c_1$	2/3 = 66.7%	2/2 = 100%

(d) Association rule

Figure 4.2: A motivating example

precedent work on ARM-based CF suggests that negative preferences are not useful for generating recommendations [89]. The second point is that in a real world application, the `Item Attribute` layer can contain multiple sub-layers (e.g. in [162]), but we use the model in Figure 4.1 for illustrations in this article for simplicity. `Item Attribute` sub-layers are further discussed in Chapter 4.6.

We now use an example to motivate the use of the proposed preference model and CARs for generating item recommendations. Figure 4.2(a) shows a transactional representation of the `Item` and `Item Attribute` nodes in Figure 4.1 in a *vertical TID-list* format [174]. The membership degrees in the example are all 1's and are therefore omitted in the figure for simplicity. Suppose the target item is i_1 and *minSupp* is 1, no item-level association rule can be mined because i_1 has not appeared with other items in any transaction. i_1 is therefore a *cold-start item*.

Our proposed approach for generating cold-start recommendations assumes that a transaction that supports an item also supports its attributes. This allows us to infer preferences for i_1 from its `Item Attribute` nodes. Suppose the attribute *Cast* is used for generating recommendations. Referring to Figure 4.1, both i_1 and i_2 have the `Item Attribute` node "Cast:Tom Hanks", denoted as c_1 in Figure 4.2. We can then obtain the transactions in Figure 4.2(b). The supporting transactions of c_1 are the union of the transactions supporting its parents (i_1 and i_2). Given the new information, the itemset $\{i_3, c_1\}$ satisfies *minSupp* (Figure 4.2(c)), and based on which the association rule "$i_3 \rightarrow c_1$" having a support of 66.7% and a confidence of 100% can be mined (Figure 4.2(d)). Our work attempts to use such a CAR to recommend the item $m1$, which was a cold-start item for which no recommendations could be made in a pure CF setting. For instance, if a user has liked i_3 previously, we may recommend i_1 to him/her given the association rule "$i_3 \rightarrow c_1$".

Based on the assumption that a transaction that supports an item also supports its attributes, the support values of `Item` and `Item Attribute` nodes in Figure 4.1 satisfy the following properties:

Property 1 *If an `Item` node is frequent, its `Item Attribute` nodes are also considered frequent, despite the different support thresholds that may be used for pruning items of different levels.*

Property 2 *For a frequent `Item Attribute` node, it is possible that none of its parents is frequent.*

This is because the support of an `Item Attribute` node is obtained from all of its parents. An example is the node "Cast:Tom Hanks" (c_1) in the described motivating example.

Property 3 *Based on **Property 2**, if an `Item` node is infrequent, it is still possible for all or some of its `Item Attribute` nodes to be frequent.*

Examples are the nodes i_1 and i_2.

Lower-level items (`Item Attribute` nodes) in the proposed preference model are likely to have more support than higher-level items (`Item` nodes) as opposed to the property of is-a hierarchies. The assumption about is-hierarchies that only descendents of frequent items are examined does not apply to our model. Given **Property 3** above, it is possible for `Item Attribute` nodes of an infrequent `Item` node to be frequent. We therefore examine `Item Attribute` nodes of an `Item` node when no item-level association rule for

it is available. Different support thresholds are still required to prune nodes at different level of the hierarchy as discussed in the next section.

4.3.1 Data Representation

The proposed preference model consists of `User`, `Item` and `Item Attribute` nodes as noted. We denote the set of `User` nodes as $u \in U = \{u_1, u_2, u_3, ..., u_m\}$, the set of `Item` nodes as $i \in I = \{i_1, i_2, i_3, ..., i_n\}$, and the set of `Item Attribute` nodes (attribute-value pairs) as $f \in F = \{f_1, f_2, f_3, ..., f_n\}$. A directed edge connecting a `User` u and an `Item` node i is a triple $\langle u, i, m_{Like}(r_{u,i})\rangle$. $r_{u,i}$ is the original rating u have given i. It is fuzzified using the function m_{Like}, the user-specified membership function of the *Like* fuzzy set as noted. A directed edge connecting an `Item` i and an `Item Attribute` node f is a pair $\langle i, f\rangle$, and the set of $\{\langle i, f\rangle\}$ forms domain knowledge about the items. We generated the set $\{\langle u, f, m_{Like}(r_{u,f})\rangle\}$ out of $\{\langle u, i, m_{Like}(r_{u,i})\rangle\}$ and $\{\langle i, f\rangle\}$ for mining CARs. The rating $r_{u,f}$ is the average rating u have given the set of $\{i\}$ containing f. Note that we described in this paragraph two sets of preference data. The first set, $\{\langle u, i, m_{Like}(r_{u,i})\rangle\}$, represents user preferences for items, while the second set, $\{\langle u, f, m_{Like}(r_{u,f})\rangle\}$, represents those for item attributes.

4.4 CLARE: Cold-start Recommendations by CAR Mining

CLARE was developed based on the FARAMS CF framework (Chapter 3), which produces collaborative recommendations using Fuzzy Association Rules And Multiple-level Similarity between items in their taxonomies (is-a hierarchies). CLARE works in three major steps, namely data preprocessing, mining user preferences, and generating recommendations. The steps for data preprocessing and generating recommendations are similar to those involved in FARAMS. We therefore describe these steps briefly, and focus on the mining user preferences step which mines CARs for cold-start items.

Table 4.1: Important notations used to represent user preference data in CLARE.

Notation	Description
u	A `User` node.
i	An `Item` node.
f	An `Item Attribute` node (attribute-value pair).
$\{i, f\}$	Domain knowledge (relationships between Item and Item Attribute nodes).
TID_i	TID-list representation of user preferences for items.
TID_f	TID-list representation of user preferences for item attributes.

4.4.1 Data Preprocessing

This step preprocesses users' preference data for mining. It involves three key tasks. The first task is to fuzzify user ratings to reflect the degree to which users liked the rated items. Note that each item i in FARAMS can be expanded into, for instance, $\langle i, L\rangle$ and $\langle i, D\rangle$ pairs for modeling positive and negative preferences for item i. Since our preference model only captures positive preferences, we omitted the fuzzy set information when describing the expanded item $\langle i, L\rangle$ for simplicity. In this chapter, hence, a rule "$i \rightarrow t$" should be interpreted as "if a user liked item i, then (s)he also liked item t".

The second task is to compute $\{\langle u, f, m_{Like}(r_{u,f})\rangle\}$, the set of user preferences for item attributes, based on $\{\langle u, i, m_{Like}(r_{u,i})\rangle\}$ and $\{\langle i, f\rangle\}$ for mining CARs. The rating $r_{u,f}$ is the average rating u have given the set of $\{i\}$ containing f as aforementioned.

The third task is to generate transactional representations of the preference data to facilitate the rule mining process. As noted, our proposed model consists of two sets of preference data, denoted as $\{\langle u, i, m_{Like}(r_{u,i})\rangle\}$ and $\{\langle u, f, m_{Like}(r_{u,f})\rangle\}$. These sets of data are transformed into vertical TID-lists as described in Chapter 3.3.1. We refer to the vertical TID-list representations of user preferences for items and those for item attributes as TID_i and TID_f respectively hereafter. Table 4.1 summarizes the important notations we described in this subsection.

4.4.2 Mining Association Rules

CLARE extended the mining algorithm of FARAMS for mining CARs for cold-start items. Given a target item $t \in I$, CLARE starts mining item-level association

rules for it in the form of "$\{i\} \rightarrow t$", where $t \notin \{i\}$, using the mining algorithm described in Chapter 3.3.2. If no item-level association rule can be mined, meaning that item t is a cold-start item that is not associated with other items in the system, CLARE attempts to infer preferences for t from its attributes. The choice of attribute types used for mining is domain dependent and can be specified by users. In what follows, we briefly revisit the mining algorithm of FARAMS, and then describe how we extended the algorithm for mining CARs for cold-start items.

Overview of the mining algorithm

The mining algorithm of FARAMS is an Apriori-like, adaptive-support algorithm that mines association rules for one target item at a time [5, 89, 80]. We summarize the key tasks of the mining algorithm as follows:

1. The algorithm iteratively generates frequent κ-itemsets, which are sets of κ items satisfying the *minSupp* constraint, using the Apriori-gen function [5]. As FARAMS aims at mining rules for a given target item t at a time, it only retains frequent itemsets containing the t for the next task.

2. The algorithm generates association rules from the frequent κ-itemsets in the form of "$\{i\} \rightarrow t$". Association rules that satisfy all user-specified interestingness constraints, such as minimum confidence and minimum correlation, are considered interesting.

The mining algorithm of FARAMS adopts the adaptive-support strategy proposed in [89] as noted. It automatically adjusts the *minSupp* used for mining, so that the number of interesting rules is between *minNumRules* and *maxNumRules*, unless fewer than *minNumRules* rules exist for the given interestingness constraints.

Mining CARs for cold-start items

CLARE adapted the mining algorithm of FARAMS for mining CARs for cold-start items by extending the concept of a single target item to a *group of attributes* of the given target item. Only $\{i\}$, `Item` nodes, are considered when the rule mining process begins, and the group of target "attributes" actually contains the target item t only. If t is found to be a cold-start item, its `Item Attribute`

Algorithm 4.4.1 *The **minCAR** algorithm (overview)*

Input: t, TID_i, TID_f

Output: R_t (the set of CARs mined for t)

Steps:

1 $\{$*candidateAttr*, $TID_{fc}\} \leftarrow$ attributes and their associated TID-lists of t;

2 $\{$*targetAttr*, $TID_{ft}\} \leftarrow$ frequent itemsets and their TID-lists mined from $\{$*candidateAttr*, $T_{fc}\}$;

3 $R_{ft} \leftarrow$ interesting CARs containing *targetAttr* in the rule heads;

4 $R_t \leftarrow$ replace the rule head of each rule $r \in R_{ft}$ by *t*;

5 **return** R_t;

---

nodes are then used for mining, and the group of attributes becomes the group of interesting attributes of the target item. Algorithm 4.4.1 describes the steps for mining CARs for cold-start items in pseudocode.

The five steps of Algorithm 4.4.1 are outlined as follows.

- Step 1 simply retrieves the target item's list of attributes from {i, f}, and the TID-lists of the attributes from TID_f. The list of attributes and their TID-lists are denoted by *candidateAttr* and TID_{fc} respectively.

- Step 2 mines frequent itemsets from *candidateAttr* using the Apriori-gen function [5]. The maximally frequent itemsets, denoted as *targetAttr*, and their TID-lists (TID_{ft}) are retained for mining CARs for cold-start items. Note that each itemset in *targetAttr* may contain more than 1 item from *candidateAttr*.

  The *minSupp* for mining *targetAttr* is determined using the *attribute-based specification* proposed in [161]. The attribute-based specification adopts the average support of itemsets containing attribute-value pairs as the *minSupp*. For example, if *States* and *Gender* are two attributes in a table (in a transactional database), the minimum support of an itemset containing a state code and a gender is then $\frac{|DB|}{50} * \frac{|DB|}{2}$, where $|DB|$ is the number of records in the table, and 50 and 2 are the numbers of possible values for the *States* and the *Gender* attributes respectively [161].

  We adopted an average support for mining *targetAttr* because different attributes can have very different generalities. In the MovieLens 100k

dataset, for example, the *Director* attribute has more than 1,000 distinct values, whereas *Genre* only has 19. Each value of *Genre* should therefore appear more frequently than that of *Director*. This means that if the adaptive-support strategy is used, the set of frequent attributes would likely be dominated by those having a small number of distinct values.

- Step 3 mines CARs containing *targetAttr* in the rule heads using the mining algorithm described in the previous subsection. Note that the Apriori-gen function generates candidate κ-itemsets, where κ is the number of items in each itemset, by joining two frequent (κ-1)-itemsets [5]. When mining CARs, however, we do not join 2 itemsets if both of them are from the *targetAttr* because this has already been done in the process of generating *targetAttr* from *candidateAttr*.

- Step 4 is a post-processing step that replaces the rule head, which was an itemset in *targetAttr*, of each CAR by the target item t. In other words, the rules mined in the previous step are used as if they were mined for t.

- Step 5 returns the set of interesting rules containing t in the rule heads, denoted by R_t. The rules are stored in a *rule base* for generating recommendations.

4.4.3 Generating Recommendations

This step generates recommendations for users based on their known preferences and the rule base. This consists of three tasks. Firstly, when an active user a requests recommendations, we determine rules that are *relevant* to user a's known preferences. A rule "$i \rightarrow t$" is considered relevant if user a had liked item i previously, but has not yet rated item t. In such case, item t is considered a *recommendable item* for user a.

Secondly, we assign to each recommendable item t a predicted preference value, determined by interestingness scores of the relevant rules containing item t in their rule heads. Such interestingness scores are in general indicators of the quality of the rules, for example, their support, confidence and correlation values [89, 71]. If a recommendable item t appears in the heads of more than one relevant rule, we sum up the interestingness scores of such rules to determine user u's predicted preference value for item t [71]. The higher the predicted

preference value obtained by a recommendable item, the more likely that the active user will like the item.

Finally, we recommend the N items with the highest predicted preference values to user a (Top-N recommendation).

4.5 Experimental Results

This section presents experimental results on validating the ability of CLARE to provide cold-start recommendations. We first describe the experimental settings and then discuss the results.

4.5.1 Dataset

We evaluated the performance of CLARE using the MovieLens 100k dataset, which contains 100,000 ratings for 1,682 movies by 943 users. Ratings were recorded in an integer 5-point scale (1 to 5). We normalized the ratings by a simple division of 5, the maximum rating in the dataset:

$$m_{Like}(r_{u,i}) = \frac{r_{u,i} - \min(s)}{\max(s) - \min(s)} \tag{4.2}$$

where s represents the rating scale used in the dataset. We then considered the normalized ratings to be the membership degrees of the original ratings with respect to the *Like* fuzzy set. Note that it is possible to learn such membership functions from the dataset for modeling user preferences (e.g. [59]). As this is not the focus of this study, we only adopted a simple membership function in our experiments.

We used four types of attributes as `Item Attribute` nodes for mining CARs. They include *Genre*, *Cast*, *Director* and *Plot Keywords* (simply referred to as *Plot* hereafter). Genres of the movies were available in the original MovieLens dataset. The cast and directors of the movies were generously provided by the GroupLens team for our study. Note that the cast of a movie only includes the first four actors/actresses listed on the MovieLens website. We collected plot keywords of the movies from the Internet Movie Database (IMDb), and weighted the keywords by the TF-IDF feature weighting scheme, described in Eqs. (2.22)-(2.24). Recall that TF-IDF is a statistical measurement indicating the importance a feature with respect to a document and a document collection.

Table 4.2: Averaged statistics about training and test sets.

Description		% of cold-start items (n)		
		10	**20**	**30**
Total no. of ratings	- Training set	83,278	67,631	49,358
	- Test set	16,722	32,369	50,642
No. of positive ratings	- Training set	48,824	42,717	36,352
	- Test set	6,551	12,658	19,023
No. of positive ratings per user	- Training set	52	45	39
	- Test set	7	13	20

In this work, a movie takes the role of a document, while a keyword takes the role of a feature. We only considered the 15 most important keywords, as determined by TF-IDF, of each movie in this study to speed up the training process, but this constraint can be relaxed.

4.5.2 Method and Evaluation Metrics

We consider a rating given by user u on item i to be positive, that is, user u liked item i, if $r_{u,i} \geq 4$ for performance evaluation purpose. We randomly selected as cold-start items $n\%$ (n = {10, 20, 30}) of movies from all movies for which recommendation could be successful (1,447 movies having at least one positive rating in the dataset). We created ten random samples for each value of n. All results reported in this article were averages of the ten samples. In each experiment, the test set consists of *all* ratings on the "cold-start" items, while the training set consists of the ratings on the remaining movies. Table 4.2 reports averaged statistics about the training sets and test sets.

We generated Top-10 recommendations for active users who had at least one positive rating in the test set. Recommendation of an item i to an active user a is considered correct if the tuple $\langle a, i\rangle$ exists in the test set and $r_{a,i} \geq 4$. It is considered incorrect if $r_{a,i} < 4$, or if $\langle a, i\rangle$ does not exist in the test set at all. We evaluated the recommendation accuracy of CLARE based on four commonly used metrics for evaluating Top-N recommendations. They include precision, recall, F_1 (f-measure) and rank score. Definitions of these metrics are given in Chapter 2.2.5.

Note that rank score consists of two adjustable parameters, which are h, the

viewing halflife, and $\delta(u, i_j)$, the contribution of a correct recommendation to the overall utility of the ranked recommendation list. We set h = 10, and $\delta(u, i_j)$ = $m_{Like}(r_{u,i})$. This means correctly recommending an item rated as 5, for example, in user u' s testset is considered more desirable than recommending an item rated as 4.

We also reported the coverage rates of CLARE to demonstrate its ability to recommend cold-start items, which are not recommendable at all in a pure CF setting.

4.5.3 Parameters

We now report the values of two parameters we adopted in the experiments. These parameters can be flexibly configured in CLARE for a specific application. The first parameter is the desired range [*minNumRules, maxNumRules*] for the number of rules mined for a target item. We chose the range [10, 40] based on a set of preliminary experiments. The second parameter is the maximum rule length, which is the maximum number of items in a rule's body [89]. We adopted a maximum rule length of 1 based on our previous experiments with FARAMS (Chapter 3.4.3).

4.5.4 Evaluation of CLARE

This subsection discusses experimental results on cold-start recommendations produced by CLARE. We also implemented a baseline algorithm for benchmarking. A commonly used baseline algorithm in CF is known as *popularity* (POP), which recommends to an active user the Top-N most popular unseen items in the training set [14, 43, 63]. POP, however, cannot be used for this study because we aim at recommending cold-start items that *did not* appear in the training set. We therefore implemented a random recommender, which recommends cold-start items to the active user randomly, as the baseline.

In what follows, we first detail the results of two experiments. The first experiment aims at determining an appropriate measure for scoring recommendable items. The second experiment evaluates the recommendation quality and coverage produced by CLARE using different item attributes for generating cold-start recommendations. We further discuss the results of this experiment, focusing on the effects of the varying percentages of cold-start items in the experiments.

When we describe results as significant in the subsequent discussions, we mean so statistically based on the Wilcoxon signed-rank test (using a 0.05 significance level), a non-parametric version of the popular paired t-test.

Predicting preferences for recommendable items

This experiment explores various measures for predicting preferences for recommendable items. As described in Chapter 4.4.3, the predicted preference for a recommendable item i is the sum of the interestingness scores of all relevant rules containing i in their heads. The score of a relevant rule is given by some of the rule's interestingness measures. We experimented with the three most popular measures, namely fuzzy support (*FS*), fuzzy confidence (*FC*) and correlation (*CORR*), as well as combinations of them. The computations of *FS*, *FC* and *CORR* are given in Eq. (2.16)-(2.18). We combined a set of interestingness measures M by taking the harmonic mean of the values in M. The harmonic mean of M, denoted as $H(M)$, is defined as:

$$H(M) = \frac{|M|}{\sum_{m \in M} \frac{1}{m}} \tag{4.3}$$

The harmonic mean of a list of values tends strongly towards the smallest value in the list. It therefore has the effect of penalizing CARs that are particularly weak in a certain aspect, such as a CAR having a high confidence but a low correlation.

Figure 4.3 reports the recommendation quality produced using different sets of M at different values of n, using *Plot* for mining CARs. In the figure, M = {*FS, FC*} means that H(*FS, FC*) of a rule is used for predicting the active user's preference for the recommendable item in the rule's head. We do not discuss coverage in this experiment as varying M only affects the ranking of recommendable items for an active user.

Figure 4.3 shows that CLARE always outperforms the baseline recommender. This is not surprising because CLARE provides personalized recommendations to users based on their known preferences in the training set, whereas the baseline does not. We therefore focus on the behavior of CLARE in the subsequent discussions.

The setting M = {*FC, CORR*} always yields the best results. However, results produced using M = {*CORR*} alone are not significantly different from

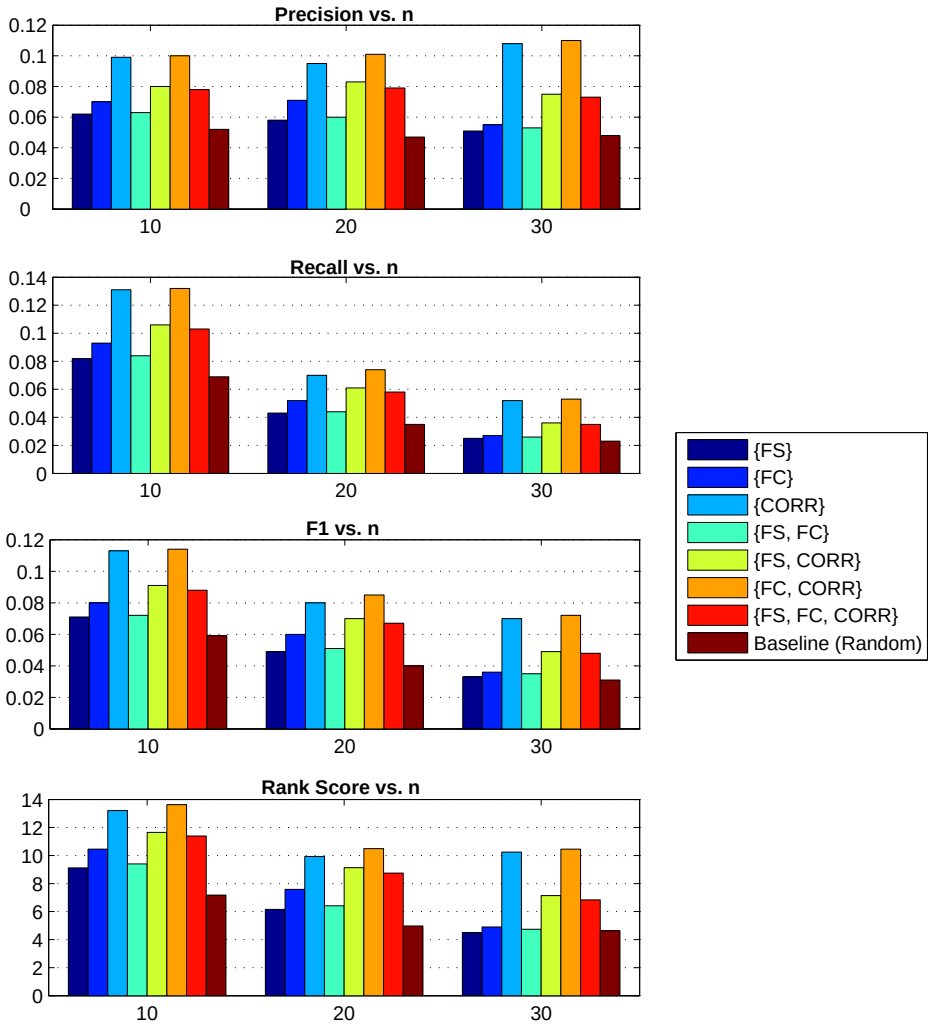

Figure 4.3: Recommendation quality produced using different interestingness measures (M) for predicting preferences for recommendable items, and *Plot* as attribute for mining CARs.

the best except for rank scores recorded at n = 20 and 30. These seem to suggest that correlation is an important interestingness measure in CLARE. Recall that CLARE infers preferences for cold-start items from the attributes they possess. Our results therefore suggest that we successfully addressed the cold-start problem by recommending cold-start items having attributes that are highly correlated with the items that the active users had liked previously.

Using $M = \{FS\}$ only outperforms the baseline recommender slightly, and produces the worst results among all other interestingness measures. Further, adding $\{FS\}$ to $\{CORR\}$ and to $\{FC\}$ impaired the performance achieved using the two individual measures greatly. For example, adding $\{FS\}$ to $\{CORR\}$ lowered the values of precision, recall, F_1 and rank score by approximately 30% at n = 30. We therefore conclude that the support of a rule does not help predicting preferences for the recommendable item in the rule's head. This is explainable in the context of a CF-based recommender system. The idea of CF is to recommend to an active user items liked by other, *similar* users. Such similarity can be derived even from users whose tastes deviate greatly from those of the majority of users. Similarly, the support of a rule indicates its statistical importance with respect to the entire database. A rule with high support means that a large amount of users had liked all items in the rule's head and body. A rule with low support, however, may still produce a good recommendation for a particular user, given that the user has liked the item in the rule's body.

Given the results in Figure 4.3, we adopted $M = \{FC, CORR\}$ in the subsequent experiments.

Effects of item attributes

This experiment evaluates the performance of CLARE when using different item attributes for mining CARs. The choice of item attributes used for mining is domain dependent. We experimented with four types of item attributes, including *Genre*, *Cast*, *Director*, and *Plot*, when applying CLARE to the movie domain. Table 4.3 reports the characteristics of these attribute types in our dataset. In the table, A denotes a specific attribute type. The function $N(f, A)$ returns the number of movies possessing the value f of the attribute A. F_A, where $F_A = \{f|N(f, A) > 0\}$, denotes the set of distinct values of the attribute A. $F_{N(f,A)>1}$ contains values in F_A that are possessed by more than one movie. $\overline{N(f, A)}$ is the average number of appearances of all attribute values

Table 4.3: Statistics about the various attributes.

Attribute (A)	$\lvert F_A \rvert$	$\lvert F_{N(f,A)>1} \rvert$	$\overline{N(f,A)}$	**Movies with** $F_{N(f,A)>1}$
Genre	18 *	18	157	99.94%
Cast	3,765	1,165	1.8	88.74%
Director	1,073	337	1.6	57.26%
Plot	5,521	5,298	7.8	98.17%

*The original MovieLens dataset has 19 genres, including the genre "unknown". We, however, did not consider "unknown" to be a meaningful genre that can characterize movies.

of A. *Movies with* $F_{N(f,A)>1}$ denotes the percentage of movies in the dataset that possess at least one value in $F_{N(f,A)>1}$.

Table 4.3 shows that the attributes have very different characteristics. For instance, *Genre* only has 18 distinct values in the entire dataset, and the average number of movies that belong to a particular genre is 157. On the contrary, *Director* is the most specific attribute. Our dataset contains a total of 1,073 directors, but only around 30% of them directed more than one movie. Interesting, the $\overline{N(f,A)}$ values of *Director* and *Cast* are very close to each other. Each value of *Director* appeared 1.6 times in the dataset on average, while each value of *Cast* appeared 1.8 times. However, 88.74% of the movies have actors or actresses who acted in more than one movie, as compared to the 57.26% for directors.

Given the varying characteristics of the attributes, we expected to have the following findings in this experiment:

1. Using *Genre* for generating cold-start recommendations should produce the worst recommendation quality but a high coverage due to its high generality.

2. Using *Director* for mining CARs might also produce unsatisfactory results, in terms of both recommendation quality and coverage. It is because only around 30% of the directors could be used for mining CARs for cold-start items.

3. *Plot* should perform well in general because it seems to have reasonable coverage of and generalities for characterizing the movies.

We now report the results of this experiment. Figure 4.4 shows the recom-

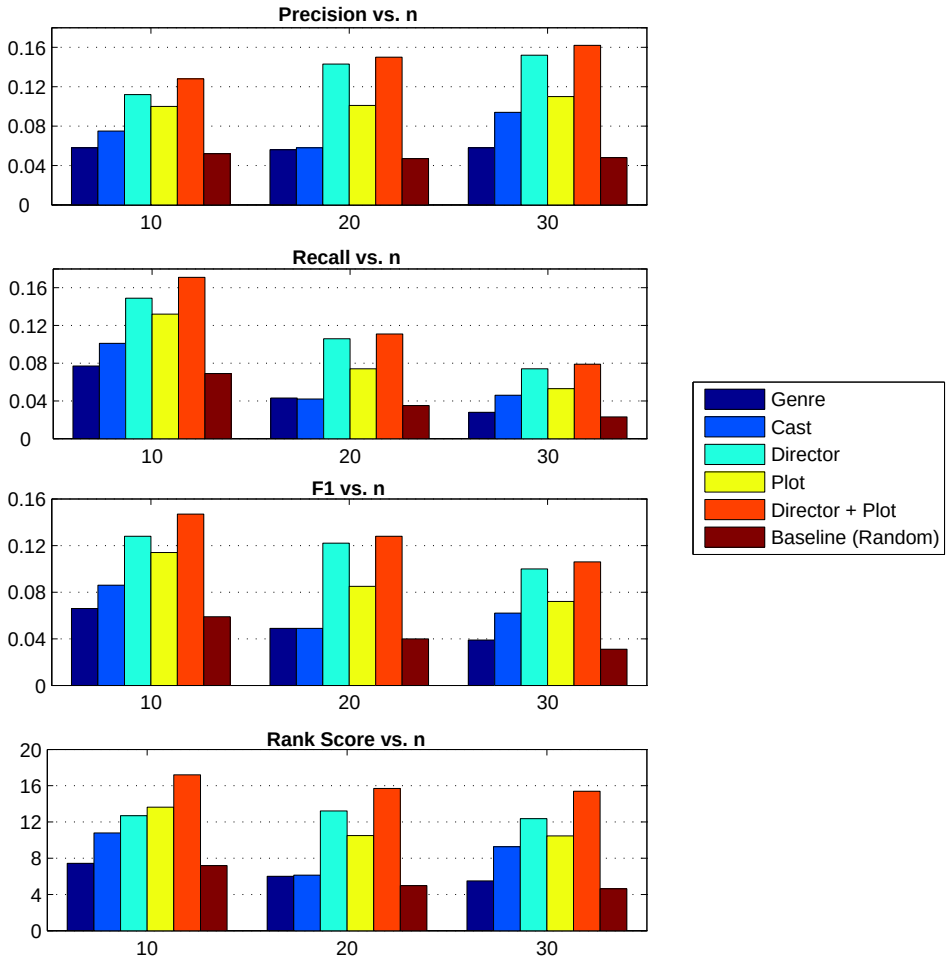

Figure 4.4: Recommendation quality produced using different attributes for mining CARs, and H(*FC, CORR*) for predicting preferences for recommendable items.

mendation quality produced by CLARE when using the different attributes for mining CARs. Figure 4.5 reports the coverage of cold-start items. It shows the percentages of cold-start items for which CARs could be mined using different attributes and at different values of n.

Figures 4.4 and 4.5 show that using different attributes for mining CARs produces varying results, because CLARE infers preferences for cold-start items from the attribute values they share with other items. In what follows, we summarize the results of this experiment, and explain the results with respect to the characteristics of the various attributes in our dataset.

Using *Genre* in CLARE slightly outperforms the baseline recommender, and

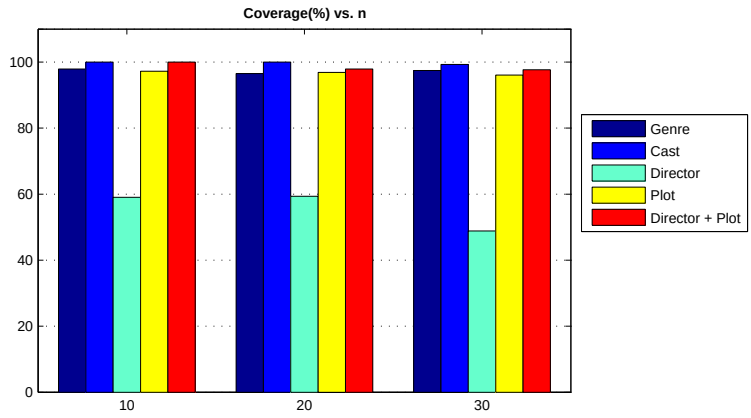

Figure 4.5: Coverage of cold-start items achieved using different item attributes for mining.

produces the worst recommendation quality as compared to other attribute types as expected (Figure 4.4). This should be a result of the high generality of the *Genre* attribute in the dataset. *Genre* only has 18 distinct values, and the average number of movies that belong to a particular genre is 157 as noted. Apparently, *Genre* is so general that it fails to capture the characteristics of the movies, resulting in the poor recommendation quality. The high coverage it produces is reasonable as almost all cold-start movies have genres that had been positively rated in the training sets.

Using *Director* produces mixed results. Interestingly, it gives the best recommendation quality but the worst coverage among all individual attribute types (except for the "*Director* + *Plot*" setting discussed next). These can also be explained with respect to the characteristics of *Director*, which is the most specific attribute type. Our dataset contains a total of 1,073 directors, but almost 70% of them directed only one movie. Consequently, these directors could not be used for inferring preferences for cold-start movies, resulting in the poor coverage rates. The high recommendation quality suggests that the more specific an attribute type, the more accurate preference prediction it can facilitate for cold-start items.

The poor coverage resulted from the high specificity of *Director* can be remedied by combining multiple attributes for mining CARs. We used *Plot*, which gives the second best recommendation quality and slightly-less-than-perfect coverage, in addition to *Director* for generating cold-start recommen-

dations as an example. The combined setting, denoted by *Director + Plot* in Figures 4.4 and 4.5, boosts recommendation quality to the best, and coverage rates to 100%, 97.9% and 97.7% at n = 10, 20 and 30 respectively. This example demonstrates that we can achieve good recommendation quality and coverage by choosing the appropriate attribute types for characterizing domain items.

To conclude, varying the choice of attributes produces quite different results due to the different characteristics of the attributes. CLARE can achieve good recommendation quality and high coverage by choosing the appropriate attributes for characterizing domain items. Given the results of this experiment, we focus on the behavior of CLARE using *Director + Plot* as attribute types in the subsequent discussions unless otherwise stated.

Discussions on results with respect to the percentage of cold-start items (n)

This subsection discusses the results of the previous experiment with respect to n, the percentage of cold-start items. The following paragraphs detail three observations we made from Figures 4.4 and 4.5.

The first observation is that recall, F_1 and rank score decline consistently as n increases, meaning that generating recommendations becomes more difficult when there are more cold-start items in the system. This decline might also be caused by the decrease in the number of known ratings of users in the training set (Table 4.2). More specifically, it is generally acknowledged that CF works better when more known preferences about users are available. We illustrate this with the help of Figure 4.6. We compared the overall recommendation quality of CLARE achieved for all users to that achieved for users who had, for example, at least 20 known ratings in the training set. Figure 4.6 shows that the precision, recall, F_1 and rank score achieved for users having at least 20 known ratings are well-above those achieved for all test users at all values of n. This might also suggest that CLARE's performance would improve over time for an active user as (s)he rates more items in the system.

The second observation is that CLARE produces better recall than precision at n = 10, but vice versa at n = 20 and 30. Related to this, as n becomes larger, precision improves but recall declines in general. These could be explained with respect to the average numbers of rating per user in the test sets [141]. We generated Top-10 recommendations for each user in an experiment as aforementioned. Referring to Table 4.2, each user has on average 7 ratings in the test set when n

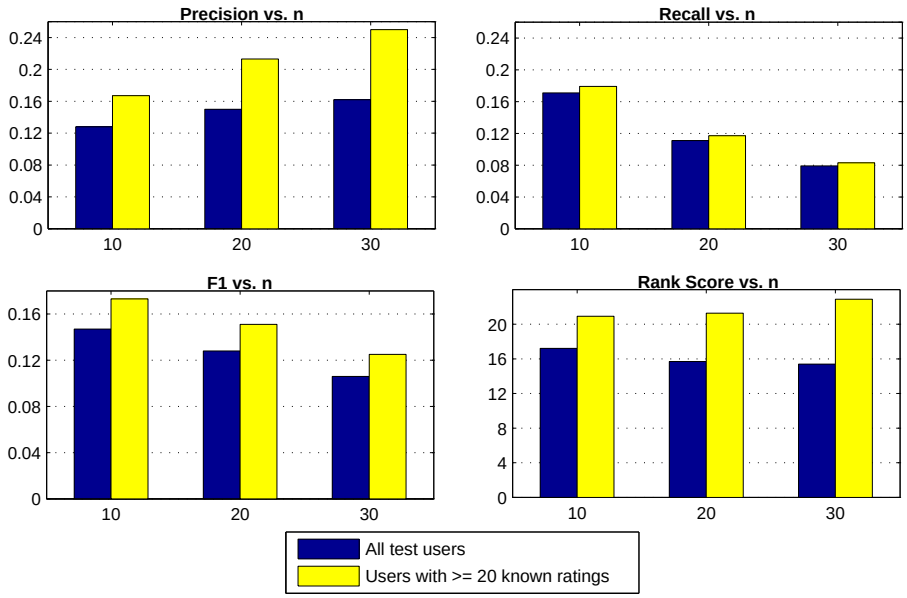

Figure 4.6: Comparison between recommendation quality achieved for all test users, and that for users having at least 20 known ratings in the training set. CARs were mined using *Director* + *Plot* as attributes.

= 10. Recommending 10 movies to a test user with 7 ratings would therefore produce at least 3 false positives, resulting in lowered precision. When n = 20 and 30, the average numbers of ratings per user in the test sets are 13 and 20 respectively. Similarly, recommending 10 movies to test users having 13 and 20 ratings would respectively produce at least 3 and 10 false negatives, resulting in lowered recall. A possible solution to address this problem is to recommend to user a only $|P_a|$ items, where $|P_a|$ is the number of items user a liked in the test set. However, this is obviously infeasible in practice as $|P_a|$ would not be known until a rated *all* items in the system.

The third observation is that the coverage of CLARE does not show any specific correlation with the value of n. This is because CLARE is able to infer preferences for cold-start items as long as there exist common attributes between those items and items in the training set.

Remarks on precision and recall

Herlocker et al. [55] suggested that precision and recall are biased and should not be interpreted as absolute measures. We also mentioned in the previous

subsection that recommending a fixed number of items to test users might produce false positives and false negatives that could adversely affect precision and recall. We nonetheless adopted them in our study for three reasons. Firstly, as discussed in Chapter 2.2.5, they do not take into consideration the value of TN (false negatives), which is likely to be extremely large in the Top-N recommendation task. Secondly, they are easy to interpret metrics that can fully support comparative studies between different algorithms across different experimental settings. Lastly, they sufficiently support our study, which focuses on recommending a small number of items to users. For these reasons, we decided to adopt precision and recall in this study, rather than other decision support metrics, such as CROC [141] and classification accuracy [99].

4.5.5 Comparisons with Related Work

We described in Chapter 4.2 two existing approaches to cold-start recommendations, namely the aspect model and the naive filterbot algorithm. The aspect model proposed by Schein et al. [140] is one of the most noticeable work focusing on the cold-start problem. However, we were unable to conduct a systematic comparison between their work and ours because they did not report evaluation results on the Top-N recommendation task, which is equivalent to the rating prediction task described in their paper [140].

We were also unable to empirically compare CLARE and the naive filterbot algorithm by Park et al. [117]. It is because the naive filterbot algorithm itself is not a recommendation algorithm. Instead, it injects ratings into the ratings matrix, on which CF algorithms are applied to generate recommendations.

Although we were unable to compare CLARE with the two related approaches, we identified and implemented two other possible algorithms for generating cold-start recommendations. The first algorithm, developed based on FARAMS, makes use of Multiple-level Similarity (MS) between items in item taxonomies, and is referred to as the MS-based recommendation algorithm in the subsequent discussions. The second algorithm is a pure content-based algorithm that recommends cold-start items based on their attributes. We describe in the following the implementation of the two algorithms, which we compared empirically against CLARE.

MS-based recommendation algorithm

We developed the MS-based recommendation algorithm, which extends the use of Multiple-level Similarity (MS) between items in the MAR [71] algorithm and in FARAMS, for generating cold-start recommendations. This was done in three main steps:

1. Given a taxonomy of items, which is the relationship between movies and their genres (G) in the movie domain, we mined MARs between genres from the dataset. We set the desired range of the number of rules to be mined to [1, $|G|$-1] following the settings in [80]. The set of MARs mined in this step is denoted by R_g.

2. We computed the fuzzified average rating each user u has given each genre g, denoted by $u[g]$, based on his/her known ratings in the training set.

3. Given an active user a, the rules mined in (1), and user a's preferences for genres determined in (2), we generated cold-start recommendations for a using Algorithm. 4.5.1.

Figure 4.7 compares the recommendation quality of the MS-based recommendation algorithm and CLARE. Experiments were performed under identical settings. The score of a rule r, denoted as $r.score$ in Algorithm 4.5.1 (Step 5), was determined using H(*FC, CORR*). The MS-based algorithm mines MARs from taxonomies, which are relationships between movies and their genres as noted. Recall that *Genre* fails to capture the characteristics of movies and therefore produces the worst performance among all attribute types in CLARE (Figure 4.4). We nonetheless conducted the comparison based on the performance of CLARE achieved using *Genre* as attribute to conform to the notion of "taxonomy" in MS-based recommendation. This also facilitates a fair comparison between the two algorithms.

CLARE outperforms the MS-based algorithm at all values of n. This seems to suggest that CARs between the attributes of cold-start items and other items in the dataset can better predict preferences for the cold-start items. Further, we point out that CLARE performs significantly better when using more specific item attributes, such as *Director* and *Plot* of movies, for recommending cold-start items as noted.

Algorithm 4.5.1 *The MS-based recommendation algorithm for recommending cold-start items to an active user a.*

Input: The active user a's preferences for genres ($a[g]$), N, and the set of MARs between movie genres (R_g)

Output: I_r, the set of recommended items for user a

Steps:

1. **for** each cold-start item i **do**
2. $s_i \leftarrow 0$;
3. **for** each genre g_n of i **do**
4. **if** there exists a rule $r = g_m \rightarrow g_n$ in R_g **then**
5. $s_i \leftarrow s_i + r.score * a[g_m]$;
6. **end if**
7. **end for**
8. **end for**
9. $P_a \leftarrow N$ items having the highest s_i;
10. **return** P_a;

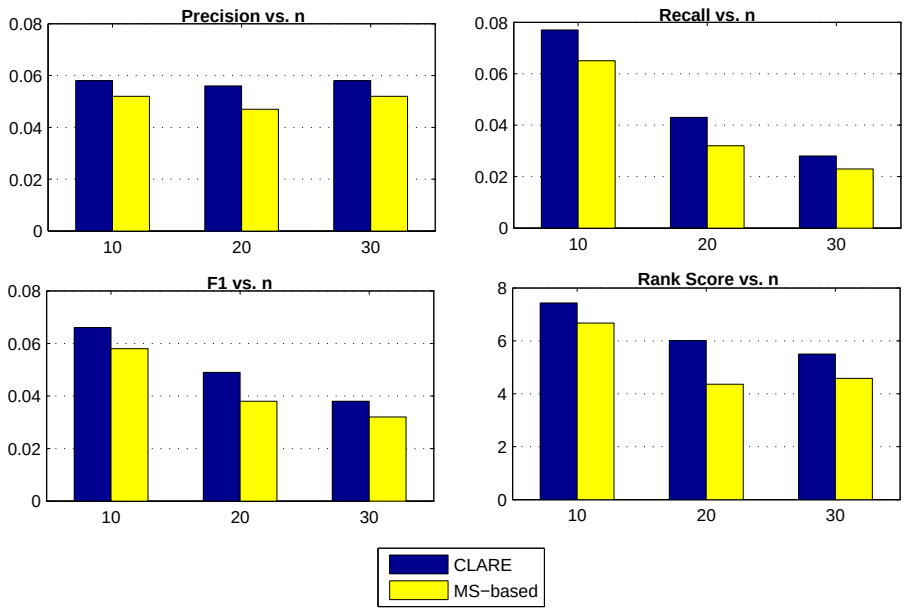

Figure 4.7: Comparison between CLARE and MS-based recommendation (hybrid), using *Genre* as attribute.

Pure content-based recommendation algorithm

The aspect model, the naive filterbot algorithm and the MS-based recommendation algorithm are hybrid algorithms that integrate content information about items into collaborative filters. Pure content-based recommendation algorithms are also capable of generating cold-start recommendations. We therefore implemented a content-based recommender, similar to that described in [110, 103, 141], based on the multinomial Naive Bayes (NB) model [100] and compared it with CLARE. In what follows, we first describe the design of the recommender, referred to as the *NB recommender*, and how it is adapted to the Top-N recommendation task. We then present results on the comparison between the NB recommender and CLARE.

The NB recommender addresses content-based recommendation as a text classification problem, in which items are regarded as documents (D), item attributes are regarded as features (F), and user-specified ratings are class labels (C) [110, 103]. As the Top-N recommendation task aims at determining items that active users may like, we used *Like* ($r_{u,i} \geq 4$) and *Dislike* ($r_{u,i} < 4$), instead of the exact ratings, as class labels. In other words, the set class labels used for building the NB recommender contains two members: $C = \{Like, Dislike\}$.

The NB recommender was implemented as follows. Firstly, given a class c_j and an attribute-value pair $f_i \in F$, the probability P($f_i|c_j$) is estimated from the training data with add-one (Laplace) smoothing using Eq. (2.28). Then, the probability that a movie i belongs to a class c_j, denoted as P($i|c_j$), is computed using Eq. (2.29) (recall that an item i takes the role of a document d in the context of movie recommendations). Note that the prior probability P(c_j) can be discarded in practice because its value is constant for all items for the same active user.

The above steps for building a NB recommender are the same as those for building a document classifier. We now describe the adaptation of the NB recommender to the Top-N recommendation task. Previous studies, such as [110, 141], used the NB recommender to assign a predicted rating (class label) to i based on the maximum posteriori probability P($i|c_j$). However, we aim at *ranking* the cold-start items for generating Top-N recommendations rather than the computing the exact predicted ratings of the items. We adapted the NB recommender to the Top-N recommendation task by first computing the

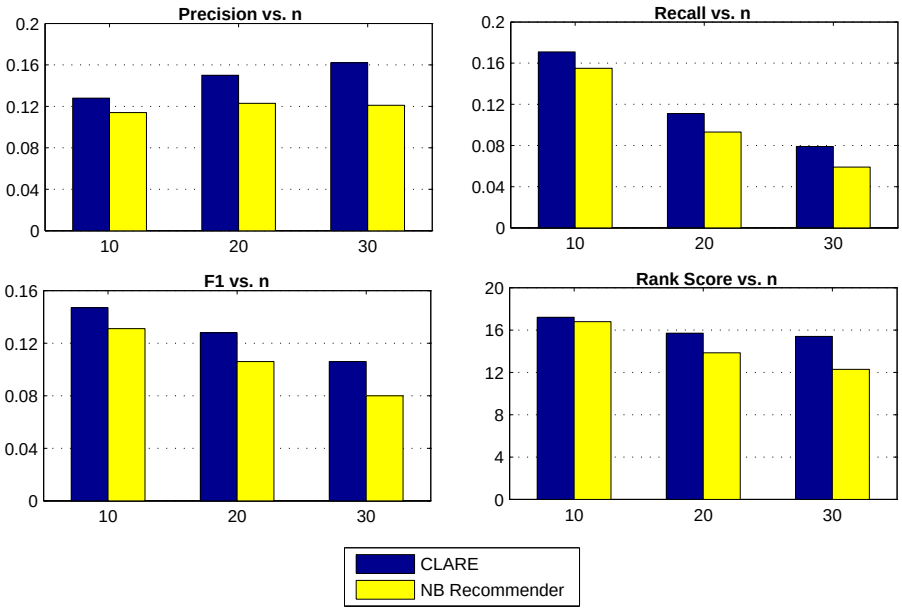

Figure 4.8: Comparison between CLARE and Naive Bayes (NB) recommender (content-based), using *Director* + *Plot* as attributes.

posterior odds of P($i|c_j = Like$) to P($i|c_j = Dislike$) for each recommendable item i for the active user a [110]. We then recommend the 10 items having the highest posterior odds to a, thereby facilitates comparison between the NB recommender and CLARE for the Top-N recommendation task.

Figure 4.8 shows the recommendation quality of the NB recommender and CLARE. We used *Director* + *Plot* of movies for training the NB recommender. Note that one NB recommender is trained for each active user, so that no collaborative information was used, and that the NB recommender is purely content-based.

CLARE is more robust than the NB recommender Figure 4.8 shows. Further, CLARE's advantage over the NB recommender becomes more significant as n increases. We conclude that CLARE is more effective than the content-based NB recommender in addressing the cold-start problem.

4.6 Summary

This chapter discusses our effort on addressing the cold-start problem in CF. We introduced a preference model comprising user-item relationships and item-

item relationships, and described how the proposed algorithm, CLARE, generates cold-start recommendations by CAR mining. The main feature of CLARE is that when no association rule for a certain item can be mined from ratings data, it takes into consideration the attributes the item has in common with other items to generate recommendations by CAR mining.

We also presented a comprehensive evaluation of CLARE. CLARE achieves good recommendation quality by making use of highly correlated items and item attributes for generating cold-start recommendations. It provides high coverage regardless of the number of cold-start items in the system. Further, it outperforms related algorithms, including MS-based and pure content-based, in recommending cold-start items. All these results are very encouraging, and suggest that our work successfully and effectively addressed the cold-start problem in CF.

We nonetheless identified two limitations of CLARE. Firstly, we mentioned in Chapter 4.3 that the `Item Attribute` layer in the proposed preference model may contain multiple sub-layers (e.g. in [162]), which are actually *attributes of the `Item Attribute` nodes*, in a real world application. We point out that such sub-layers are not likely to be useful for inferring preferences about the `Item` nodes in the context of recommender systems. Consider IMDb as an example, where `Item` nodes are movies, while `Item Attribute` nodes are directors, actors/actresses, plot summaries and so on. An example of an `Item Attribute` sub-layer is the portfolio of a particular actor, such as his place of birth, birthday and biography. One might agree that the presence of an actor in a movie may affect a user's rating for or interest in watching the movie, but the portfolio of the actor himself may not. For instance, it is unreasonable for a movie recommender system to predict that the user would like the movie because its leading actor was born on a certain date. This is why we do not consider `Item Attribute` sub-layers in our current work. It may still be interesting to study the behavior of CLARE with the presence of such sub-layers from a data mining perspective.

Secondly, our preference model assumes that a transaction supporting an item also supports all of the item's attributes. In other words, it assumes that a user giving a rating on a movie would also give the same rating on the movie's attributes, such as its cast and directors. This assumption can be relaxed with the current advent of sentiment analysis techniques [83, 79], which aim at extracting

and analyzing user opinions from textual reviews. This motivated our work on sentiment analysis and its possible integration with collaborative filters, as discussed in the next two chapters.

Chapter 5

Augmenting Ratings from Reviews for CF by Rating Inference

5.1 Introduction

Sentiment analysis deals with the automatic identification, extraction, and classification of opinions in texts. It can be used to develop applications that assist decision makers and information analysts in tracking user opinions about topics that they are interested in [170, 61, 168]. An example of sentiment analysis is the classification of a movie review as "thumbs up" or "thumbs down" [115].

One interesting application of sentiment analysis that has not yet received much research attention is the use of sentiment analysis to augment ratings for performing CF. User preferences in CF are usually collected either implicitly by capturing users' interactions with the system (e.g. purchase histories), or explicitly by asking users to give scalar ratings on items they have examined as noted. With the advent of Web 2.0 technologies, user-generated reviews are now popular a means for users to express their comments or preferences. Some review hubs, such as Amazon.com and the Internet Movie Database (IMDb), allow end-users to provide reviews in free text format. Such reviews can also be considered a type of "user ratings", although they are natural language texts that are not readily usable by existing CF algorithms. While the PHOAKS system [153] classifies web sites recommended by users in new group messages, it does not involve mining user preferences from texts.

We proposed hybrid recommendation algorithms that utilize concept hierarchies of domain items for addressing the problems of data sparseness and

cold-start recommendations in Chapters 3 and 4. In this chapter and the next, we attempt to address these problems along another dimension by the use of user-generated reviews. We describe in this chapter our work on utilizing user-generated reviews for CF by means of *rating inference*. Our work is motivated by the fact that while CF suffers from the problem of data sparseness, sentiment analysis is able to elicit user preferences expressed in textual reviews that are not readily usable for performing CF. More specifically, we propose to bridge the gap between sentiment analysis and CF. Our proposal offers two advantages. Firstly, it addresses the well-known data sparseness problem in CF by enabling existing CF algorithms to use user-generated reviews as an additional source of user preferences. Secondly, it helps extending CF to domains where numerical ratings on products are difficult to collect, or where preferences on domain items are more natural to be expressed as texts. An example of such domains is travel and tourism, in which the most successful recommender systems are built upon content- or knowledge-based techniques [128], although numerous textual reviews are available as travel journals and reviews. Integrating sentiment analysis and CF allows for the use of existing reviews for personalization purpose.

As described in Chapter 2.5.1, rating inference is a sentiment analysis task that aims at representing the overall polarity of opinions in text documents, which are user-generated reviews in our work, as numerical ratings. Such ratings can readily be used by existing CF algorithms, allowing easy and direct integration of sentiment analysis and CF. We demonstrated this empirically in this chapter based on our proposed Probabilistic Rating infErence Framework (PREF). PREF applies existing language processing techniques to extract interesting information from reviews. It determines the SO (sentimental orientation) and strength of opinion words using our proposed relative-frequency-based method, and then assigns numerical ratings to the reviews based on a probabilistic rating inference model. We compared PREF to several related studies to validate its robustness, and demonstrated its successful integration with CF.

The major technical contributions of our work are two-fold. Firstly, we proposed novel, simple yet effective methods for determining the SO and strength of opinion words, as well as the overall ratings of reviews. Secondly, we empirically demonstrated that rating inference is a feasible method for enabling review-based CF by successfully integrated PREF with the classical user-based CF algorithm.

To the best of our knowledge, this has not been done in any precedent work, and we hope our work can compel further research into the integration of sentiment analysis and CF.

The next section describes the dataset we collected to facilitate the purpose of this study. It also discusses the observations we made from the dataset. Chapter 5.3 details the design of PREF, the proposed rating inference framework. Chapter 5.4 discusses experimental results which validated the effectiveness of PREF, while Chapter 5.5 demonstrates how rating inference enables review-based CF. Finally, we summarize our contributions and findings in this work in Chapter 5.6.

5.2 Analysis of Movie Reviews

We used movie reviews as the domain of this study for two reasons. Firstly, there exist a large number of movie reviews on the Web that can be used for our experiments. Many of those reviews are accompanied by user-specified ratings that can be used as ground truth in our experimental study. Secondly, the movie domain is the most well-studied domain that has received great success in CF. We therefore would like to base this work, which represents our initial effort on integrating sentiment analysis and CF, on such domain.

We collected a set of movie reviews from the IMDb. We then examined the dataset to determine the linguistic processing tasks that should be included in our proposed framework. We performed some preliminary experiments on the processed dataset to analyze the characteristics of the dataset, especially the use of opinion words in the reviews. Experimental results helped us design appropriate methods for extracting opinion words and determining their SO and strength. The following subsections describe our data collection method and the observations we made from the preliminary experiments.

5.2.1 Data Collection

We first explain the need for collecting a new reviews dataset rather than adopting existing benchmarking CF and sentiment analysis datasets. Obviously, existing CF datasets, such as the MovieLens datasets and the book-crossing [178] dataset, cannot facilitate our study because they only contain ratings data. Existing

sentiment analysis datasets, however, do not fit the purpose of our study as well. One characteristic of CF applications is that the numbers of items and users, especially in large-scaled e-commerce applications, are large. We were, however, unable to find any sentiment analysis dataset having a considerable number of users and items that are comparable to widely used CF datasets (e.g. MovieLens datasets). In view of these, we collected our own dataset for this study.

We collected movie reviews from IMDb for the movies in the MovieLens 100k dataset, courtesy of GroupLens Research [75]. The MovieLens dataset contains ratings on 1,692 movies by 943 users. We removed movies that are duplicated or unidentifiable (movies without names), and crawled the IMDb, with a six- to ten-second delay between requests, to download reviews for the remaining movies. The reviews were downloaded as HTML pages. Each page contains 0 to 10 reviews. We developed a program to extract all reviews in each page. The resulting dataset contains approximately 50k reviews on 1,536 movies, provided by 1,805 different users. We then filtered out reviews without user-specified ratings, which are used for evaluating the proposed rating inference framework. We also discarded contributions from users who have provided fewer than 10 reviews to facilitate our future experiments on integrating rating inference with CF. The final dataset contains approximately 30k reviews on 1,477 movies, provided by 1,065 different users.

Each complete review in our dataset contains several headers and a text body. The headers include *movie ID*, *user ID*, *review date*, *summary*, which is a one-line summary in natural language text written by the user, and a *rating*, which is a user-specified number ranging from 1 (awful) to 10 (excellent). The text body is the user's comments on the movie.

5.2.2 Preliminary Experiments and Observations

We performed a set of preliminary experiments on the collected data. The purpose of the experiments is to examine the use of opinion phrases in reviews. We are particularly interested in this because the ultimate goal of our work is to integrate sentiment analysis of CF, due to the observation that reviews contain detailed preferences information that may be useful for generating collaborative recommendations. We seek to identify appropriate methods for extracting and

understanding user opinions in reviews based on the preliminary experiments.

In what follows, we first describe the method we used for identifying opinion words in reviews, and then detail the setup and results of our preliminary experiments.

Identifying opinion words

Previous work on subjectivity analysis suggests that *adjectives* have a strong association with subjectivity [17, 165]. We therefore consider adjectives to be opinions as in several other related studies [156, 61, 62]. In other words, our preliminary experiments focus on examining the use of adjectives in reviews, and we need to apply **POS tagging** to our dataset in order to identify adjectives from the reviews.

The most well-known POS tagger in the NLP literature is the Brill tagger [15]. However, we adopted another NLP processor known as MontyLingua [92] in our work, because MontyLingua was developed based on the Brill tagger, but produces higher tagging accuracy (around 97%). The sentence below shows an example of the output produced by the POS tagging function of MontyLingua.

> "*Good/JJ beginning/NN and/CC end/NN but/CC unpleasant/JJ middle/NN*"

In the above sentence, each *token*, which is a whitespace-delimited string, represents a "word/POS tag" pair. The POS tags *JJ*, *NN* and *CC* represent *adjective*, *noun* and *coordinating conjunction* respectively.

We also paid attention to the use of negation words, including "not", "never" and "neither", in the reviews. Consider the following sentence extracted from a review in our dataset:

> "*The first half is a painful experience, while the second half is simply* ***not good***."

In the above example, the use of the word "not" has a negation effect on the adjective "good". We applied a simple **negation tagging** heuristic to address the effects of negation words [24, 115]. Specifically, if the negation tagging process identifies a negation word in a sentence, it adds a special tag to the adjectives appeared after the negation word in the rest of the sentence. In the

above example, the adjective good would be changed to "NOT_good", which is then treated as a separate opinion word. Negation words may be written in short forms, such as "don't". We therefore applied fuzzy string matching by regular expressions when detecting negation words in a sentence to allow word variants such as "do not" and "dont", which is not uncommon in casual forms of writing including user-generated reviews.

The use of opinion words in reviews

We performed a set of preliminary experiments to analyze the use of opinion words in reviews as noted. The purpose of doing so is to investigate appropriate methods for determining the SO and strengths of opinion words.

We first applied POS tagging and negation tagging to our dataset as aforementioned. We then randomly sampled from the dataset three training sets, namely *T10*, *T5* and *T1*, each containing 500 reviews having user-specified ratings of 10/10, 5/10 and 1/10 respectively. These ratings were chosen as they seem to be appropriate representative cases for positive, neutral and negative sentiments.

We used a program to extract the processed adjectives from our dataset, and compute their frequency counts from each of the training sets. Some frequent opinion words appeared the training sets were selected for further analysis.

The number of distinct opinion words appeared in the training sets is 4,545, among which 839 (around 18.5%) appeared in two of the three training sets, and 738 (around 16.2%) appeared in all three. We further examined opinion words that appeared in more than one training set. Table 5.1 lists the 15 most frequent opinion words (Top 15) of this kind in each training set in descending order of their frequency counts. In the table, the number in brackets following an opinion word is its relative frequency in the particular training set, computed as its frequency count in the training set divided by its total frequency count in all training sets. Boldface is used to highlight words having the highest relative frequency among the three training sets.

We made the following observations based on the analysis on the occurrence and relative frequencies of opinion words in the training sets:

1. In general, the relative frequencies of opinion words that are in general considered to be "Positive" are usually, *but not always*, the highest in T10

Table 5.1: Top 15 opinion words with relative frequencies.

Training set	Opinion words with relative frequencies
T10	**best (0.68)**, **great (0.66)**, good (0.33), **many (0.47)**, **first (0.38)**, **classic (0.71)**, better (0.30), **favorite (0.75)**, **perfect (0.75)**, **greatest (0.85)**, **wonderful (0.83)**, **excellent (0.70)**, funny (0.36), **sad (1.00)**, **brilliant (0.81)**
T5	**good (0.39)**, **more (0.54)**, **much (0.51)**, bad (0.35), **better (0.41)**, other (0.32), **few (0.73)**, great (0.21), first (0.34), best (0.19), **little (0.47)**, many (0.29), **funny (0.38)**, **NOT_good (0.45)**, **NOT_bad (0.55)**
T1	**bad (0.65)**, good (0.28), **worst (0.89)**, much (0.49), more (0.46), other (0.28), first (0.28), better (0.29), many (0.24), great (0.13), best (0.13), **stupid (0.56)**, **boring (0.56)**, NOT_good (0.36), **only (0.48)**

Table 5.2: Top 1 opinion words with relative frequencies.

	Understood	**Relative frequency in:**		
Opinion word	**SO (strength)**	**T10**	**T5**	**T1**
best	positive (strong)	**0.68**	0.19	0.13
good	positive (mild)	0.33	**0.39**	0.28
bad	negative (strong)	0	0.35	**0.65**

and the lowest in T1. On the contrary, those of "negative" opinion words are usually the highest in T1 and the lowest in T10. Table 5.2 lists as examples the relative frequencies of the most frequent opinion word (Top 1) in each training set. Boldface is used to highlight the highest relative frequency of each opinion word.

This observation probably suggests that relative frequencies may help determining the SO and strengths of opinion words. For example, the word "best" appeared in T10 for 68% of the time. It may therefore be considered a positive opinion word with a strength of 0.68.

2. Referring to the previous observation, even opinion words having strong positive (resp. negative) SO in the T10 (resp. T1) training set appeared in the other two training sets as well. We suggest to allow an opinion word to

have multiple SO when performing rating inference. This models the fact that an opinion word can appear in reviews having different ratings.

Adopting the fuzzy set concept [173], which means that an attribute can be a member of some fuzzy sets to certain degrees, we allow an opinion word to have multiple SO and strength. A membership degree is determined by a membership function, and its value is in the range [0, 1]. In the context of our work, such "membership degree" with respect to a certain SO can be determined by the relative frequency of a word in the corresponding training set. For instance, the word "best" have SO *Positive*, *Neutral* and *Negative* with the strengths 0.68, 0.19 and 0.13 respectively.

3. Using a relative-frequency-based method to determine the SO and strengths of opinion words means that opinion words appeared more frequently in T10 are considered positive sentiments, and vice versa. We found that the resulting SO of opinion words may not agree with their generally understood SO. An example is the word "frightening" which seems to express a negative sentiment. In our movie review dataset, however, its relative frequency in T1 is only 0.29.

4. We further conclude that synonyms do not necessarily have similar SO based on the previous observation. For example, the word "terrible", is a synonym of the word "frightening", but its relative frequency in T1 is 0.75 (that of "frightening" is 0.29).

   We described in Chapter 2.5.3 a class of existing techniques that makes use of a set of seed adjectives and the semantic similarities between word meanings to determine SO of opinion words [61, 62, 72, 68]. These studies assumed that *semantic similarity implies sentimental similarity*. Our analysis, however, indicates that this is not necessarily true. While such semantic-similarity-based methods may help performing topic-based classification, it may not be applicable to sentiment analysis. This further suggests that our relative-frequency-based method overcomes a major limitation of existing semantic-similarity-based methods, because it allows similar words to have totally different SO.

5. We noticed that many words appeared in the training sets may not have clear sentiments, or are not expressing sentiments at all. Examples include

"116-minute", "yellow" and "year-old". Further, many words with the strength 1 are typos or meaningless words that appeared only once in the training sets. Examples include "woodi", "directer" and "so-good-i-wanna-see-it-again-and-buy-the-dvd". We therefore attempted to prune the extracted opinion words, or the *opinion dictionary*, with the concern that these irrelevant words, or *noises*, may have adverse effects on the performance of the proposed framework. Chapter 5.4.3 describes the pruning method and results.

To sum up, our analysis of movie reviews suggests that relative frequencies of opinion words may be useful indicators of their SO and opinion strength. To the best of our knowledge, this simple relative-frequency-based method has not been adopted in related studies, thus its effectiveness has to be evaluated empirically. Further, we propose to allow opinion words to have both positive and negative to certain degrees to facilitate rating inference. Although the algorithm in [68] determined both positive and negative SO of words, it only used the primary (dominant) SO of words when performing sentiment classification. The effectiveness of using multiple SO should therefore be studied. We are also concerned about effects of noises (irrelevant words) that may have on the performance of the proposed framework. We therefore attempted to prune the opinion dictionary to minimize their adverse effects, if any.

5.3 PREF: A Probabilistic Rating Inference Framework

PREF is a probabilistic rating inference framework developed to support the integration of sentiment analysis and CF. It includes four major steps, namely data preparation, feature extraction, opinion dictionary construction and rating inference. Figure 5.1 depicts an overview of PREF and how it is related to CF.

5.3.1 Data Preparation

The data preparation step preprocesses user reviews for the subsequent analysis. Different preprocessing may be required depending on the data sources. For

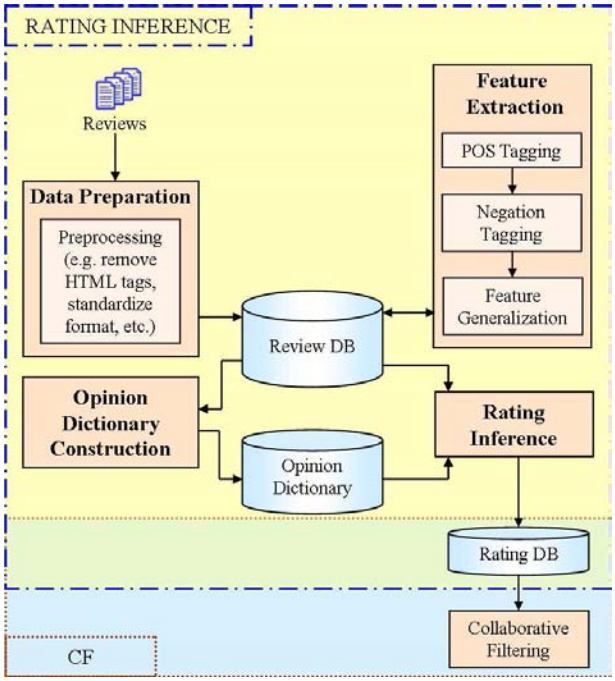

Figure 5.1: Overview of PREF

example, if user reviews are downloaded as HTML pages, the HTML tags and non-textual contents they contain are removed in this step.

A user review is usually a semistructured document, containing some structured headers and an unstructured text body (comments). Sentiment analysis algorithms usually do not require information other than the user-specified comments and ratings (e.g. for performance evaluation), but PREF extracts also the identities of users and domain items because they are needed for performing CF. Note that the term "reviews" hereafter refers to the textual comments given by users.

5.3.2 Feature Extraction

This step extracts features, including opinion words and item features, from the reviews. In additional to the aforementioned POS tagging and negation tagging tasks, we also performed *feature generalization* on our dataset.

Feature generalization is about generalizing interesting features that may be overly specific [25], such as product brands and movie names. Such features are identified with the aid of POS tags by matching proper nouns in reviews with attributes of domain items. In the movie reviews domain, for example, a

sentence "*Toy Story is pleasant and fun.*", in which "Toy Story" is the name of the movie being reviewed, is generalized to "*_MOVIE is pleasant and fun.*". We adopted this task as a kind of discourse analysis to facilitate rating inference which involves assigning weights to opinions associated with interesting item features.

After analyzing a review d using the above tasks, this feature extraction step produces a representation of d as a list of opinion words, V_d = $\{v_1, v_2, ..., v_n\}$, and a list of item features, F_d = $\{f_1, f_2, ..., f_n\}$. Each v_i is an adjective, and f_i is the nearest noun or noun phrase (item feature) in the sentence from which v_i is extracted. If the associated feature of an entry in V_d cannot be identified, then the corresponding entry in F_d is null. V_d is used in the following opinion dictionary construction and rating inference steps, while F_d is used in the rating inference step for assigning weights to opinions.

5.3.3 Opinion Dictionary Construction

An opinion dictionary contains opinion words, their estimated SO and the strength of their SO. We described in Chapter 5.2 that existing semantic-similarity-based methods for determining SO assume that similar word meanings imply similar SO, but such assumption may not hold in sentiment analysis. We therefore proposed a novel relative-frequency-based method to overcome this. Our method estimates the strength of an opinion word v_i with respect to a sentiment class c_j, denoted as $OS(v_i, c_j)$, as follows:

$$OS(v_i, c_j) = \frac{N(v_i, c_j) + \alpha}{\sum_{k=1}^{|C|} \frac{N(V_j)}{N(V_k)} * [N(v_i, c_k) + \alpha]} \tag{5.1}$$

where α is a small number that serves as a smoothing factor to avoid zero probabilities of unseen opinion words, c_j and c_k are elements in C, which is the set of sentiment classes used for determining SO. Note that sentiment classes in the rating inference task are ratings, for example, C = $\{1, 2, 3, 4\}$. $N(v_i, c_j)$ and $N(v_i, c_k)$ denote the frequency counts of v_i in c_j and c_k respectively in the training corpus. $\frac{N(V_j)}{N(V_k)}$ is a normalization factor, where $N(V_j)$ and $N(V_k)$ are the total frequency counts of the opinion words in classes c_j and c_k respectively. More specifically, it normalizes the frequency of v_i in a given sentiment class by the frequency of all opinion words in that class.

The proposed method offers three advantages. Firstly, it allows an opinion word to have multiple SO, each with a corresponding strength. This addresses the facts that opinion words may appear in more than one sentiment class, and that even a review associated with a high rating could contain negative comments on the relevant subject, and vice versa. Secondly, the SO and strength of an opinion word are determined by its relative frequencies of appearance in different sentiment classes. The SO of an opinion word is therefore not limited to its generally understood SO or the SO of its semantically related words, thereby overcomes a major limitation of semantic-similarity-based methods. Thirdly, the opinion dictionary can be maintained easily because new words can be added without having to rebuild the entire dictionary. The frequencies of existing opinion words can also be updated incrementally.

We mentioned in Chapter 5.2.2 that our opinion dictionary contains noises, because our method identifies opinion words based on POS information. Noises include words without clear SO or are not expressing sentiments, meaningless words and typos. Due to the concern that the noises might adversely affect the performance of PREF, we attempted to prune the opinion dictionary and then experimentally evaluated the effect of the pruning. Results generally suggest that the noises do not affect performance adversely, thus pruning is not necessary in PREF. Chapter 5.4.3 provides details about the pruning method and results.

5.3.4 Rating Inference

Rating inference aims at determining the overall SO of a review based on the SO of the opinion words it contains. It has been viewed as a multi-category classification task, in which class labels are scalar ratings, such as 1 to 5 "stars". It is, however, different from standard topic-based classification because class labels in the rating inference task are ordered, and there exist different degrees of similarity between the class labels. For instance, "5 stars" is intuitively closer to "4 stars" than to "2 stars". Previous studies therefore addressed rating inference as a regression task [112, 114], and it is important to take the intuitive similarities between class labels into consideration in the rating inference process.

PREF infers a predicted rating from an unseen review d as follows. Firstly, d goes through the feature extraction step which returns the list of opinion words in d, $A_d = \{a_1, a_2, ..., a_i\}$, and the product features associated with A, $F_d =$

$\{f_1, f_2, ..., f_i\}$. Secondly, the weights of F, $W_d = \{w(f_1), w(f_2), ..., w(f_i)\}$, are determined based on some predefined criteria. Note that $w(f_i)$ can be defined with respect to a particular review d, which is written by a single user in the system. This means that individual users' preferences for different features can be catered if such preferences are available. Finally, a rating, which is the overall SO of d, is estimated for d given the opinion dictionary, A_d and W_d.

Weighting item features

PREF provides the flexibility of weighting different opinion words according to the estimated importance of their associated product features in reviews, if any. Adopting weights also allows easy integration with user-specified interest profiles if necessary. For example, if a certain user of a movie recommender system specified that he is particularly interested in a certain actor, then the acting of that actor in a movie may have stronger influence on the his preference for the movie.

Learning user-specific feature weights is an interesting task in its own right. It is, however, not within the scope of this study. We therefore only experimented with features that are intuitively important for rating inference. Specifically, we hypothesized that opinions towards a product as a whole may be more important for determining the overall rating of a review, and assigned more weights to such features in the rating inference process as described in Sect. 5.4.2. We point out that learning user-specific feature weights can be conducted independent of PREF, and such weights can flexibly be used in PREF once they are available.

Inferring a rating from a review

An overall rating is inferred from a review d based on the SO of the opinion words d contains. Our rating inference model, inspired by the NB classifier, consists of three steps. Firstly, assuming conditional independence of opinion words, we assign a predicted score, denoted by $PS(d, c_j)$, to d with respect to a sentiment class c_j as follows:

$$PS(d, c_j) = P(c_j) \prod_{i=1}^{|V_d|} OS(v_i, c_j)^{1/w(f_i)} \tag{5.2}$$

where $P(c_j)$ is the prior probability of c_j in the training data, $OS(a_i, c_j)$ is the

strength of a_i with respect to c_j (Eq (5.1)), and $w(f_i)$, where $w(f_i) > 0$, is the weight assigned to the feature f_i in d. Opinion words in A_d that are also in the opinion dictionary will contribute to the calculation of $PS(d, c_j)$. Note that Eq. (5.2) is not a rigorous probability formulation. It serves as an approximation of how likely d belongs to the sentiment class c_j.

Secondly, after computing $PS(d, c_j)$ for every $c_j \in C$, the values of the set of resulting $PS(d, c_j)$ are normalized so that their sum equals 1:

$$PS\prime(d, c_j) = \frac{PS(d, c_j)}{\sum_{k=1}^{|C|} PS(d, c_k)} \tag{5.3}$$

The normalized value $PS\prime(d, c_j)$ is the estimate of the probability that d belongs to the sentiment class c_j.

Finally, the overall SO of d, denoted as $SO(d)$, is estimated as follows:

$$SO(d) = \sum_{j=1}^{|C|} c_j * PS\prime(d, c_j) \tag{5.4}$$

where c_j is the value of the j^{th} class label in C (recall that class labels are numerical ratings). Considering $PS\prime(d, c_j)$ to be the probability that d has the rating c_j, Eq. (5.4) is equivalent to the calculation of Expected Value in probability theory. We use an example to explain our design. Suppose we want to rate d on a 5-point scale. Consider a simplified situation where d describes a 5-star (excellent) feature and a 1-star (awful) feature, assuming equal weights of the features. It would then be reasonable to expect a rating of approximately 3 out of 5 although d does not describe any 3-star feature. In this light, we point out that rating inference shall not be considered a standard classification problem. Computing $SO(d)$ as an Expected Value takes into consideration the continuity of class labels in the rating inference task.

A characteristic of our model is that the final rating assigned to d need not be a member in C. As class labels in C are numerical values, $SO(d)$ can fall between two adjacent c_j values in C. For example, if $SO(d)$ is 2.8 and C = $\{1, 2, 3, 4\}$, it can be rounded off to the nearest value in C which is 3, but it is also quite natural to say "the predicted rating of this review is 2.8 out of 4". We therefore do not conclude how the final rating for d should be calculated from $SO(d)$ when $SO(d)$ falls between two adjacent c_j values as this is a flexible, application-dependent decision.

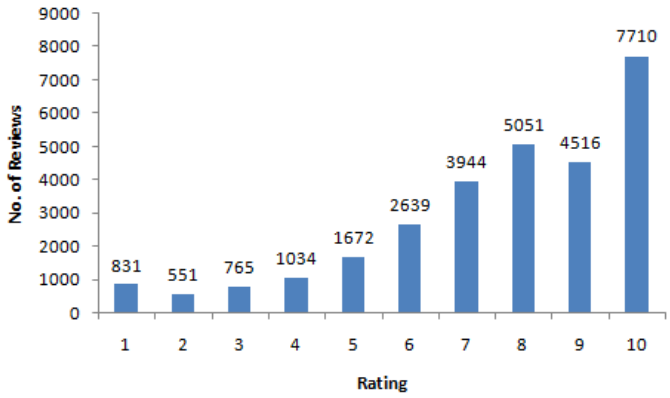

Figure 5.2: Distribution of ratings in our movie reviews dataset.

5.4 Experimental Results

We conducted extensive experimental studies on PREF to validate its effectiveness in inferring ratings from reviews. This section first describes the evaluation method we used, and then discusses the results of two groups of experiments. The first group of experiments, presented in Chapter 5.4.3, focuses on the performance and behavior of PREF, whereas the second group of experiments, presented in Chapter 5.4.4, aims at validating the robustness of PREF against various related algorithms.

5.4.1 Method

We used the movie reviews dataset described in Chapter 5.2.1 for our experiments. We randomly split the dataset into five non-overlapping, roughly equal-sized folds, and reported all results based on the averages of the five folds. In each experiment, one of the five folds was used as the test set, and a certain number of reviews were randomly sampled from the other four folds as the training set for building the opinion dictionary. This was designed to facilitate our experiments on the effects of using different numbers of training reviews for classifying the same set of test reviews. In the subsequent discussions, the term "training set" refers to the set of reviews that was actually used for building the opinion dictionary. Each training set contains 1,500 randomly sampled reviews with uniform class distribution unless otherwise stated. The effect of skewed class distribution in the dataset is out of the scope of this work. The same sets of training and test reviews were used for all comparative evaluations.

PREF was evaluated with respect to a 3-point and a 4-point integer rating scale, following the settings in [114]. The original ratings (r) in the dataset were recorded on a 10-point scale in the distribution shown in Figure 5.2. They are transformed into $r\prime$ to facilitate our experiments:

$$r\prime = \begin{cases} 1, & \text{if } 1 \leq r \leq 3 \\ 2, & \text{if } 4 \leq r \leq 7 \\ 3, & \text{if } 8 \leq r \leq 10 \end{cases}$$

in the 3-point case, and

$$r\prime = \begin{cases} 1, & \text{if } 1 \leq r \leq 3 \\ 2, & \text{if } 4 \leq r \leq 5 \\ 3, & \text{if } 6 \leq r \leq 7 \\ 4, & \text{if } 8 \leq r \leq 10 \end{cases}$$

in the 4-point case. Note that the terms "class" and "class label" refer to $r\prime$ in the subsequent discussions.

We adopted MAE and MSE as evaluation metrics because this work is concerned with the rating prediction task. We also measured the coverage rates achieved the various algorithms. Coverage in this work is defined as the percentage of reviews in the test set for which ratings can be inferred. We found that the coverage rates produced by the various algorithms in different experimental settings only varied slightly, with a range from 98.7% to 100%. We therefore did not include this metric in the subsequent discussions.

Note that classification accuracy (ACC), which is the percentage of reviews that were classified (rated) correctly, has been commonly used for evaluating sentiment classification algorithms (e.g. [156, 115, 25, 114, 41]). However, we point out that ACC is not an appropriate metric for evaluating rating inference algorithms for two reasons. Firstly, it is necessary to round off $SO(d)$ to the nearest value in C in order to obtain the ACC of an algorithm. When doing so, a very small difference in the $SO(d)$ of two reviews can cause them to be assigned two different ratings, a problem known as the *sharp boundary problem*. For instance, a $SO(d)$ value of 2.49 will be rounded off to 2, whereas that of 2.5 will be rounded off to 3 using an integer scale. Secondly, the notion of "accuracy" reflected by ACC is too coarse-grained for the rating inference task as ACC treats mis-rating of all intensity equally. Specifically, given a $r\prime_d$ of 4,

an algorithm that produces a $SO(d)$ of 3 is obviously more "accurate" than one that produces a $SO(d)$ of 1 [25]. We therefore adopted MAE and MSE instead of ACC in our experimental studies.

5.4.2 Parameters

We now report the test-set-optimal values of two adjustable parameters in PREF. The first parameter is α, which is a small number that serves as a smoothing factor in Eq. (5.1) when computing the strength of opinion words. We experimented with the values $\alpha \in \{0.001, 0.01, 0.1, 1\}$, and set $\alpha = 0.01$ in the subsequent experiments as such a setting produced the best results.

The second parameter is the weights assigned to product features for reflecting their estimated importance to the overall sentiments of users (Sect. 5.3.4). Note that individual users' preferences for various product features are not known in our work. We therefore applied the same weights for all users to two types of features that can be identified from the data and are intuitively important for rating inference:

1. The *_MOVIE* feature. If a proper noun or noun phrase represents the name of the movie being reviewed, it was generalized as *_MOVIE* in the feature generalization step. The words "movie" and "film" were also replaced by the tag *_MOVIE*.

2. All features appeared in the *summary* of a review, as we observed that users tend to state their overall recommendations for the movies concerned in the one-line summary of the reviews.

We experimented with the weights {1, 2, ..., 10}, and found that assigning both types of features a weight of 3 produced the best results. This suggests that users's opinions on such features are indeed more important for determining the overall SO of reviews.

5.4.3 Evaluation of PREF

We implemented two baseline algorithms for benchmarking purpose:

1. **Majority baseline algorithm:** This algorithm always assigns a test review to the majority class, which is 3 in the 3-point setting and 4 in the 4-point setting.

Table 5.3: Summary of experimental results on PREF and the baseline algorithms. Each training set contained 1500 reviews with uniform class distribution.

		3-point		4-point	
	Algorithm	**MAE**$_3$	**MSE**$_3$	**MAE**$_4$	**MSE**$_4$
1.	*Majority baseline*	0.475	0.625	0.645	1.280
2.	*Heuristic baseline*	0.702	0.870	1.106	1.981
3.	PREF	**0.339**	**0.341**	**0.566**	**0.799**
4.	PREF with pruning	**0.353**	**0.340**	**0.576**	**0.790**

2. **Heuristic baseline algorithm:** Given a test review d, if the movie m that d is concerned with appeared in the training set, the heuristic baseline algorithm assigns the average rating that m received in the training set to d. Otherwise, it assigns a rating in C to d at random.

Table 5.3 summarizes our experimental results on the baseline algorithms and PREF. In the subsequent discussions, we use MAE$_n$ and MSE$_n$ to respectively represent the MAE and MSE produced using the n-point rating scale. When we describe results as significant, we mean so statistically based on the Wilcoxon signed-rank test (using a 95% significance level), a non-parametric version of the popular paired t-test. We boldfaced the best results and those that are statistically indistinguishable from the best in all tables presenting the results.

In what follows, we first discuss the results in Table 5.3, followed by an investigation into the effects of the size of training set on the performance of PREF. We then compared PREF to several related studies to validate its effectiveness.

General observations

We made four general observations from the experimental results on the majority baseline, the heuristic baseline, and PREF (lines 1-3 of Table 5.3). Firstly, all algorithms perform better in the 3-point setting than in the 4-point setting, a finding that is qualitatively consistent with that reported in [114]. This suggests that the task of rating inference becomes more challenging when a finer-grained rating scale is used.

Secondly, regarding the two baseline algorithms, the majority baseline always outperforms the heuristic baseline significantly. In fact, it might be

reasonable to conclude that the majority baseline performs quite well despite its simplicity, because on average more than 60% of test reviews fall into the majority class in our dataset. We therefore only compare the performance of the various algorithms with that of the majority baseline in the rest of this section.

Finally, PREF always produces significantly better results than the majority baseline algorithm. We conclude that PREF is capable of learning user preferences from reviews and performing rating inference effectively.

Pruning the opinion dictionary

We observed that our method for building the opinion dictionary produces noises, including words without clear SO or are not expressing sentiments, as described in Chapter 5.3.3. This is because our method identifies opinion words only based on POS information. We attempted to prune the noises in the opinion dictionary and evaluated the effect of the pruning.

The pruning method we adopted was inspired by the SO determination method of Hu and Liu [61]. We first constructed an opinion dictionary, from which we manually selected a set of 30 seed adjectives (the seed set) having clear SO. Table 5.4 lists the seed adjectives we used for pruning the opinion dictionary. The set contains both positive and negative opinions, such as "amazing" for positive SO, and "awful" for negative SO. We systematically examined the opinion words in the "noisy" opinion dictionary one by one. We used WordNet [107] for determining the synonyms and the antonyms of a given word v_i. If the seed set contained any of those words, v_i was then added to the seed set as well. This process repeated until no more words were added to the seed set. Finally, the words in the opinion dictionary that were not in the seed set were pruned. The pruned dictionary was then used for performing rating inference. The performance achieved by this set of experiments is shown in line 4 of Table 5.3 with the label "PREF with pruning".

The performance of "PREF with pruning" is not statistically better than that of "PREF", meaning that pruning does not improve the performance of PREF. There are two possible reasons for this. Firstly, meaningless words and typos, most of which appeared only once in the training reviews, are not likely to appear in unseen reviews again. They therefore would not hurt performance. Secondly, opinion words that are not noises might be pruned because they do not have synonyms or antonyms (either direct or indirect) in the seed set as determined

Table 5.4: Seed adjectives used for pruning the opinion dictionary.

SO	**Seed list**
Positive	amazing, brilliant, classic, excellent, favorite, first, fun, interesting, original, perfect, real, special, top, wonderful, young
Negative	awful, bad, boring, dull, hard, late, least, little, old, only, own, poor, stupid, terrible, worst

by WordNet. An example of such words is "unforgettable". It appeared in the best rated reviews 72.1% and 60.1% of the time on average in the 3-point and the 4-point cases respectively but was pruned in any case. This might be rectified by defining an "optimal" seed set that can retain *all* meaningful opinion words, but coming up with such a set is obviously difficult, if not impossible. In view of these, we conclude that pruning of "noises" is unnecessary in PREF.

Effects of the size of training set

A training set refers to the set of reviews that were actually used for building the opinion dictionary. We investigated how the sizes of training set affect the performance of PREF. We started with a training set with 300 randomly selected reviews. We iteratively incremented the size of the training set by 300, until it reached 4,800. While the training reviews were selected by random, we ensured that the training set contained the same number of reviews per sentiment class. In each iteration, we updated the frequencies of opinion words in the opinion dictionary and then performed rating inference on the test set based on the updated dictionary. Figure 5.3 shows the results of this experiment. In the figure, the notation "A-n" denotes the performance of algorithm A using the n-point rating scale. Note that the performance of the majority baseline (dashed lines in Figure 5.3) remains constant because the majority class in the dataset does not change with the size of the training set.

We first observed from Figure 5.3 that the performance of PREF improves as more reviews are added to the training set, and the improvements are more significant when the size of the training set grows from 300 to 1,200. These suggest that PREF can produce more accurate rating inference over time as more reviews are available in the system, although diminishing improvements shall be

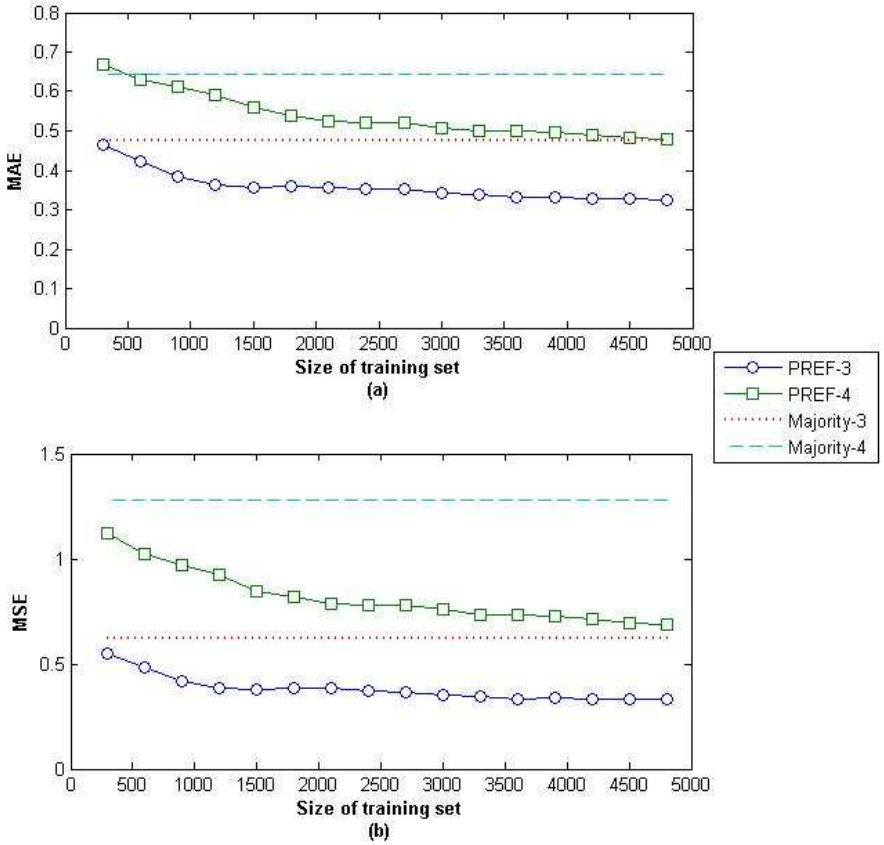

Figure 5.3: Learning curves of PREF and the majority baseline showing how (a) MAE, and (b) MSE change with respect to the size of training set in the 3-point and the 4-point settings.

expected.

PREF always outperforms the majority baseline significantly in terms of MSE. It yields lower MAE values than the majority baseline when 300 and 600 training reviews were used in the 3-point and the 4-point settings respectively. We conclude that PREF achieves reasonably good performance when only limited labeled (rated) reviews are available. For instance, using 300 training reviews for building the opinion dictionary produced a MAE_3 of 0.466 and a MAE_4 of 0.667. These values mean that the $SO(d)$ values predicted by PREF were within approximately 23% of the true ratings using both rating scales. This finding is very encouraging as it indicates that it is possible to apply our work to corpora where labeled (rated) reviews are rather limited. Examples of such corpora include weblog and newsgroup messages that are not collected from review hubs so that hand-labeling of the documents may be necessary.

5.4.4 Comparisons with Related Work

We compared our work with four pieces of related work capable of performing rating inference. They include a standard NB classifier, an extension to the "baseline method" of Dave et al. [25], SVR, and the graph-based method proposed by Goldberg and Zhu [41]. Note that comparisons were always conducted based on the same feature set (opinion words).

Table 5.5 summarizes part of our comparative experiments. We included the majority baseline in the table to help indicate the effectiveness of the various algorithms. The following subsections detail the design of the experiments and discuss the results.

Naive Bayes (NB) classifier

NB has been well-known for its simplicity and its successful applications to many classification problems. It has also been applied to binary sentiment analysis, and has been found to perform generally well despite its simplicity [115, 25]. We implemented a standard NB classifier for rating inference by regarding rating inference as a multi-category classification problem. We only used opinion words (V) as features in the NB classifier as in PREF, and members in C as class labels.

The performance of the NB classifier is reported in line 3 of Table 5.5.

Table 5.5: Summary of comparisons between the majority baseline, PREF, NB classifier, a method based on Dave et al. [25], and SVR. Each training set contains 1,500 reviews with uniform class distribution.

		3-point		4-point	
	Algorithm	**MAE**$_3$	**MSE**$_3$	**MAE**$_4$	**MSE**$_4$
1.	*Majority baseline*	0.475	0.625	0.645	1.280
2.	PREF	**0.339**	**0.341**	**0.566**	**0.799**
3.	NB Classifier	0.411	0.492	0.646	1.080
4.	Dave et al.	0.396	0.475	**0.578**	1.007
5.	SVR	0.569	0.479	0.861	1.033

Results show that PREF always outperforms the NB classifier significantly. One contributing factor to this is that the NB classifier assigns a review d to the sentiment class having the *highest* posterior probability. This neglects the fact that rating inference is about determining the *overall* SO of a review. PREF, in contrast, addresses the continuity of class labels (scalar ratings) as every $PS(d, c_j)$ value contributes to the determination of its overall SO.

Baseline Method of Dave et al.

The "baseline" method proposed by Dave et al. [25] was designed for binary sentiment classification, but can be flexibly adapted to the multi-point rating inference task. We did this based on the idea of the OVA approach [132] for extending a binary classifier for n-ary outputs. We first detail the adaptations we made to their method and then describe the results.

Dave et al. assigned to an opinion word v_i the score $score(v_i)$ as a measure of bias ranging from -1 to 1 using Eq. (2.32) (Chapter 2.5.3 on page 49). We extended Eq. (2.32) based on the idea of OVA classification to compute the score of v_i for every $c_j \in C$. Such score, denoted by $score(v_i, c_j)$, is normalized so that the set of $score(v_i, c_j)$ obtained by v_i sums to 1. It is used as the estimated OS of v_i with respect to c_j. Formally, $OS_{Dave}(v_i, c_j)$ is defined as:

$$OS_{Dave}(v_i, c_j) = \frac{score(v_i, c_j)}{\sum_{c \in C} score(v_i, c)} \tag{5.5}$$

Given an unseen review d and the list of opinion words $V_d = \{v_1, ..., v_i\}$ it contains, Dave et al. [25] summed up the total SO of V_d with respect to

each sentiment class, and then assigned d to the class with the dominant SO. Following their ideas, we first assign a predicted score to d with respect to a sentiment class c_j as follows [25]:

$$PS_{Dave}(d, c_j) = \sum_{i=1}^{|V_d|} OS_{Dave}(v_i, c_j) \tag{5.6}$$

After computing $PS_{Dave}(d, c_j)$ for every $c_j \in C$, $SO_{Dave}(d)$ is estimated to be the sentiment class c_j having the highest $PS_{Dave}(d, c_j)$ value. That is,

$$SO_{Dave}(d) = \arg\max_{c_j \in C} PS_{Dave}(d, c_j) \tag{5.7}$$

Line 4 of Table 5.5 reports the performance of the above method based on Dave et al. [25]. Such a method always outperforms the majority baseline significantly, and performs well in terms of MAE_4. However, PREF shows significantly superior performance in terms of MAE_3, MSE_3 and MSE_4. In other words, PREF always performs better than the method based on Dave et al. [25] in the 3-point setting, and it always produces fewer large errors.

Support Vector Regression (SVR)

Support Vector Regression (SVR) [159, 147] has been used for performing rating inference in [114, 112, 41]. SVR assumes that class labels come from a discretization of a continuous function. It is therefore able to capture the order and continuity of class labels in the rating inference task [114, 112]. Its performance has been found to be comparable to sophisticated machine learning models and to human performance [114, 112, 41].

We applied SVR to our rating inference task based on the SVM^{light} package [66] in three steps. Firstly, we created feature vector representations of our dataset. We used only opinion words as features for representing each review as a $\{0, 1\}$ word-presence vector, and such vector is normalized as in [41]. Secondly, we applied linear ϵ-insensitive SVR with all default parameters to our training sets, following the settings in [41]. The output of this step is a set of learned regression models, one for each training set. Finally, we applied the learned models to the corresponding test sets to perform regression. Line 5 of Table 5.5 reports the results of this experiment.

PREF always outperforms SVR significantly in this experiment as shown in the table. Further, SVR produces mixed results as compared to the majority

Table 5.6: Comparison with SVR: Each training set contains 4,800 reviews with uniform class distribution.

		3-point		4-point	
	Algorithm	**MAE$_3$**	**MSE$_3$**	**MAE$_4$**	**MSE$_4$**
1.	*Majority baseline*	0.475	0.625	0.645	1.280
2.	PREF	**0.326**	**0.333**	**0.480**	0.686
3.	SVR	0.542	0.438	0.606	**0.630**

baseline. More specifically, its MAE values are significantly higher than those of the majority baseline, but it beats the majority baseline in terms of MSE using both rating scales. This finding is quite surprising because SVR has been found to perform reasonably well for the rating inference task in related studies as aforementioned [114, 112, 41].

The unsatisfactory performance of SVR, as compared to the majority baseline, might be due to our experimental settings in which a small training set (1,500 reviews) was used for classifying a relatively large test set (5,743 reviews per test set on average). Another possible reason is that our training sets were designed to have uniform class distribution, which is quite different from the original class distribution of our dataset as Table 5.2 reveals. We conducted two additional experiments on SVR to investigate into these two issues, summarized as follows.

In the first experiment, we increased the size of training set from 1,500 to 4,800 while maintaining uniform class distribution. We then reran the rating inference experiments on PREF and SVR. Table 5.6 report the results from which we made three observations. Firstly, PREF still performs the best in the 3-point setting, and produces significantly lower MAE$_4$ than SVR. Secondly, SVR produces better results than the majority baseline except for MAE$_3$. Thirdly, referring to lines 2 and 5 of Table 5.5, both PREF and SVR perform better using a larger training set, and improvements are more significant in the 4-point setting. We conclude from these observations that PREF generally outperforms SVR, and that the unsatisfactory performance of SVR in the previous experiment is partially due to the relatively small training sets we used for learning regression models.

In the second experiment, we examined the effects of class distribution on PREF and SVR. Each training set in this experiment contains 1,500 randomly

Table 5.7: Comparison with SVR: Each training set contains 1,500 reviews, and retained the original class distribution of the dataset.

		3-point		4-point	
	Algorithm	**MAE**$_3$	**MSE**$_3$	**MAE**$_4$	**MSE**$_4$
1.	*Majority baseline*	0.475	0.625	0.645	1.280
2.	PREF	**0.342**	**0.349**	**0.487**	**0.691**
3.	SVR	0.449	**0.333**	0.599	**0.679**

sampled reviews and retained the original class distribution of our dataset. We conducted our rating inference experiments on the new training sets, and provided the results in Table 5.7. We made two observations from the results. Firstly, PREF performs the best in all aspects, although its MSE values are statistically indistinguishable from those of SVR based on Wilcoxon tests ($p < 0.05$). Secondly, by comparing Tables 5.5 and 5.7, we observed that SVR is much more sensitive to the class distribution of the training sets than PREF is.

To conclude, PREF generally outperforms SVR with a linear kernel for our rating inference task under identical experimental settings. Nonetheless, SVR performs well, especially in terms of MSE, if regression models are learned from training sets that can reliably reflect the class distribution of the underlying data.

Graph-based method of Goldberg and Zhu

Goldberg and Zhu [41] proposed a graph for modeling the rating inference task, and a closed-form solution to the optimization problem described in Eq. (2.34) in Chapter 2.5.4 (page 52). Their method extends the metric labeling approach of Pang and Lee [114] by supporting transductive learning as aforementioned. It was found to outperform Pang and Lee's approach when the number of labeled training reviews were limited. We implemented the closed-formed solution of Goldberg and Zhu and applied it to our dataset. Note that our current work only uses labeled reviews for training, thus we are actually solving the optimization problem described in Eq. (2.35).

We now describe the implementation of Goldberg and Zhu's work [41]. We computed the initial predicted ratings of test reviews ($\{\hat{r}_i\}$) using SVR as in [41]. Given the experimental results on SVR reported in the previous subsection, we used training sets that retained the original class distribution of our dataset for

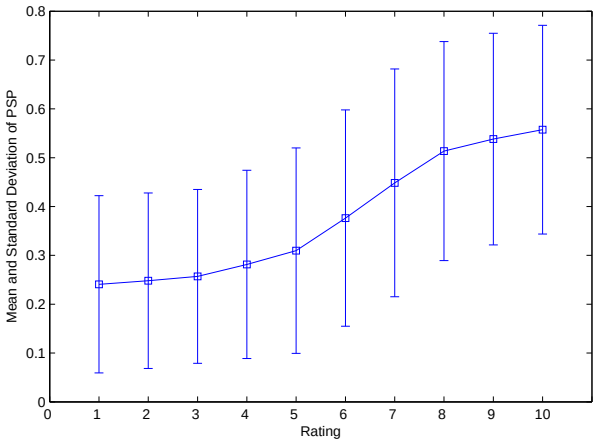

Figure 5.4: Mean and standard deviation of PSP of reviews having different ratings in our dataset.

this experiment. We computed $sim(d_i, d_j)$, the similarity between reviews d_i and d_j, based on PSP [114]. To determine PSP, we first trained a NB classifier based on the "subjectivity dataset v1.0" in [113] for retaining subjective sentences in reviews. We then trained another NB classifier based on the "sentence polarity dataset v1.0" in [114] for identifying positive sentences so that PSP can be determined. $sim(d_i, d_j)$ is computed as the cosine angle between the two vectors (PSP$_i$, 1-PSP$_i$) and (PSP$_j$, 1-PSP$_j$), where PSP$_i$ denotes the PSP of review x_i. Figure 5.4 shows the relationship between the PSP and user-specified ratings of the reviews in our dataset.

Previous work found that the user-specified ratings and PSP of reviews tend to be positively correlated [114, 41]. We observed from Figure 5.4 the same qualitative result, but apparently with greater standard deviation than those observed in [114, 41]. This may be attributed to the mixed-author setting of our dataset (recall that the four corpora used in [114, 41] were author-specific).

We now provide the values of the parameters $\mathcal{M}$, k and α. $\mathcal{M}$ is the weight assigned to labeled training reviews, and we set its value to 10^8 as in [41]. k is the size of nearest neighbors of test reviews, while α is the weight assigned to such neighbors. The test-set-optimal values, which we tuned empirically, of k and α are respectively 5 and 0.1.

Table 5.8 reports the results of this experiment. The graph-based approach always produces the lowest MSE values. PREF, however, performs the best in terms of MAE, and its MSE$_3$ is statistically indistinguishable from that of the

Table 5.8: Comparison with the graph-based approach [41] to rating inference: Each training set contains 1,500 reviews, and retained the original class distribution of the dataset.

		3-point		4-point	
	Algorithm	**MAE**$_3$	**MSE**$_3$	**MAE**$_4$	**MSE**$_4$
1.	*Majority baseline*	0.475	0.625	0.645	1.280
2.	PREF	**0.326**	**0.333**	**0.480**	0.686
3.	Graph-based approach	0.390	**0.322**	0.552	**0.647**

graph-based approach. These results are very encouraging, as they indicate that the performance of PREF is comparable to or better than that of the graph-based model in most cases despite its simplicity.

5.5 Integrating PREF and CF

We pointed out that integrating sentiment analysis and CF offers two advantages. Firstly, it is possible to perform CF even when user-specified ratings are not available, and secondly, it addresses data sparseness by using reviews as an additional source of user preferences for CF. We validated our ideas experimentally in this section. Note that it is possible to integrate other rating inference algorithms with CF, or integrate PREF with other CF algorithms. In this section, however, we only present results based on PREF and the classical user-based CF for two reasons. Firstly, we have already validated the robustness of PREF against related algorithms as discussed in the previous section. Secondly, we would like to keep the focus of this work on rating inference. We therefore decided to integrate PREF with the classical user-based CF model, which is well-acknowledged to be a promising CF model in the literature.

The experiments were set up as follows. We first sampled three datasets, denoted by DS1, DS2 and DS3, from our movie review dataset:

- **DS1**: contains around 15k reviews given by 480 users on 1,415 movies, with a sparisty level of 97.7%. All reviews in this dataset have user-specified ratings. This dataset was used to show the performance of classical CF.

- **DS2**: contains the same reviews in DS1, but the ratings of the reviews were

predicted using PREF. This dataset was used to evaluate the recommendation quality of predicted ratings. Note that when using this dataset for performing CF, the actual ratings given by the users in DS1 are the ground truth for performance evaluation.

- **DS3**: contains around 4.5k reviews given by users in DS1, but no user-specified ratings were available in the original reviews. The ratings in this dataset were therefore predicted using PREF. This dataset was used to complement DS1 to demonstrate the effect of augmenting ratings for existing CF by rating inference.

We then used the above datasets for performing CF. We adopted the best known user-based CF model [127, 14] in the experiments to keep the focus of this article on rating inference. Recall that user-based CF predicts the rating of a given item for a given active user in three steps. They are similarity weighting, neighbor selection and prediction computation. The similarity weighting step requires all users in the dataset to be weighted according to their similarity with the active user. Similarities are reflected in the ratings that users have given items. Pearson correlation coefficient (Eq. (2.1) on page 20), among others, is the most popular similarity measure studied in the CF literature, and is therefore adopted in our experiments. The neighbor selection step selects a number of k-nn, who are users having the highest similarity weights, of the active user as item predictors. Finally, the prediction computation step computes the item's predicted rating based on some partial information of the active user and the interests of his/her k nearest neighbors.

We adopted the Collaborative Filtering Engine (CoFE)[9] for testing as it provides a well-tested implementation of k-nn with Pearson correlation coefficient. Specifically, we prepared our datasets in the format required by CoFE and fed the datasets into it. The default value of k was 50 in CoFE. The choice of such value is consistent with the empirical findings of Herlocker et al. [55], which suggest that setting k to 20 to 50 would produce reasonable performance in real-world applications.

We reported the experimental results based on 5-fold cross validations using three evaluation metrics. They are MAE, MSE and coverage (COV), which is the percentage of items in the test sets for which CF predictions can be made.

[9]CoFE Collaborative Filtering Engine: http://eecs.oregonstate.edu/iis/CoFE

Table 5.9: Performance achieved using different datasets for performing CF.

	3-point			4-point		
Dataset	**MAE**$_3$	**MSE**$_3$	**COV**$_3$	**MAE**$_4$	**MSE**$_4$	**COV**$_4$
1. *Average baseline*	0.513	0.496	**100%**	0.722	1.098	**100%**
2. DS_1	**0.436**	0.430	74.3%	0.623	0.949	74.3%
3. DS_2	0.477	0.479	87.6%	0.680	1.003	87.9%
4. DS_1 + DS_3	**0.429**	**0.391**	92.3%	**0.599**	**0.836**	91.7%

We report MAE and MSE but not decision accuracy metrics such as precision and recall because the user-based k-nn algorithm generates numerical predicted ratings as output. Before we describe our results, we point out that the purpose of this set of experiments on CF is to predict users' preferences for items they have not yet observed based on their known preferences. This should not be confused with the purpose of our rating inference experiments, which is about evaluating how accurate various algorithms can predict the user-specified ratings of reviews based on the reviews' contents.

Table 5.9 reports the performance achieved using the three datasets described earlier in this section (lines 2-4), and that of the *average baseline* algorithm which always assigns the average rating obtained by a given item as its predicted rating for the active user (line 1)[10]. As shown in the table, using DS1 produces better CF predictions than using DS2 in terms of MAE and MSE. This is reasonable as ratings in DS1 were user-specified. Using only PREF-predicted ratings for performing CF (DS2) nonetheless produces MAE$_3$, MAE$_4$ and MSE$_4$ that are significantly lower than those of the average baseline. This seems to suggest that rating inference is a practical enabling technique to CF in case only reviews, but not user-specified ratings, are available.

Using DS3 to complement DS1 always produces the best results in terms of MAE and MSE. The improvements it can bring to the performance achieved using DS1 alone are more significant in the 4-point setting. Further, it raises COV to approximately 92% using both rating scales, as a result of increasing the density of the user-item ratings matrix of DS1. These results are very encouraging as they support our idea that rating inference is a useful technique

[10]We experimented with three baseline algorithms, including the majority baseline, the average baseline and the random baseline, in our preliminary experiments. The average baseline produced the lowest MSE values, and is therefore adopted for benchmarking.

for addressing data sparseness by augmenting ratings for CF.

5.6 Summary

This chapter describes our work on integrating sentiment analysis and CF. We collected and analyzed a movie reviews dataset as part of our study. We made several observations from the use of opinion words in user-generated reviews. Based on our observations, we identified the weaknesses of existing semantic-similarity-based methods for determining the SO of opinion words, and proposed a relative-frequency-based method for performing such task. In addition, we proposed and described a probabilistic rating inference framework, known as PREF, which determines the overall SO of a review based on the SO of the opinion words it contains. We conducted extensive experimental studies for validating the effectiveness of PREF, which shows superior performance to various related algorithms. Further, our results suggest that PREF does not rely on a large training corpus to function, which can be an important concern when applying sentiment analysis to new domains where labeled (rated) reviews are limited.

We also studied the effectiveness of PREF in augmenting ratings for CF. Rating inference is a task that transforms user preferences expressed as unstructured, natural language texts into scalar ratings. This enables a direct integration of sentiment analysis and CF, allowing CF to utilize textual reviews as a source of user preferences. We demonstrated the advantages of this by integrating PREF with classical user-based CF algorithm. Specifically, we made use of the predicted ratings of reviews generated by PREF for performing CF. Encouraging results were observed: the predicted ratings produced reasonably good CF predictions, and PREF improved the performance of CF significantly by augmenting ratings from reviews.

We identified two issues to consider when enhancing our work on generating personalized recommendations using user-generated reviews. Firstly, PREF was tested on the movie domain, where plenty of reviews associated with user-specified ratings are available on the Web and where CF has been widely studied and adopted. We would like to base our continuing work on other domains, such as tourist attractions, where numerous user-generated reviews are available but CF-based recommendations are not common.

Secondly, rating inference can be considered a technique for summarizing the overall sentiments of reviews and then representing the sentiments as scalar ratings. This enables a straightforward integration of sentiment analysis and CF at the expense of detailed preference information. For instance, user preferences for specific item features can be extracted from the reviews, and can potentially be useful for generating personalized recommendations.

We continue our discussions on these issues in the next chapter.

Chapter 6

Towards Review-based Recommender Systems: A Case Study on TripAdvisor

6.1 Introduction

In the previous chapter, we proposed a direct and simple method for utilizing user-generated reviews for CF. Specifically, we proposed a rating inference framework for determining and representing the overall sentiments expressed in reviews as numerical ratings. Such ratings can be used by existing CF algorithms, thereby facilitates a direct integration of sentiment analysis and CF.

This chapter continues to describe our effort on integrating sentiment analysis and CF. However, instead of simply augmenting ratings from reviews for CF, we attempt to make use of the detailed user preferences and item features expressed in reviews for making personalized predictions. Our work is motivated by our conjecture that such detailed information are more precise and valuable descriptions of users preferences, and the fact that they have not yet been utilized for making personalized recommendations or predictions for users. We designed a set of prediction models along the idea of CF. Those models are designed to utilize the contents of user-generated reviews for enriching the interest profiles of users, items and categories to different extents. While the review-based prediction models make use of review contents for generating recommendations, our work is different from pure content-based recommender systems, which do not involve the sentiment analysis of reviews (or textual contents in general).

We base our study on a set of tourist attraction reviews. As described in the previous chapter, we would like to extend our study to a new item domain. This is motivated by two factors. Firstly, we have already achieved successful results on the movie reviews domain, which is the most well-studied domain in both sentiment analysis and CF. Secondly, numerous user-generated reviews are available for travel-related products, but CF-based recommendations for such products are not common.

The major contributions of this study are two-fold. Firstly, we explored the use of review contents for deriving similarities between items and item categories, and the use of the SO (sentimental orientation) of opinion phrases extracted from reviews for modeling user preferences. Empirical results suggest that review contents do contain valuable information that can be used for making personalized recommendations for users. Secondly, recent user studies have been conducted on the use and effects of travel reviews in the decision making progress of travelers when planning their trips [131, 46], but our study is the first one reporting empirical research on how travel reviews can actually be used for performing personalization.

We point out that our study is not concerned with the travel decision making process of customers. Instead, our focus is on making personalized predictions based on user-generated travel reviews. We further point out that the results of our work is not limited to travel reviews. We chose this domain for our study mainly because research on personalized recommendations for travel-related products is not common, and most travel recommender systems are knowledge-based with a limited degree of personalization [128, 129]. We hope our work can shed new light on review-based recommender systems research in general.

The remainder of this chapter is organized as follows. Chapter 6.2 reviews existing studies that are related to our work. Chapter 6.3 describes user-generated reviews on TripAdvisor[11]. It also details our data collection process and the characteristics of our dataset. Chapter 6.4 describes the tasks involved in performing review-based personalization. They include the sentiment analysis of reviews, the construction of user and item profiles based on review contents and opinion phrases, as well as the prediction models we used for making predictions users. We discuss experimental results in Chapter 6.5, and summarize our findings in Chapter 6.6.

[11]TripAdvisor: http://www.tripadvisor.com

6.2 Relation to Other Work

We noticed two studies in the literature that involve the use of user-generated reviews in recommender systems. The work of Wietsma and Ricci [167] incorporates user-generated reviews into a mobile recommender system of tourist attractions. Their work generates *content-based* recommendations for tourist attractions, and displays reviews written by the nearest neighbors of the active user *as a decision making aid*. Specifically, their work utilizes reviews to help explain the recommendations made by their content-based algorithm. It does not generate item recommendations from review contents as our work does.

The Informed Recommender described in [2, 3] attempts to analyze the contents of user-generated reviews for generating recommendations. Its focus is, however, on mapping review contents onto a manually defined ontology of domain items, which are digital cameras. Such ontology defines the vocabulary and relationships between words to describe the following information:

- Users' skill levels in using the digital cameras they reviewed, such as *Beginner* or *Professional*
- Item features that might appear in reviews, such as the *Lens* and *Flash* of cameras.

The Informed Recommender requires the active user to specify the item of interest (a specific camera model), and select the item features that they are most interested in when requesting recommendations. It then retrieves the reviews whose contents are relevant to the specified item or item features with the support of the ontology, and makes a recommendation based on what other users said about that item or its features.

Our work is different from the Informed Recommender in two major aspects. Firstly, the success of the Informed Recommender depends heavily on the mapping of review contents onto the predefined domain-specific ontology. As we discuss in Chapter 6.3.3, such a solution that relies on well-defined ontology or concept hierarchies may not be suitable for review-based recommendations due to the heterogeneity of domain items. Our work therefore applies sentiment analysis techniques to automatically identify interesting features from review contents based on linguistic features, and utilizes such features for making personalized item predictions. Secondly, the Informed Recommender *does not*

take the past experience of the active user into consideration when generating recommendations. On the contrary, we designed review-based models that make predictions for users based on their previously expressed preferences, as well as on the similarities between items and item categories derived from the contents of user-generated reviews.

6.3 Travel Reviews on TripAdvisor

We chose to use reviews on TripAdvisor because it is one of the most prominent providers of user-generated travel reviews and ratings. It provides more than 15 million travel reviews contributed by more than 6 million registered members as of 2007 [154]. Reviews submitted to TripAdvisor are examined by trained personnel before they are publicly posted on the site.

TripAdvisor allows users to write reviews on hotels, restaurants, cruises, cities or towns, as well as tourist attractions (Things to Do). Among these five types of reviews, attraction reviews offer much richer, but fuzzy, information because "attraction" itself is a fuzzy domain that is not well-defined and well-structured. In contrast, restaurants and hotels can be described using more objective characteristics, such as the price range and type of cuisine of a given restaurant, and the availability of a swimming pool in a given hotel [131]. We therefore decided to base our study on attraction reviews. We expected that attraction reviews are more challenging to handle, and may reveal new research issues for the sentiment analysis of user-generated reviews, as well as for generating review-based recommendations.

To the best of our knowledge, there is no existing work on generating personalized recommendations with the support of sentiment analysis of attraction reviews. The work of Wietsma and Ricci [167], as aforementioned, only displays a set of reviews written by neighbors of the active user as a decision making aid. We take a data-centric approach in this study, which is experimental in nature, due to the lack of precedent literature on the task. Figure 6.3 depicts an attraction review on TripAdvisor. Each review contains a title, a numerical rating (in a 5-point integer scale), the name of the user who wrote the review, the user's location, the date of the review, the number of users who found the review useful, and the content of the review. Note that information about the attraction concerned is not shown in the figure, but is available on the webpage containing

Figure 6.1: A user-generated travel review on TripAdvisor.

that review on TripAdvisor.

In what follows, we detail our data collection method, and describe the major entities and their relationships in our dataset. We then discuss several characteristics of our dataset, and their implications for our work on generating review-based recommendations.

6.3.1 Data Collection and Filtering

We collected a set of reviews on attractions in the United States for this study. Data collection was done in two rounds:

1. We started by crawling all attraction reviews (as of April 22, 2008) on the 20 most popular cities in the USA. This dataset contains a total of 15,696 reviews by 11,355 users on 2,401 attractions. For experimental evaluation to be possible, active users must have contributed at least two reviews, so that we can have at least 1 review for training and 1 review for testing. Only 6,236 reviews written by 1,895 satisfied this constraint.

2. We then collected reviews on other USA attractions that have been reviewed by the 1,895 users. After merging this set of reviews with that collected in the previous round, we got a total of 2,085 users who contributed at least two reviews. The resulting dataset contains 8,637 reviews on 2,767 attractions.

We performed the following data filtering on the dataset. Firstly, we discarded reviews written by users who contributed only one review as noted to

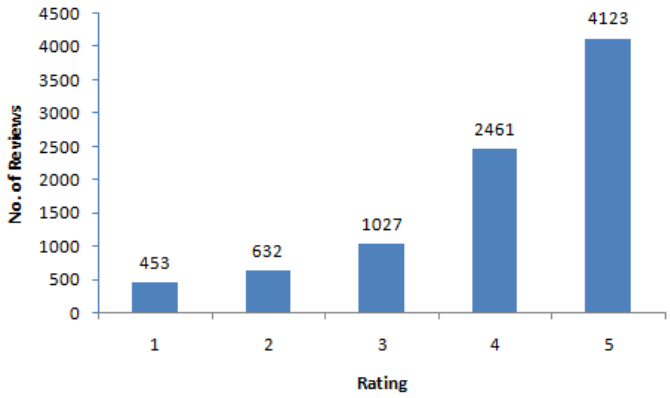

Figure 6.2: Distribution of ratings in our attraction reviews dataset.

facilitate our experimental study. Secondly, we removed reviews on attractions without a specific attraction type. Specifically, all attractions on TripAdvisor are classified under a *attraction type* taxonomy. We removed attractions that belong only to the attraction type "Other".

The finalized dataset used in this study contains 8,696 reviews by 2,074 users on 2,741 attractions, which belong to a set of 148 distinct attraction types. This yields a sparsity level of 99.85%. Figure 6.2 shows the distribution of user-specified ratings (associated with the reviews) in our dataset. The distribution is highly skewed: around 47% (4123/8696) of reviews were assigned the highest rating of 5, whereas only around 5% were assigned the rating of 1.

6.3.2 Data Model

We now provide a formal data model of our dataset. The data model contains three major types of *entities*, namely users, items (attractions) and categories of items (attraction types), as well as the relationships between them.

Formally, there are three sets of entities, including a set of users U, a set of items I, and a set of categories G. There exists an active user $a \in U$ who seeks prediction for a target item $t \in I$ as in CF. Each item $i \in I$ can be mapped to one or more categories in G, and G_i denotes the set of categories to which item i belongs. I_u denotes the set of items reviewed by user u. A review document $d \in D$ captures the relationships between a user u and an item i. Each document d is conceptually a four-tuple: $d = \langle u, i, r_{u,i}, \overrightarrow{d} \rangle$, where $r_{u,i}$ denotes the user-specified rating associated with the review, $\overrightarrow{d}$ represents a

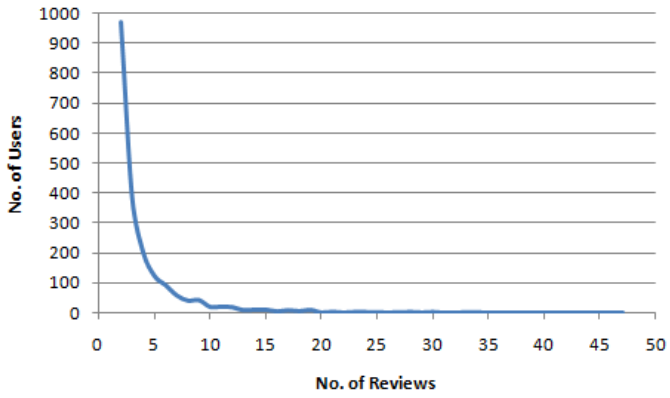

Figure 6.3: Distribution of review count by user.

feature vector representation of d. Each dimension of $\overrightarrow{d}$ corresponds to an item feature extracted from review. Its value is the user's *vote* for the corresponding item feature. We defer the description of the possible methods for determining users' votes for item features to Chapter. 6.4.2.

6.3.3 Data Characteristics and Implications

We observed the following characteristics from our dataset:

Data sparseness

After performing the first round of data collection, we immediately observed that the dataset was extremely sparse. Data sparseness is a well-known issue in CF, and has been addressed usually various approaches as noted. Given the fact that we are already considering the 20 most popular cities in the USA, one shall expect that most attractions in less popular destinations would have never been reviewed by any user. We may turn a blind eye to this problem by performing more data filtering, for instance, by restricting our experiments on users who have contributed a considerable amount of reviews, and on attractions that have received a certain number of reviews. However, severe data sparseness is a realistic issue that must be addressed in a practical solution.

Figures 6.3 and 6.4, both exhibiting a power law distribution, help illustrate the sparseness of our dataset. Figure 6.3 shows the distribution of review count by user. Around 47% (974/2074) of users have provided only two reviews, and approximately 81% of users in total provided not more than five

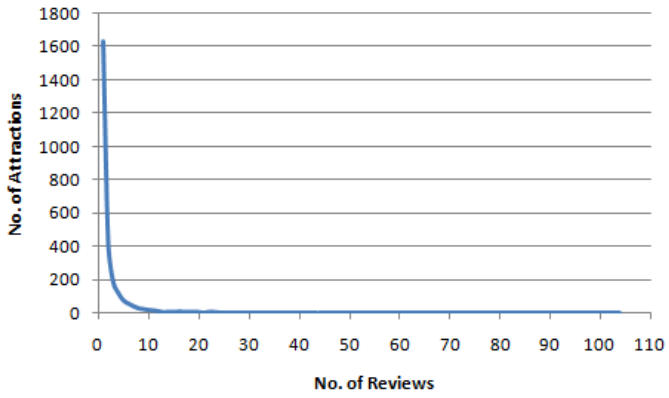

Figure 6.4: Distribution of review count by attraction.

reviews. Figure 6.4 shows the distribution of review count by attraction. Almost 60% (1629/2741) of attractions have only been reviewed once. This has two implications. Firstly, classical and promising CF methods operating on co-rated items would not work for most items in our dataset as shown empirically in Chapter 6.5.4. Secondly, for a solution to be useful, it must be able to generate predictions or recommendations for cold-start items.

We point out that it is possible to apply FARAMS and CLARE for addressing data sparseness and the cold-start problem to our tourist attractions dataset by using attraction types as the concept hierarchy of items. It may also be possible to augment ratings for performing CF from additional travel journals without user-specified ratings using PREF. However, we identified the need to propose new paradigms for review-based recommendations, as discussed in the next subsection.

Describing heterogeneous domain items

Most studies on recommender systems deal with only one type of domain items, such as movies, CDs and books. While these items may be classified into categories, they can still be described using a common set of attributes. Such attributes are defined by domain experts or information providers rather than by end-users.

We found that defining a common set of attributes for describing tourist attractions is very difficult, if not impossible. While one may consider "tourist attractions" to be a kind of domain items, different types of attractions may possess very different characteristics. For instance, commonly-described features

of a *state park* include "dam" and "view", whereas those of a *museum* include "exhibit" and "collection". In other words, the information structure of domain items is not well-defined.

Note that features mentioned in user-generated reviews and item attributes defined by domain experts are different in nature. The set of attributes of an item is "globally the same" for all users in a given system. In contrast, different users may comment on different features of the same item in user-generated reviews, as in the case of user-assigned tags [155]. Further, item attributes are usually concrete, explicit and objective properties of domain items that can be easily identified. Using the movie domain as an example, attributes for describing movies on IMDb include their as cast, directors, awards, and genres. A study on sentiment analysis of movie reviews, however, reveals that features extracted from movie reviews can be classified into more than 30 different aspects [176], including those that are *not likely to be considered and defined as attributes* of movies. Examples are the "editing" and "social implications"of movies.

The heterogeneity of domain items, as well as the differences between item attributes and features, have two implications for review-based recommender systems research. Firstly, instead of looking for a common set of attributes for describing heterogeneous items (attractions), a more feasible solution would be to identify interesting features of the attractions directly from the reviews. Secondly, recommender systems that operate on well-defined concept hierarchies and ontology may not be suitable for review-based recommendations. While such systems, including our own work (FARAMS and CLARE) and the Informed Recommender [3], are interesting contributions in their own right, there is a need to develop novel recommendation paradigms that can address the characteristics and fuzziness of information in user-generated reviews.

High-dimensional, evolving feature set

User-generated reviews are free-form texts, and users can comment on any features about the subject matter concerned in reviews. This, together with the fact that new user-generated reviews are added to the web everyday, we shall expect the feature set being mentioned in a reviews dataset to be ever evolving and expanding. We shall therefore pay attention to the expansion of the feature set as more reviews are added to the system. Further, feature selection or dimensional reduction techniques may be essential to help control the complexity

of the task at hand.

6.4 Generating Item Predictions from Reviews

We address the task of generating rating predictions for items from user-generated reviews in this study. The experimental setup of our study emulates a real world situation when a registered user (active user a) of, for instance, TripAdvisor, clicks on a hyperlink to web page of a certain attraction (target item t) (s)he has not reviewed or rated before, and our task is to predict $p_{a,t}$ to indicate how much user a may like item t.

Generating rating predictions from reviews can be decomposed into three major tasks, involving two different areas of research. The first task is related to sentiment analysis. It aims at identifying and analyzing interesting features from reviews for the subsequent tasks. The second task is the construction of user, item and category profiles, based on which personalized predictions are made. The third task is concerned with CF, which makes predictions for users. We now detail the three tasks in the following subsections.

6.4.1 Sentiment Analysis of Reviews

Figure 6.5 depicts the tasks involved in the sentiment analysis of reviews. The whole analysis process starts with a collection of reviews, which we collected from TripAdvisor in this study. It then preprocesses the reviews, such as removes the HTML tags in the downloaded reviews. The feature extraction task aims at identifying interesting features, including both features of items and opinions of users, based on various linguistic processing techniques. This task represents the contents of each review, which were free-form texts, as a list of "feature:opinion" pairs. The extracted features and opinions are then analyzed, and are respectively stored in the feature database (Feature DB in the figure) and the opinion dictionary of the system. They are used to facilitate the construction of user, item and category profiles in the next step.

We now describe our methods for extracting interesting features and building the opinion dictionary in detail.

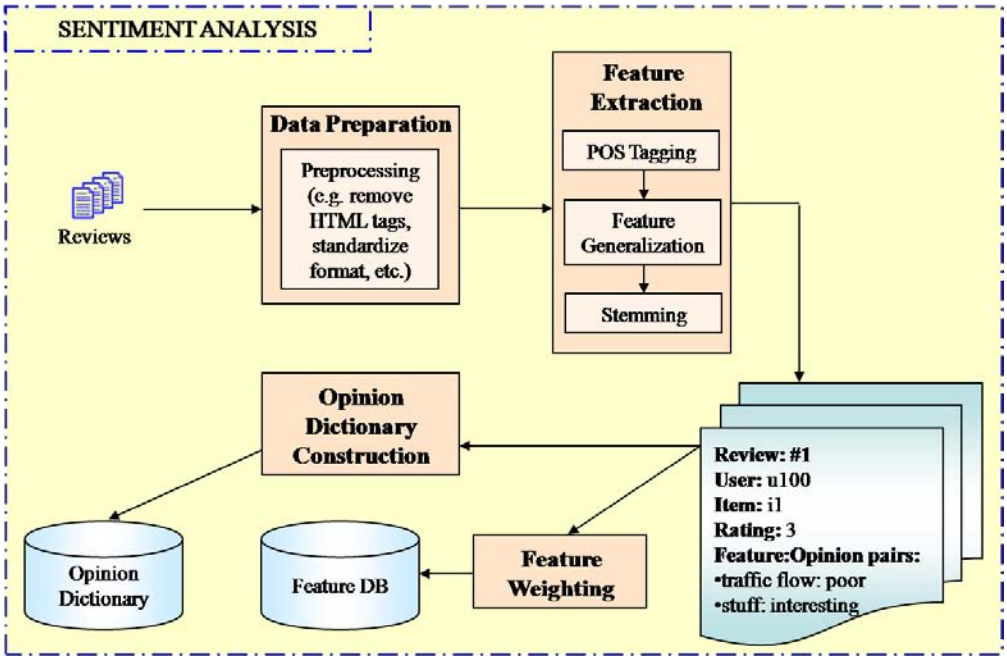

Figure 6.5: Tasks in the sentiment analysis of user-generated reviews.

Extracting features

There are three possibles approaches for extracting interesting features from reviews. The first approach is an extensive domain engineering approach. It relies on domain experts to define interesting features of each attraction or attraction type. This approach can be very expensive due to the heterogeneity of attractions. It may also be infeasible in practice due to the tremendous number of attractions available on TripAdvisor, or items in general in other review portals. The second approach is to perform statistical analysis or data mining based on training data for identifying interesting features. Hu and Liu [62], for example, applied the association rule mining technique to item features and opinions extraction. We did not consider this approach in view of the sparsity of data. Further, this approach is generally applied to and tested on one single type of domain items. It might not be suitable for the tourist attraction domain due to the heterogeneity of domain items. The third approach performs extraction based on POS information. We adopted this approach as it is a more general approach that is less dataset-dependent. Note that POS tagging, however, is a language-dependent task. Our current work deals with reviews written in English.

We reused some of the review analysis techniques used in PREF for this study. They include POS tagging and feature generalization. Recall that we

only considered unigrams (isolated adjectives and nouns) in PREF. In this work, we slightly modified the feature extraction module of PREF to extract n-grams, which are n adjacent tokens in a sentence.

We first applied POS tagging to our dataset, and then performed feature generalization, which aims at generalizing features that may be overly specific. The feature generalization step identifies and adds a special tag to noun phrases that represent the names of the attractions being reviewed. We excluded such generalized features in the feature extraction process, because they are not features that describe the characteristics of attractions. For instance, while the proper noun "Museum" refers specifically to the museum being reviewed, it is obviously not a feature that describes the characteristics of the museum.

After performing POS tagging and feature generalization, we applied a set of extraction rules to look for noun phrases, which are considered to be item features, and opinion phrases in the reviews. The extraction rules were defined based on findings of previous work on sentiment analysis plus some heuristics. For opinion phrases, we considered not only isolated adjectives, but n-grams that consist of adjectives, preceded by zero or more adverbs. For instance, the phrases "good", "very good" and "not good" are all opinion phrases. We did not explicitly perform negation tagging as n-grams extraction might be an alternative way to capture the contextual effects of negation words. For item features, we extracted n-grams, where $n \leq 3$, made up of nouns or proper nouns, but excluded terms that are generalized features as aforementioned. We stemmed the item features identified using the well-known Porter Stemmer [122]. For instance, the word "waterfalls" is reduced to "waterfall" in the stemming process. This helps identifying common item features among attractions.

This task extracts a list of item features, $F_d = \{f_1, f_2, f_3, ..., f_n\}$, and a list of opinion phrases $V_d = \{v_1, v_2, v_3, ..., v_n\}$ for a review d. An opinion phrase v_i is the nearest opinion phrase that is found in the same sentence from which f_i is extracted. If no associated opinion phrase can be identified for f_i, then the corresponding entry v_i in V_d is null. The lists F_d and V_d are used in the next tasks for building user, item and category profiles.

Remarks on computational complexity: This sentiment analysis step has linear computational complexity. Specifically, the computational complexity of the POS tagging task is $O(|D|)$, where $|D|$ equals to the number of reviews and $|D| << (|U| * |I|)$ due to data sparseness. The feature generalization and

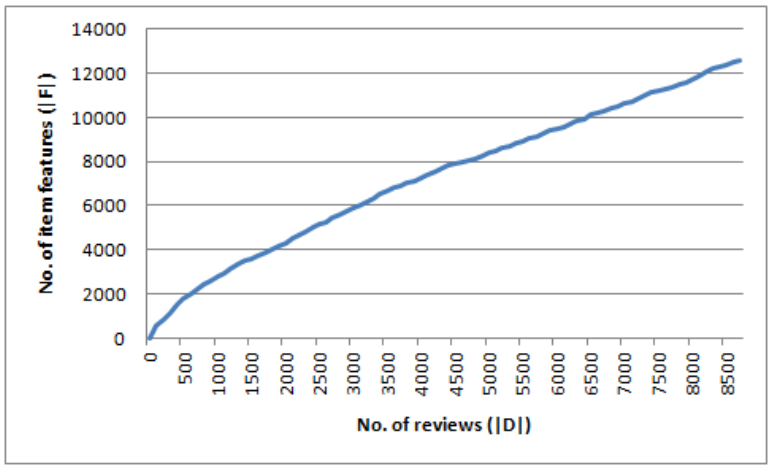

Figure 6.6: Number of distinct features versus number of reviews.

stemming tasks deal with $|D|$ reviews, each containing $|F|$ item features in the worst case. The computational complexity of both tasks is therefore $O(|D| \times |F|)$. Note that in practice, the number of item features that can appear in a single review $d \in D$ is small and limited. The longest review in our dataset contains only 152 appearances of item features.

We suggested in Chapter 6.3.3 that the feature set extracted from reviews shall be ever expanding. In practice, however, the size of the feature set should be *large but limited* to a certain extent for two reasons. The first reason is the fact that the usable vocabulary range of human has a practical limit. In fact, the number of distinct features in our dataset increases only as a linearithmic function of the number of reviews as Figure 6.6 shows. The second reason is that feature selection or filtering is commonly done in algorithms that deal with feature terms extracted from text documents. The size of feature set in the system is therefore controllable and adjustable. Note that performing feature selection or filtering not only helps maintain the scalability of the task, but also improves prediction accuracies. In Chapter 6.5.4, we confirm by experiments that using only a small proportion (for instance, 1%) of the best features for building user, item and category profiles produces better prediction accuracies than using the entire feature set.

Building the opinion dictionary

The **opinion dictionary** contains the set of *opinion phrases* extracted from reviews, and their *frequency counts* in each rating class $c_j \in C$. Figure 6.7

Opinion\Rating	1	2	3	4	5
interesting	19	30	111	219	169
excellent	5	13	26	107	284
stunning	1	0	5	19	71
poor	15	13	11	10	14
terrible	28	16	10	8	9
.....	...	...	...	...	...

Figure 6.7: Illustration of the contents of the opinion dictionary.

is an illustration of the contents of the opinion dictionary built from our dataset. Rows and columns in the opinion dictionary are respectively opinion phrases and rating classes, while elements in the opinion dictionary denote frequency counts of opinion phrases in the various rating classes. The SO of opinion phrases are computed from the counts. The reason for keeping the frequency counts instead of the SO of opinions is that when new reviews are added to the system, we can update the counts of existing opinion phrases incrementally without having to rebuild the entire dictionary. Further, new opinion phrases can also be added to the opinion dictionary easily. We now describe the method for computing the SO of an opinion phrase from its frequency counts.

We proposed a relative-frequency-based approach to SO and opinion strength determination in Chapter 5.3.3. In our original approach, we allow an opinion word to have multiple SO, each with a corresponding opinion strength. In other words, we compute the opinion strength of an opinion word v_i with respect to each sentiment class c_j, denoted by $OS(v_i, c_j)$, as a local measure. This is designed to facilitate the rating inference task, which aims at assigning an overall rating to a review in a fine-grained rating scale. The goal of this work is different from that of rating inference, as we are interested in the *overall SO* of an opinion phrase as an indication of a user's opinion. Therefore, we adapted our proposed relative-frequency-based method in PREF to compute $SO(v_i)$, which is the SO of v_i with respect to the entire training corpus, as follows:

$$SO(v_i) = \frac{\sum_{j=1}^{|C|} c_j * N(v_i, c_j) * w(c_j)}{\sum_{j=1}^{|C|} N(v_i, c_j) * w(c_j)} \tag{6.1}$$

Note that c_j in rating inference and in this work is a numerical value (in a 1 to 5 scale in our dataset). $N(v_i, c_j)$ is the frequency count of v_i in c_j, and $w(c_j)$

is a weight used to factor out the effect of uneven class (rating) distribution in the dataset. Our dataset contains 453 reviews rated as 1 and 4123 reviews rated as 5 (Figure 6.4). The same opinion phrase is therefore expected to appear more (resp. less) frequently in reviews rated as 5 (resp. 1), regardless of the importance of the phrase in those reviews. $w(c_j)$ is defined as:

$$w(c_j) = \alpha \frac{\sum_{k \in C} N(V_k)}{N(V_j)} \tag{6.2}$$

where $N(V_k)$ denotes the total frequency count of all opinion phrases in V_k, and α is a normalizing factor such that the set of $w(c_j)$ for all $c_j \in C$ sums to unity.

6.4.2 Building User, Item and Category Profiles

User, item and category profiles can be built from user preferences data, in terms of user-specified ratings, item features, as well as SO of opinion phrases. When utilizing only user-specified ratings, however, the resulting interest profiles of the three types of entity are simply ratings matrixes used in classical CF. Our discussions therefore focus on how to build enriched entity profiles based on review contents in the following.

User profiles

It is possible to build user profiles based on the set of item features extracted from reviews. In such case, a feature vector representation of the interest profile of user u, denoted by $\overrightarrow{u}$, is an n-dimensional vector, with each dimension corresponds to an item feature. The element of the j^{th} dimension is a weight $w(f_j, u)$ that reflects the interestingness of the item feature f_j with respect to user u. There are three methods for determining $w(f_j, u)$. The first method determines $w(f_j, u)$ by the *frequency count* of f_j in the reviews contributed by user u. This mainly reflects the fact that such feature has been *mentioned* by user u in reviews. We did not consider this method because it does not capture any explicit preferences expressed by the user.

The second method weights a feature f_j appeared in a review d based on the *user-specified rating* ($r_{u,i}$) associated with that review. In other words, all features appeared in the same review are given the same weight. We illustrate this with an example. The following sentence is extracted from a review in our dataset:

"*Poor traffic flow when crowded, but interesting stuff.*"

Our feature extraction method described in the previous subsection identified two feature-opinion pairs from the above sentence: "traffic flow-poor" and "stuff-interesting". The user-specified rating of the original review is 3. This method therefore weights both opinion phrases "poor" and "interesting" based on the rating of 3. If a feature f_j is mentioned in different reviews, we take the average of all ratings it obtained in all reviews written by user u.

The third method weights f_j based on the *SO of the opinion phrase* v_i associated with it. If no associated opinion phrase can be identified for it, then it is weighted based on $r_{u,i}$. Different item features in the same review may be assigned different weights using this method. Referring to the previous example, the estimated SO of "poor" is 1.8 while that of "interesting" is 3, computed using our SO determination method described in Eq. (6.1) and Eq. (6.2). Note that the same item feature may appear in the same review, or different reviews written by the same user, more than once. In such case, we take the average of the weights obtained by f_j across all of its appearances in user u's reviews as the value of $w(f_j, u)$. Recall that one motivation of our work is that user-generated reviews contain detailed preferences information that are seemingly useful, but have not yet been utilized for performing personalization. We therefore believe, and demonstrate empirically, that using this method to build interest profiles of users is indeed useful, and is able to make more accurate predictions than using the overall user-specified ratings only (Chapter 6.5.4).

Item and category profiles

Item profiles are constructed for deriving item similarities, to be used for making predictions for items. In classical rating-based CF, the similarity between two items i and t is determined based on ratings they received from users who rated both items. In our work, we attempt to derive the similarity between the two items based on the item features users used to describe the items in reviews. Such a similarity measure can be considered to be content-based, but the content information about items comes from user-generated reviews rather than from item attributes defined by information providers. In other words, users collaborate to provide items features for constructing item profiles.

The profile of item i is constructed by aggregating item features appeared in

reviews on i. Similar to the case of user profiles, a feature vector representation of an item i, denoted by $\overrightarrow{i}$, is an n-dimensional vector. Each dimension corresponds to an item feature. As we are capturing content similarity between items rather similarity between user preferences, the weight $w(f_j, i)$ reflects the importance of the item feature f_j with respect to item i, and it is computed as the normalized frequency count of f_j in all reviews of i. Note that this representation of item profiles actually simplifies the original multi-dimensional relationship between an item, an item feature, and a user due to the simplicity of CF. Specifically, an item profile models an item by the set of features that have been used to describe an item, but does not consider which user uses which feature terms to describe it.

Category profiles are constructed in a similar way as item profiles. We aggregate all feature terms used to describe items belong to g in order to construct the profile of a category g.

6.4.3 Making Predictions

We experimented with eight prediction models in our study, summarized in Table 6.1. Each model is assigned a model number Mn for easy reference.

Table 6.1: Brief descriptions of prediction models.

Model	**Description**	**Preference Data**
M1	User-based CF.	$U \times I$
M2	Item-based CF.	$U \times I$
M3	Makes predictions based on user a's average rating on $g \in G_t$ (categories of item t).	$U \times G$
M4	Adapted from M2, computes item similarities based on review contents.	$U \times I + I \times F$
M5	Generalized version of M4, computes category similarities based on review contents.	$U \times F + U \times G + G \times F$
M6	Makes predictions based on based on user a's preferences for features of G_t.	$U \times F + G \times F$
M7	Majority baseline, not personalized.	$U \times I$
M8	Random baseline, not personalized.	–

We describe the design of Models M1 to M8 in the following subsections.

We also discuss how the models scale with the numbers of users ($|U|$) and items ($|I|$) in the system. The models can be classified into three types based on the preference data they used for making predictions: pure rating-based models (M1, M2, M3), review-based models (M4, M5, M6), and non-personalized models (M7, M8). Note that M7 and M8 are only baseline models to help indicate the effectiveness of the other models. All models produce a predicted rating $p_{a,t}$, a numerical rating showing how much the active user a may like the target item t, as output.

Rating-based Models

Rating-based models, as the name implies, are those that make predictions solely based on ratings users have given items. They operate on user preferences captured in the $U \times I$ ratings matrix.

Model M1 is equivalent to the classical user-based CF algorithm. It first computes similarity between users based on ratings data. It then selects the set of k-nn of the active user a as predictors for the target item t. This model depends heavily on the existence of co-rated items between the active user and his/her neighbors. Recall that the user-based CF algorithm predicts $p_{a,t}$ as (Eq. (2.2)):

$$p_{a,t} = \overline{r}_a + \alpha \sum_{u \in knn(a)} w(a,u)(r_{u,t} - \overline{r}_u)$$

where $\overline{r}_a$ is the mean rating user a has given items in the training data, $u \in knn(a)$ denotes the k-nn of a determined based on $w(a,u)$, which is the similarity weight between users a and u. α is a normalizing factor such that the absolute values of similarity weights w(a,u) sum to unity. $r_{u,t}$ denotes the rating user u has given item t.

We model user, item and categories as feature vectors as described in the previous subsection. It is therefore natural to compute $w(a,u)$ as the vector similarity between $\overrightarrow{a}$ and $\overrightarrow{u}$, which are the vector representations of the interest profiles of users a and u respectively. One widely adopted vector similarity measure is the cosine similarity measure:

$$w(a,u) = \cos(\overrightarrow{a}, \overrightarrow{u}) = \frac{\overrightarrow{a} \cdot \overrightarrow{u}}{||\overrightarrow{a}||_2 * ||\overrightarrow{u}||_2} \tag{6.3}$$

where the dot $(\cdot)$ indicates the dot-product of the two vectors. Note that in this model, feature vectors are constructed using user-specified ratings. In other

words, the j^{th} element in $\overrightarrow{a}$ is user a's rating on the j^{th} item, and similar for $\overrightarrow{u}$.

Model M1 needs to compute $|U| \times (|U| - 1)$ similarities between users in U, each potentially requires $|I|$ operations. The upper bound of the computation complexity of Model M1 is therefore $O(|U|^2 \times |I|)$. Due to data sparseness, however, the number of co-rated items between most pairs of users is very limited, meaning that the actual complexity required is much smaller than the specified upper bound. The rating prediction step of Model M1 has a complexity of $O(k)$ because it depends on the number of neighbors considered. Note that $k << |U|$ in reality. Herlocker et al. suggest that in a neighborhood size of 20 to 50 would produce reasonable performance in real-world situations [54].

As our experimental study is based upon a static dataset, we are able to break down the operations required by Model M1 into a model construction (similarity computation) phase and a rating prediction phase. In traditional user-based CF, however, the so-called model construction phase is done in real-time, which would add complexity to the rating prediction phase. Making a prediction for the active user a requires computing the similarity between the user and all other users, resulting in a computational complexity of $O(|U| \times k)$.

Model M2 is the item-based CF algorithm, which exploits the similarity between items based on the ratings they received from users. It makes a prediction based on how the active user has rated items that are similar to the target item t. The success of this model depends on whether the active user has rated the target item's neighbors. Recall that the item-based CF model predicts $p_{a,t}$ as (Eq. (2.4)):

$$p_{a,t} = \alpha \sum_{i \in knn(t)} w(t,i)(r_{a,i})$$

where $i \in knn(t)$ denotes the set of k-nn of item t, $r_{a,i}$ denotes the active user's rating on item i, and α is a normalizing factor such that the absolute values of similarity weights $w(t,i)$ sum to unity. Similar to Model M1, $w(t,i)$ is the similarity between items t and i, computed using the cosine similarity measure described in Eq. (6.3). Feature vectors in this model are also constructed using user-specified ratings, but with items taking the role of users, and vice versa. The j^{th} element in the feature vector $\overrightarrow{t}$ is the rating item t received from the j^{th} user in the training set.

The computational complexity of Model M2 in the model construction phase is $O(|I|^2 \times |U|)$ as it needs to compute $|I| \times (|I| - 1)$ similarities, each requires at most $|U|$ operations. As aforementioned, the actual complexity is much smaller due to the lack of co-rated items in user profiles. The complexity of the rating prediction step of Model M2 is $O(k)$.

Model M3 utilizes higher-level information about domain items, denoted by categories G, for making predictions. We chose to make use of categories information for two reasons. Firstly, both Models M1 and M2 depend heavily on co-rated items, which are not likely to exist for many users and items in a real-world application due to data sparseness. This also happens in our real-world dataset collected from TripAdvisor as noted. Precedent work in CF indicates that the use of item taxonomy, or is-a hierarchies, can lessen the problem of data sparseness and improve recommendation quality (e.g. [109]). We therefore considered the categories of items, which are *attraction types*, in this work. Secondly, we hope to maintain the applicability of our work to other domains in the future. Specifically, item categories exist in most application domains, although the mapping between items and their categories is domain specific. Note that we did not deal with the construction of item taxonomy in this work. We used the attraction types taxonomy collected from TripAdvisor as-is, although it is possible to refine its taxonomy by, for examples, defining sub-types or combining existing attraction types if necessary.

In Model M3, we first computed users' mean ratings for each category, given the rating matrix $U \times I$ and the mapping between items and categories. The prediction $p_{a,t}$ is then computed for user u_a as:

$$p_{a,t} = \frac{1}{|G_t|} \sum_{g \in G_t} \overline{r}_{a,g} \tag{6.4}$$

where G_t denotes the set of categories to which t belongs, and $\overline{r}_{a,g}$ is the mean rating a has given items that belong to g in the training data.

The model construction phase of Model M3 involves computing $|U|$ average ratings for $|G|$ categories, each requires at most $|I|$ operations. Note that $|G| << |I| < |U|$ in reality. Model M3 has a computational complexity of $O(|U| \times |I|)$ for model construction. Its rating prediction phase depends on the number of categories to which t belongs, and has a computational complexity of $O(|G_t|)$. The largest value of $|G_t|$ is 9 in our dataset.

Review-based models

We designed three review-based models (M4, M5 and M6) for generating personalized predictions. These models are mainly adapted from Model M2, the item-based CF model, but make use of reviews in different ways. They involve an additional sentiment analysis step, which has a linear computational complexity as discussed in Chapter 6.4.1. They build user, item and categories profiles based on item features extracted from reviews. The set of features appeared in reviews can be potentially large. Its size, however, has a practical limit and is controllable by means of feature weighting and filtering as aforementioned.

Model M4 is similar to Model M2 which performs item-based CF, but it computes $w(t, i)$ between items t and i based on review contents rather than on ratings. Note that in a pure CF setting, similarity between two items t and i, or $w(t, i)$, can be derived only if they have been rated by the same user. In Model M4, however, $w(t, i)$ can be derived if the two items share common item features. This model addresses the problems of data sparseness and non-transitive association [70], because common features mentioned in the reviews of different items help linking items that may be related or similar to each other, but have never been rated by the same user in the system. This model is expected to improve the coverage rate of Model M2, or the classical item-based CF algorithm.

The model construction phase of Model M4 mainly scales with $|I|$, and has a computational complexity of $O(|I|^2)$ for computing item similarities. The rating prediction phase of the model has a computational complexity of $O(k)$.

Model M5 can be viewed as a generalized version of Model M4. It takes G into consideration as an effort to address the problems of data sparseness and cold-start recommendations. We expect that this model can boost coverage rate because it is capable of generating predictions for cold-start items whose categories are known, but have not been rated by any user yet.

Note that the process of k-nn selection in this model is slightly different from that in the previous models. An item can belong to multiple categories, and each category has its own set of k-nn. It is therefore possible for a certain category g_k to be the nearest neighbor of more than one $g \in G_t$. In this case, we sum up the weights of g_k as it might be more closely related to item t. Algorithm 6.4.1 gives an overview of the $knnG()$ algorithm for determining the k-nn of G_t.

Algorithm 6.4.1 *Algorithm $knnG()$ for determining the k-nn of a set of categories G_t.*

Inputs: k (the number of nearest neighbors to return),
G_t (the set of categories to which item t belongs).

Output: k-nn across the set of categories in G_t

Steps:

1. KNN $\leftarrow \emptyset$;
2. **for** each g in G_t **do**
3. **for** each $g_k \in knn(g)$ **do**
4. **if** KNN contains g_k **then**
5. add $w(g_k, g)$ to $w(g_k)$ in KNN;
6. **else**
7. $w(g_k) \leftarrow w(g_k, g)$;
//weights of g_k with respect to different g's are combined
8. add $(g_k, w(g_k))$ to KNN;
9. **end if**
10. **end for**
11. **end for**
12. **return** the k elements in KNN with the highest weights;

After computing the similarities between categories from reviews, Model M5 predicts $p_{a,t}$ as:

$$p_{a,t} = \alpha \sum_{g \in knnG(G_t)} w(g)(r_{a,g}) \tag{6.5}$$

where $w(g)$ is the weight of g computed across the set of all neighbors of G_t (lines 5, 7 of Algorithm 6.4.1).

The model construction phase of Model M5 requires computing the similarities between categories based on a controllable number of item features. It also needs to computes users' average ratings for G based on I. Its computational complexity is therefore $O(|U| \times |I|)$. Model M5 determines the set of $knnG(G_t)$ in the rating prediction phase. It has a computational complexity of $O(G_t \times k)$, but k is small (5 in our work), and the maximum value of $|G_t|$ is also small (at

most 9 as aforementioned).

Model M6 makes use of opinion phrases and item features extracted from reviews to build user and category profiles. It makes a prediction for item t based on the active user a's preferences for features of G_t. We consider features of G_t rather than those of target item t in view of severe data sparseness and the considerable number of cold-start items in reality. This model has a more direct utilization of review contents, as compared to the other two review-based models (M4 and M5) which do not make use of detailed review contents for building user profiles. Further, this model is not neighborhood-based.

Note that if a user has showed preference for a feature f_n of item i, it does not necessarily imply that (s)he would have the preference for the same feature of another item j. However, if a user tend to express positive sentiments on certain feature terms, it may imply that the user is interested in those features in general. We try to make use of such coarse-grained preference information in this model. In fact, our evaluation results show that this model is the most promising one among all prediction models.

This model can also make predictions for cold-start items, as long as the items belong to some categories that possess features on which the active user has previously expressed opinions. It does not involve the similarity weighting between pairs of users, items or categories as in Models M2 to M5. It computes $p_{a,t}$ by reflecting how much user a has liked the features associated with G_t. Formally, it computes $p_{a,t}$ as:

$$p_{a,t} = \frac{1}{|G_t|} \sum_{g \in G_t} \left(\frac{1}{|N|} \sum_{n \in N} w(f_n, a) \cdot w(f_n, g) \right) \tag{6.6}$$

where N denotes the set of common features associated with g and the interest profile of user u. $w(f_n, a)$ is the weight of f_n in the interest profile of a, determined based on overall ratings or SO of opinion phrases as described in Chapter 6.4.2. Similarly, $w(f_n, g)$ is the weight of f_n with respect to the category g. It can be estimated by standard feature weighting schemes, such as information gain and χ^2. It reflects the importance of f_n to g.

This model maintains both user profiles and category profiles based on features extracted from reviews. Its model construction phase mainly depends on the sentiment analysis step, which as a linear computational complexity as noted. Model M6 is therefore more scalable and efficient than the other quadratic

prediction models, including Models M1, M2 and M4. Its rating prediction phase has a complexity of $O(|F|)$. The actual complexity is again likely to be much smaller because of data sparseness, and the fact that the value of $|F|$ is limited and controllable due to feature selection.

Baseline models

Baseline models, including Models M7 and M8, generate non-personalized predictions, with or without using information about the $U \times I$ ratings matrix. They are included in our study to help indicate the effectiveness of the various prediction models.

Model M7 is the Majority baseline, which simply predicts $p_{a,t}$ to be the majority vote in the training data. Almost half (47.93%) of the reviews in our dataset had the majority vote of 5. We therefore expected this simple baseline model to produce reasonable prediction accuracy.

Model M7 determines the majority rating in the dataset with a computational complexity of $O(|U|\times|I|)$. It makes a prediction with a constant time complexity of $O(1)$.

Model M8 is a Random recommender which, as its name implies, assigns a random rating within the given rating scale (1 to 5 in our dataset) as $p_{a,t}$. Models that cannot beat the this Random model would not be desirable.

Model M8 does not require model construction, and makes a prediction with a constant time complexity of $O(1)$.

6.5 Experimental Study

The main goal of this experimental study is to compare the prediction quality of rating-based prediction models, which make use of user-specified ratings for making predictions as in classical CF, and that of review-based models, which make predictions based on user-generated reviews.

Most models are adapted from item-based CF, which was proposed in view of the scalability issue of user-based CF. Hence, the actual computational requirements of the various models are not the focus of our study. We aim at systematically evaluating the usefulness of prediction models that make use of user-generated reviews for making predictions.

We now describe the experimental setup and then discuss the results.

6.5.1 Method

The dataset we used contains attraction reviews we collected from TripAdvisor as noted. We employed the all-but-1 protocol in this study. Specifically, for each active user a in our dataset, all except 1 reviews written by a are used as the training set, based on which predictions for the hidden reviews in the test set are made. Note that our task at hand is to make prediction for a given user-item pair (a, t). The contents of the hidden review on t are not available to the prediction models. This experimental design emulates the situation when a registered user (a) of, for instance, TripAdvisor, clicks on a hyperlink to web page of a certain attraction (t) (s)he has not reviewed or rated before, and our task is to compute a predicted rating $p_{a,t}$ to indicate how much user a may like item t.

We adopted MAE, MSE and coverage for evaluating the performance of the various models. Coverage in this work is defined as the percentage of withheld ratings in the test set for which predictions can be made. All reported results are averaged results obtained from five randomly created training-test splits. In the subsequent discussions, when we state that results are significant, we mean so statistically based on the Wilcoxon signed-rank test, which is a non-parametric alternative to the paired t-test, at a 5% significance level.

6.5.2 Existing Tools Adopted

We adopted two existing tools for performing some tasks in our work. We used MontyLingua [92] to assign POS tags to reviews. It yields tagging accuracy of around 97%. Moreover, we adopted RapidMiner[12], formerly known as YALE, to compute feature weights based on IG and χ^2. As we discussed in Chapter 2.4.3, $IG(f_i, c_j)$ and $\chi^2(f_i, c_j)$ of a feature f_i are computed locally with respect to a given class c_j. RapidMiner estimates $IG(f_i)$ (and similarly, $\chi^2(f_i)$) as a global measure by first computing $IG_{sum}(f_i)$. It then normalizes all features having non-zero weights by the maximum $IG_{sum}(f_i)$ obtained by the feature set. We followed the same strategy when computing the TF-IDF of a feature f_i as a global measure.

[12]RapidMiner: http://www.rapidminer.com

6.5.3 Parameters

Four prediction models compute $p_{a,t}$ based on the set of k-nn of the active user (Model M1) or the target item (Models M2, M4, M5). Herlocker et al. suggested in [54] that using a neighborhood size of 20 to 50 would be reasonable for a real-world application. We performed a set of preliminary experiments based on Models M1 and M5, and varied the value of k within the suggested range. We observed no significant difference between the results obtained, and we therefore set the value of k to 50 in the four aforementioned models.

Model M6 predicts $p_{a,t}$ based on the k-nn of categories G. Our dataset contains a total number of 148 categories (excluding the category "Other"). This number is much smaller than the numbers of users (2,074) and items (2,741) in the dataset. We therefore used a smaller neighborhood for Model M6. We varied the value of k within the range of 3 to 10 in our preliminary experiments. Once again, the resulting prediction accuracies of the model are not sensitive to the changing values of k. We picked the value of 5 for k in our experiments on Model 6.

6.5.4 Results and Discussions

In what follows, we first present results on the effects of using SO, followed by a set of experiments that help us select an appropriate feature weighting scheme and feature selection (FS) level for the subsequent experiments. We then discuss and compare the prediction quality produced by the various prediction models described in the previous section.

Effects of using SO

This experiment studies the effects of using SO for building user profiles. Recall that one motivation of our work is that user-generated reviews contain detailed user preferences that are seemingly useful, but have not yet been utilized for personalization purpose. Our first experiment therefore aims at validating whether such user preferences, which are first extracted as opinion phrases and then quantified as SO, are indeed useful.

We used Model M6 described in Eq. (6.6) in this experiment, because it makes predictions directly from the active user's preferences for item features. We built two sets of user profiles as $w(f_n, a)$, one based on SO of opinion

phrases, another one based on user-specified ratings associated with the original reviews. We used the three feature weighting schemes described before, namely IG, χ^2 and TF-IDF, as $w(f_n, g)$.

Figure 6.8 summarizes the results of this experiment obtained using the various feature weighting schemes, at FS levels of 100% and 1%. Note that χ^2 is displayed as "Chi" in all figures presenting results. Detailed FS results are given in the next subsection. Labels on the x-axes marked with asterisks (*) indicate that the differences between the results generated using SO and those generated using user-specified ratings are statistically significant ($p < 0.05$). As a complement to Figure 6.8, Table 6.2 lists the percentage of improvement in MAE and MSE produced using SO, as compared to those produced using user-specified ratings. The first row of the table, for example, shows that using IG as the feature weighting scheme and a FS level of 100%, the MAE produced using SO for building user profiles is 8.2% lower than that produced using user-specified ratings.

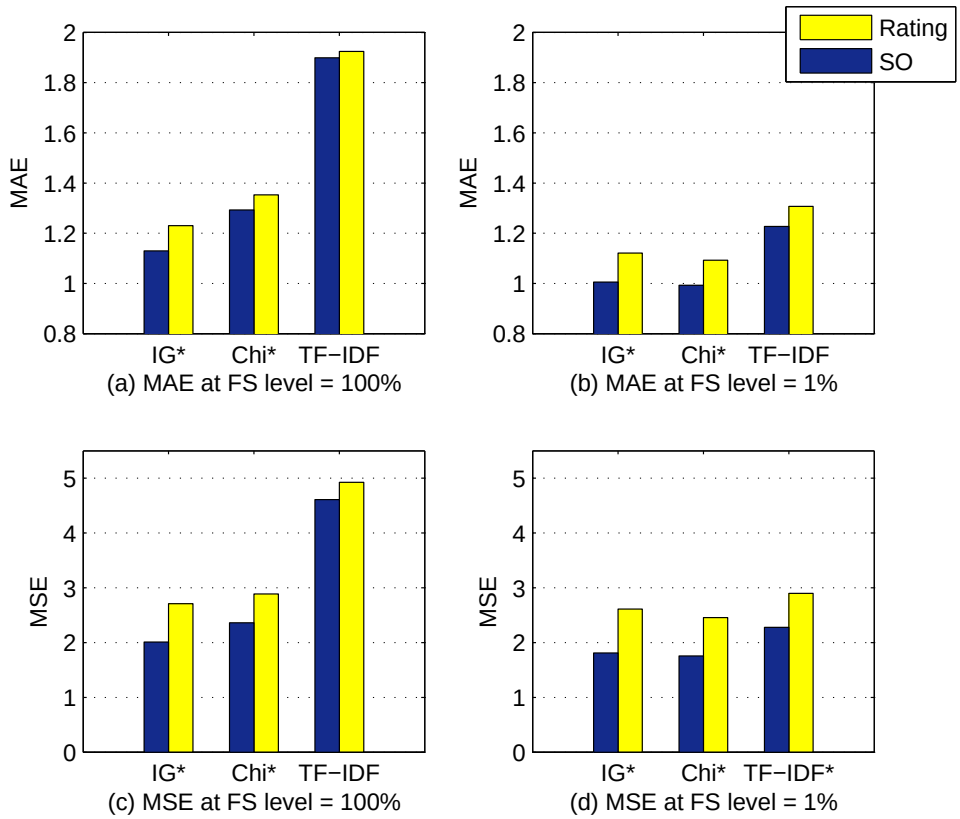

Figure 6.8: Summary of results achieved using SO and user-specified ratings.

Table 6.2: Relative improvements in MAE and MSE achieved using SO.

		Improvement in (%)	
Feature Weighting	FS Level	MAE	MSE
IG	100%	8.2	25.8
Chi (χ^2)		4.4	18.1
TF-IDF		1.3	6.4
IG	1%	10.3	30.7
Chi (χ^2)		9.1	28.6
TF-IDF		6.1	21.4

The absolute MAE and MSE values produced under different experimental settings are not of interest here, but rather the comparison between using SO and using ratings for building user profiles. Results show that using SO always makes better predictions than using user-specified ratings. The improvements it brings are significant in most cases as can be seen from Figure 6.8. Further, using SO yields great improvements in prediction accuracies in terms of MSE, especially when only the top 1% important features are used to make predictions as Table 6.2 shows.

To sum up, results of this experiment suggest that the detailed user preferences expressed in reviews effectively improve prediction accuracies and help reduce large prediction errors. These results are very encouraging, and confirm the value of our work on studying review-based recommendations. In the subsequent experiments, we report our results generated using SO for building user profiles in Model M6.

Effects of feature weighting and selection

This experiment evaluates the effects of the three feature weighting schemes, namely IG, χ^2 and TF-IDF. The purpose of this experiment is to help us determine an appropriate scheme and FS level for the subsequent experiments.

We first calculated feature weights based on the aforementioned feature weighting schemes. In each experiment, $s\%$ of the most important features (those with the highest weights) are used for building item profiles in Model M6. The set of values of s tested is $\{100, 90, ..., 20, 10, 5, 2, 1\}$. The setting s = 100 means that features were weighted, but no feature selection was actually

performed. Figure 6.9 shows the results.

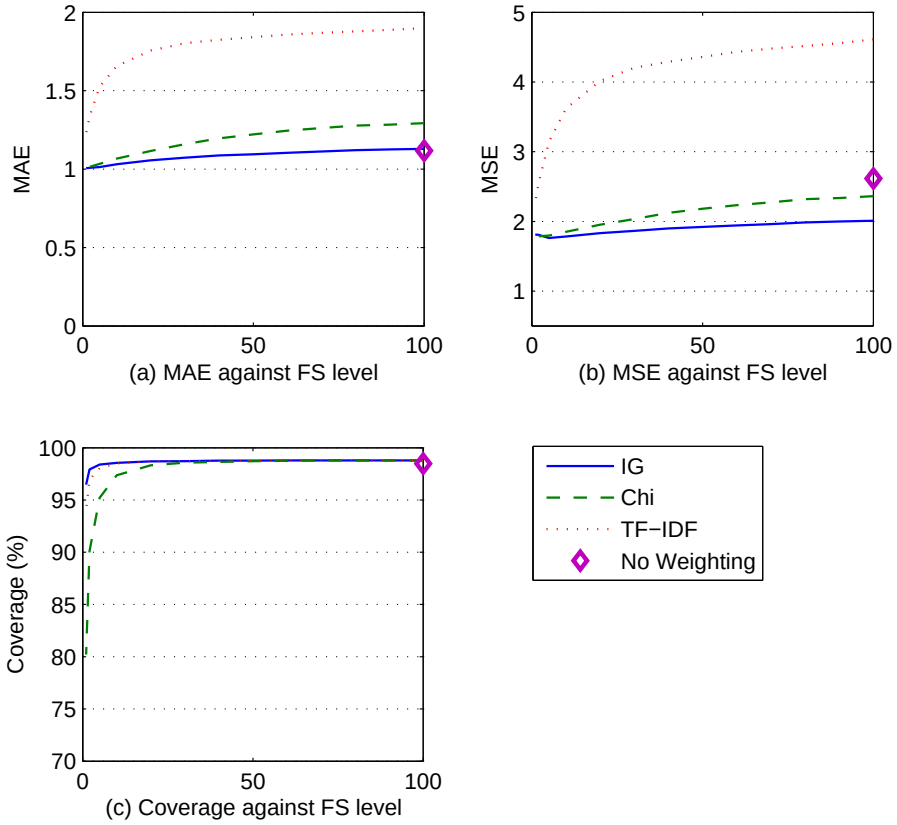

Figure 6.9: Summary of results achieved using various feature weighting schemes and feature selection levels.

We first observed from the results that prediction accuracies produced by the various feature weighting schemes decline gradually as more features are considered. This trend is more obvious when s is relatively small ($\leq$ 20). This suggests that using only more important features produces better prediction accuracies.

The three feature weighting schemes generate varying prediction accuracies. Overall speaking, IG is consistently the best performing feature weighting scheme at all FS levels, followed by χ^2, and then TF-IDF. At more rigorous FS levels (when $s \leq 10$), however, the MAE and MSE values produced using IG and χ^2 are statistically indistinguishable. At s = 1, for instance, χ^2 outperforms IG very slightly in terms of MAE (0.993 vs. 1.005, $p \leq 0.310$) and MSE (1.756 vs. 1.811, $p \leq 0.056$).

Using TF-IDF for weighting features yields poor prediction accuracies, even if only the 1% most important features are retained. It produces a lot of large errors as reflected by its MSE values, which at least doubled those produced by IG at $s \geq 10$.

All the three weighting schemes yield coverage rates of over 98% when $s \geq 20$, with IG always performs the best. χ^2 is more sensitive than the other two schemes to the changing values of s. Its coverage rates decline rapidly when s is small. For example, when s = 5, 2 and 1, the coverage rates of χ^2 are respectively 95.2%, 90.1% and 80.2%, whereas those of IG are respectively 98.4%, 97.9% and 96.5%.

To sum up, using only the more important features as item descriptors makes better predictions. IG yields the best results in terms of MAE and MSE in general, but χ^2 produces comparable results when $s \leq 5$. IG also performs the best in terms of coverage. Note that there is not a single feature weighting scheme and FS level setting that can give an optimal performance. On the one hand, the lowest absolute values of MAE and MSE are achieved by χ^2 at n = 1. However, those achieved by IG are statistically indistinguishable from the best when $n \leq 5$. On the other hand, IG always beats χ^2 in terms of coverage significantly at $n \leq 10$. Taking all MAE, MSE and coverage into account, we adopted IG for weighting features with a FS level of 1% in the subsequent experiments.

Performance analysis of prediction models

We now analyze the performance achieved using the various prediction models, summarized in Figure 6.10. In the figure, labels on the x-axis of each sub-plot are model names. Models that produced the best results, or results statistically indistinguishable from the best, are marked with asterisks (*). We now discuss our findings from this set of experiments.

The best performing models: We first observed from the first sub-plot of Figure 6.10 that Model M7, the majority baseline, gives the lowest absolute MAE value of 0.982. This should not be too surprising because almost half of the reviews in our dataset were associated with the majority rating, which is 5 as aforementioned.

Model M6, which makes use of detailed SO extracted from reviews for making predictions, produces a MAE of 1.005. This value is statistically

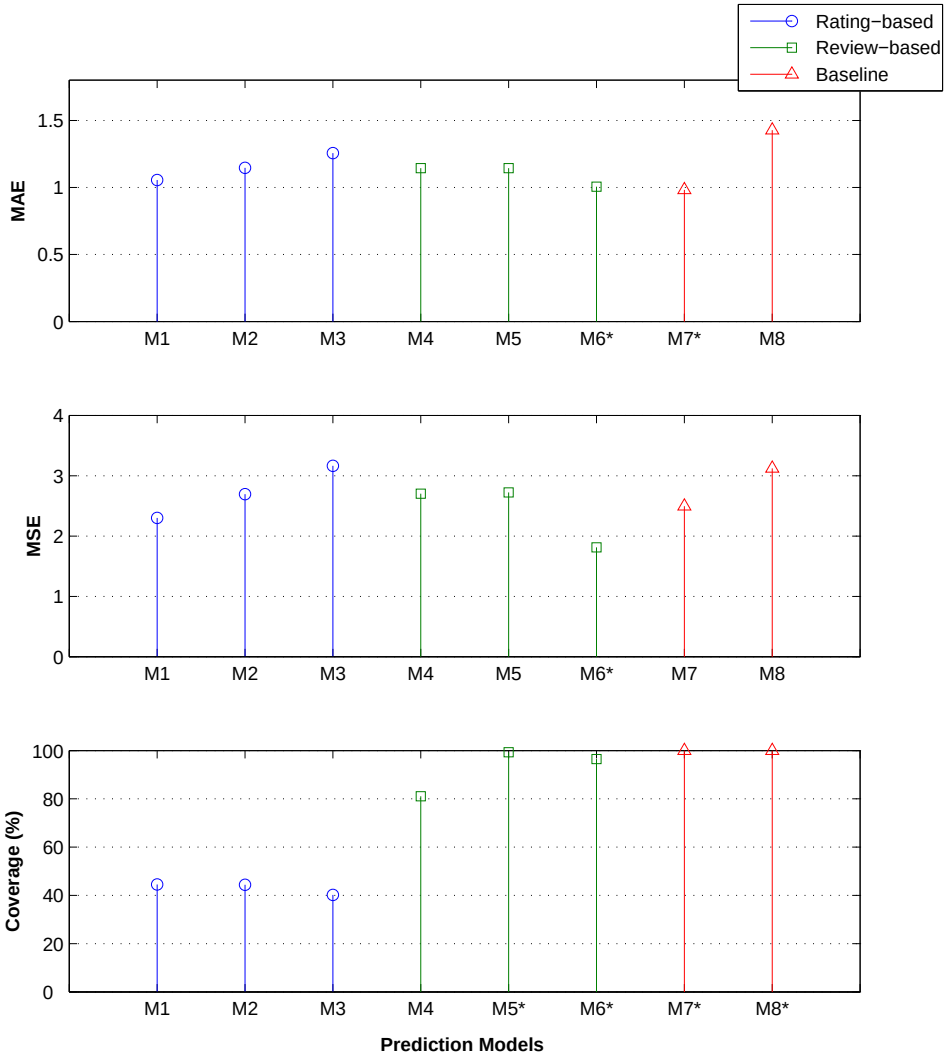

Figure 6.10: Summary of prediction quality and coverage of all prediction models.

indistinguishable from the best ($p \leq 0.015$). Model M6 also performs the best in terms of MSE (1.811), and gives a slightly-less-than-perfect coverage rate of 96.46%. This coverage rate is obtained at a FS level of 1% as noted. Model M6 is able to provide a 98.79% coverage when no feature selection is performed.

Note that the perfect coverage rates given by the two baseline models M7 and M8 are actually not meaningful. They are only non-personalized models that can generate "predictions" for any items and any users.

The worst performing models: Model M8 gives the highest MAE (3.123), which is not surprising. Model M3 gives a MAE of 1.256, a MSE of 3.164, and a coverage rate of 40.2%. It performs worst among all personalized models based on all three metrics. This indicates that mean ratings on categories alone are too general for making predictions, and are not sufficient for addressing the problem of data sparseness. For instance, if user a has not rated any items in G_t, this model will not be able to make a prediction for t.

Rating-based models: Model M1, which is the user-based CF model, always outperforms the other two rating-based models in terms of MAE and MSE. It is unable to beat the non-personalized Model M7 in terms of MAE, but its MSE value of 2.3 is significantly lower than the MSE of Model M7. This means that Model M1 produces fewer large errors than the majority baseline. Note that all three rating-based model suffers severely from data sparseness, as can be seen from the poor coverage rates they achieved. This emphasizes the need to utilize additional preference data along with the user-specified ratings used in existing CF-based systems for making personalized recommendations.

Review-based models: Model M6, which builds user profiles based on users' SO for item features, and derives similarities between categories based on item features as well, always gives the best performance statistically. The other two review-based models, Models M4 and M5, produced mixed performance. Both models show inferior prediction accuracies as compared to the majority baseline (M7), but Model M5 yields an excellent coverage rate of 99.32% as a result of utilizing category similarities for making predictions.

The excellent coverage rates achieved by the three review-based models indicate that we effectively address the problems of data sparseness and cold-start recommendations by enriching user, item and category profiles using review contents.

Comparisons between related prediction models: We compared two

groups of related prediction models, summarized as follows. The first group of models we compared consists of Models M2, M4 and M5. Recall that Model M2 is the item-based CF model. Model M4 is similar to Model M2, but uses item features extracted from reviews for computing similarities between items. Model M5 is a generalized version of M4. Instead of computing similarities between items, it considers similarities between the categories of items. Such similarities are also derived from item features extracted from reviews. Figure 6.10 shows that the three models produce indistinguishable MAE and MSE, but Models M4 and M5 yield much better coverage rates than Model M2. In other words, these two review-based models can greatly increase the number of recommendable items in the system with *no loss of prediction accuracy*.

The second group of models we compared consists of Models M5 and M6. Both models derive similarities between categories based on reviews, but M6 predicts a rating with respect to the active users' SO on features. We observed from our first experiment (Chapter 6.5.4) that using SO of opinion phrases for building user profiles yields better prediction accuracies than using user-specified ratings in Model M6. The advantage of using SO is even more obvious, especially in terms of MSE, when we compare the performance of Models M5 and M6 in this experiment. Once again, we conclude that our experimental results are very encouraging.

6.6 Summary

This chapter presents a closer look at the use of user-generated reviews for generating personalized recommendations. In view of the success of our work on rating inference in the movie domain, and the fact that CF-based recommendations are not common in travel-related domains, we based this study on an attraction reviews dataset collected from TripAdvisor. We identified three characteristics of user-generated travel reviews from the dataset. The first and the most noticeable characteristic is severe data sparseness, which is also a crucial challenge to traditional rating-based CF. This implies that a practical review-based recommendation algorithm must address data sparseness, and should be able to generate cold-start recommendations. The second characteristic is the heterogeneity of domain items. Specifically, "tourist attractions" is not a single item domain, as different types of attractions can be totally different in nature and

possess very different properties. The third characteristic is that the feature set used to describe domain items in user-generated reviews shall be ever evolving and expanding as a large volume of reviews (or contents in general) are added to the Web everyday.

To facilitate our experimental study on review-based recommender systems, we applied sentiment analysis techniques for identifying item features and opinion phrases from the user-generated reviews, as well as to determine the SO of the opinion phrases. We explored the use of the SO of opinion phrases for building interest profiles of users, and the use of item features extracted from reviews for deriving similarities between items and those between item categories. We then designed a variety of prediction models that predict how much an active user would like a given item (attraction).

Our study is experimental in nature, and is the first study reporting empirical results on the use of review contents for generating personalized recommendations. Due to the lack of precedent literature, we adopted in our work the classical user- and item-based CF models, based on which we developed several variants of the item-based CF models for generating review-based recommendations. Despite the simplicity of the models, results suggest that user-generated reviews do contain valuable user preferences that can be utilized for making personalized recommendations. Further, deriving similarities between items and those between item categories based on item features extracted from reviews effectively addresses data sparseness and the cold-start problem. Specifically, classical user- and item-based CF models operating on the original $U \times I$ ratings matrix can only make predictions for approximately 40% of items in the test set. Enriching user, item and category profiles by user-generating reviews, however, produces almost perfect coverage rates.

One limitation of our work is related to the sentiment analysis of user-generated reviews. We adopted a POS-based approach to feature term and opinion phrase extraction because such approach is domain-independent, and it has been adopted in various studies in sentiment analysis. We have not explicitly evaluated the precision of such approach in this study. However, we point out that our study is still fair and is able to achieve its goals because we use the same set of feature terms and opinion phrases for training the various review-based prediction models.

Another limitation of our experimental study is that it is based on a static

dataset, meaning that we have not yet addressed the *evolving feature set* characteristic of user-generated reviews. In the review-based prediction models, feature weights with respect to items and categories need to be recalculated when new reviews are added to the system. A simple method to address this issue to recalculate feature weights, as well as retrain user, item and category profiles regularly to capture the effects of the newly-added features. However, training and retraining based on huge and high-dimensional datasets can be computationally expensive. While research on algorithms that allow for incremental training is beyond the scope of this thesis, such algorithms would be desirable for performing review-based recommendations.

Chapter 7

Conclusions

We addressed in this thesis the crucial challenges in CF, in particular the problems of data sparseness and cold-start recommendations, along two different dimensions. Firstly, we proposed novel methods for integrating content information about domain items into the CF process. Secondly, we investigated into the use of user-generated reviews for generating personalized recommendations. We summarize our contributions, and reflect on possible future research directions in the following subsections.

7.1 Summary of Contributions

On hybrid CF- and content-based recommendations

We proposed two ARM-based recommendation techniques, namely FARAMS and CLARE. ARM allows for the flexibility to integrate concept hierarchies into the rule mining process. By taking advantage of this, and by utilizing taxonomies as well as attributes of domain items, we addressed the problems of data sparseness and cold-start recommendations in CF. We showed experimentally that our proposed techniques outperform related techniques in terms of recommendation accuracies.

Chapter 3 describes FARAMS, a CF recommendation framework based on fuzzy association rule mining and multiple-level similarities between items in item taxonomies. FARAMS extends existing studies on ARM-based CF algorithms. It addresses the sharp boundary problem, which arises from the boolean discretization of numerical ratings data, by modeling ratings data using

the fuzzy set concept. It also approaches the problem of data sparseness in CF by taking advantage of multiple-level similarities that are implicit in the taxonomies of items. Results on FARAMS show that FAR mining is more effective on datasets containing continuous ratings and that FARAMS outperforms existing techniques in similar experimental settings.

Chapter 4 presents CLARE, a novel cold-start recommendation algorithm developed based on FARAMS. CLARE operates on a preference model comprising both user-item (ratings data) and item-item (item attributes) relationships. It applies the CAR mining technique to discover interesting associations between domain items and the attributes posses by a given target item. CLARE is capable of recommending cold-start items, which are not recommendable in a pure CF setting. It shows superior recommendation accuracies to related cold-start recommendation algorithms, including MS-based and pure content-based algorithms. Furthermore, it achieves high coverage rates regardless of the number of cold-start items in the system.

On utilizing user-generated reviews for personalized recommendations

Our work in this area bridges the gap between sentiment analysis and CF. On the one hand, CF can be considered an application that utilizes the outputs of sentiment analysis for personalization purpose. On the other hand, sentiment analysis enables CF algorithms to make use of user-generated reviews as a source of user preferences along with user-specified ratings. Our work marks a starting point for review-based personalized recommendations from either perspective by integrating sentiment analysis and CF.

Chapter 5 presents our initial effort on integrating sentiment analysis of CF. CF is concerned with user preferences, therefore we are particularly interested in users' expressed opinions in reviews. We observed that the *semantic similarity between opinion words does not necessarily imply their sentimental similarity*. This, however, has been the underlying assumption of a class of SO determination techniques, such as those described in [61, 62, 72, 68]. We further observed that the relative frequencies of opinion words across different sentiment classes might be useful indicators of the opinion words' SO and strength. Based on these observations, we proposed a relative-frequency-based method for determining the SO and strength of opinion words as part of our rating inference framework, PREF. PREF aims at understanding the overall sentiments

of reviews and mapping such sentiments onto a multi-point rating scale. We empirically demonstrated the rating inference approach to the integration of sentiment analysis and CF by using the predicted ratings generated by PREF for performing CF. We observed the following encouraging results: the predicted ratings produce reasonably good prediction accuracies, and PREF improves the performance of CF significantly by augmenting ratings from reviews.

Chapter 6 investigates further into the use of user-generated reviews for generating personalized recommendations. In CF, user profiles are constructed from users' ratings on items, while in hybrid recommendations algorithms, user and item profiles may be built upon ratings as well as item attributes. Our work described in Chapter 6 attempts to construct user profiles based on their preferences for item features extracted from reviews. It also makes use of item features to derive similarities between items and those between item categories. We performed an experimental evaluation of eight prediction models, two of them being non-personalized baseline models. The evaluation has two major goals. Firstly, it aims to show that sentiments extracted from reviews are more precise descriptions of user preferences than user-specified ratings. Secondly, it compares the effectiveness of prediction models that generate predictions based on user-generated reviews, and those that perform rating-based predictions. Results suggest that user-generated reviews do contain valuable user preferences that can be utilized for making personalized recommendations. Further, review-based models improve the coverage of rating-based models, with comparable or significantly better prediction accuracies. This indicates that deriving similarities between items and those between item categories based on feature terms extracted from reviews effectively addresses data sparseness and the cold-start problem.

7.2 Suggestions for Future Research

Research on CF has been focusing on more scalable and accurate algorithms, but our work in this thesis represents an important pioneering step towards the development of a novel review-based CF paradigm. Future research on CF shall continue to explore the use of user-generated reviews, or other types of Web 2.0 contents, for CF. Further, we pointed out in Chapter 6.3.3 that user-generated reviews are characterized by a high-dimensional and evolving feature

set. Algorithms that can handle this by, for instance, allowing incremental training of user, item and category profiles would be desirable for supporting reviews-based recommendations.

Regarding research on sentiment analysis, we see an emerging trend in the paradigm shift from binary sentiment classification to *multi-point rating inference*. Recent studies, including ours, suggest that rating inference shall not be tackled as a classical multi-category classification task because the *continuity and ordering of class labels* in the rating inference task are essential [112, 114, 41]. This opens up an interesting direction in ordered multi-category classification for future research.

Another interesting step to consider is the to *recommend reviews for users to read* according to their interest profiles. Several user studies suggest that user-generated reviews do play an important role in the online purchasing behavior of users [131, 78, 46]. Specifically, when a user makes a decision about an online purchase, (s)he tends to consult reviews written by other users who have purchased or examined the product concerned. Instead of recommending to users what to buy, it might also be useful to recommend what reviews to read to facilitate the users' decision making processes. The work of Wietsma and Ricci [167] incorporates reviews written by the neighbors of the active user as a decision making aid, but it does not involve the sentiment analysis of reviews in the recommendation process. This actually reveals a gap between the state-of-the-art in review-based recommendations and sentiment analysis.

Bibliography

[1] AAMODT, A., AND PLAZA, E. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications 7*, 1 (1994), 39–59.

[2] ACIAR, S., ZHANG, D., SIMOFF, S., AND DEBENHAM, J. Recommender system based on consumer product reviews. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence* (2006), pp. 719–723.

[3] ACIAR, S., ZHANG, D., SIMOFF, S., AND DEBENHAM, J. Informed recommender: Basing recommendations on consumer product reviews. *IEEE Intelligent Systems 22*, 3 (2007), 39–47.

[4] ADOMAVICIUS, G., AND TUZHILIN, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge Engineering 17*, 6 (2005), 734–749.

[5] AGRAWAL, R., IMIELINSKI, T., AND SWAMI, A. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (1993), pp. 207–216.

[6] AGRAWAL, R., AND SRIKANT, R. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Databases* (1994), pp. 487–499.

[7] ANDERSON, M., BALL, M., BOLEY, H., GREENE, S., HOWSE, N., LEMIRE, D., AND MCGRATH, S. RACOFI: A rule-applying collaborative filtering system. In *Workshop on Collaboration Agents:*

Autonomous Agents for Collaborative Environments, in conjunction with the IEEE/WIC International Conference on Web Intelligence/Intelligent Agent Technology (2003).

[8] BAEZA-YATES, R., AND RIBEIRO-NETO, B. *Modern Information Retrieval*. Addison Wesley, 1999.

[9] BALABANOVIC, M., AND SHOHAM, Y. Fab: Content-based, collaborative recommendation. *Communications of the ACM 40*, 3 (1997), 66–72.

[10] BASU, C., HIRSH, H., AND COHEN, W. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the 15th National Conference on Artificial Intelligence* (1998), pp. 714–720.

[11] BELKIN, N. J., AND CROFT, W. B. Information filtering and information retrieval: two sides of the same coin? *Communications of the ACM 35*, 12 (December 1992), 29–38.

[12] BIKEL, D. M., SCHWARTZ, R., AND WEISCHEDEL, R. M. An algorithm that learns what's in a name. *Machine Learning 34*, 1-3 (1999), 211–231.

[13] BILLSUS, D., AND PAZZANI, M. Learning collaborative information filters. In *Proceedings of the 15th International Conference on Machine Learning* (1998), pp. 46–54.

[14] BREESE, J. S., HECKERMAN, D., AND KADIE, C. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence* (1998), pp. 43–52.

[15] BRILL, E. A simple rule-based part-of-speech tagger. In *Proceedings of the 3rd Conference on Applied Natural Language Processing* (1992), pp. 152–155.

[16] BRIN, S., MOTWANI, R., ULLMAN, J. D., AND TSUR, S. Dynamic itemset counting and implication rules for market basket data. In *Proceedings ACM SIGMOD International Conference on Management of Data* (1997), pp. 255–264.

[17] BRUCE, R., AND WIEBE, J. Recognizing subjectivity: A case study of manual tagging. *Natural Language Engineering 5*, 2 (1999), 187–205.

[18] BURKE, R. Knowledge-based recommender systems. *Encyclopedia of Library and Information Systems 69*, 32 (2000).

[19] BURKE, R. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction 12* (2002), 331–370.

[20] CHESLEY, P., VINCENT, B., XU, L., AND SRIHARI, R. Using verbs and adjectives to automatically classify blog sentiment. In *Proceedings of AAAI-CAAW-06, the Spring Symposia on Computational Approaches to Analyzing Weblogs* (2006).

[21] CHO, Y. H., AND KIM, J. K. Application of Web usage mining and product taxonomy to collaborative recommendations in e-commerce.

[22] CIMIANO, P., AND VOLKER, J. Text2onto: A framework for ontology learning and data-driven change discovery. In *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems* (2005), pp. 227–238.

[23] CLAYPOOL, M., GOKHALE, A., MIRANDA, T., MURNIKOV, P., NETES, D., AND SARTIN, M. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR Workshop on Recommender Systems* (1999).

[24] DAS, S., AND CHEN, M. Yahoo! for Amazon: Extracting market sentiment from stock message boards. In *Proceedings of the Asia Pacific Finance Association Annual Conference* (2001).

[25] DAVE, K., LAWRENCE, S., AND PENNOCK, D. M. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th International World Wide Web Conference* (2003), pp. 519–528.

[26] DIEDERICH, J., KINDERMANN, J., LEOPOLD, E., AND PAA, G. Authorship attribution with support vector machines. *Applied Intelligence 19*, 1/2 (2003), 109–123.

[27] E., S., KOKKINAKIS, G., AND FAKOTAKIS, N. Automatic text categorization in terms of genre and author. *Computational Linguistics 26*, 4 (2000), 471–495.

[28] ESULI, A., AND SEBASTIANI, F. Determining the semantic orientation of terms through gloss classification. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)* (2005), pp. 617–624.

[29] ESULI, A., AND SEBASTIANI, F. SentiWordNnet: A publicly available lexical resource for opinion mining. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)* (2006).

[30] FAN, W., WALLACE, L., RICH, S., AND ZHANG, Z. Tapping into the power of text mining. *Communications of ACM 49*, 9 (2006), 76–82.

[31] FAWCETT, T. An introduction to ROC analysis. *Pattern Recognition Letters 27*, 8 (2006), 861–874.

[32] FELDMAN, R., AND DAGAN, I. Knowledge discovery in textual databases (KDT). In *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining* (Montreal, Canada, 1995), pp. 112–117.

[33] FESENMAIER, D. R., AND RICCI, F. Dietorecs: Travel advisory for multiple decision styles. In *International Conference on Information and Communication Technologies in Tourism (ENTER)* (2003), pp. 232–241.

[34] FLACH, P. A. ICML'04 Tutorial on the many faces of ROC analysis in machine learning, 2004.

[35] FOLTZ, P. W., AND DUMAIS, S. T. Personalized information delivery: An analysis of information filtering methods. *Communications of the ACM 35*, 12 (December 1992), 51–60.

[36] FU, A. W. C., WONG, M. H., SZE, S. C., WONG, W. C., WONG, W. L., AND YU, W. K. Finding fuzzy sets for the mining of fuzzy association rules for numerical attributes. In *Proceedings of the First*

International Symposium on Intelligent Data Engineering and Learning (1998), pp. 263–268.

[37] GABRILOVICH, E., AND MARKOVITCH, S. Text categorization with many redundant features: Using aggressive feature selection to make SVMs competitive with C4.5. In *Proceedings of the 21st International Conference on Machine Learning* (2004), pp. 321–328.

[38] GAMON, M., AUE, A., CORSTON-OLIVER, S., AND RINGGER, E. K. Pulse: Mining customer opinions from free text. In *Proceedings of the 6th International Symposium on Intelligent Data Analysis* (2005), pp. 121–132.

[39] GAO, L., CHANG, E., AND SONG, H. Powerful tool to expand business intelligence: Text mining. In *Proceedings of World Academy of Science, Engineering and Techology (Volume 8)* (October 2005), pp. 110–115.

[40] GENG, L., AND HAMILTON, H. J. Interestingness measures for data mining: A survey. *ACM Computing Surveys 38*, 3 (2006). Article No. 9.

[41] GOLDBERG, A. B., AND ZHU, X. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *Proceedings of the HLT-NAACL Workshop on TextGraphs: Graph-based Algorithms for Natural Language Processing* (2006), pp. 45–52.

[42] GOLDBERG, D., NICHOLS, D., OKI, B., AND TERRY, D. Using collaborative filtering to weave an information tapestry. *Communications of the ACM 35*, 12 (1992), 61–70.

[43] GOLDBERG, K. Y., ROEDER, T., GUPTA, D., AND PERKINS, C. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval 4*, 2 (2001), 133–151.

[44] GOOD, N., SCHAFER, J. B., KONSTAN, J., BORCHERS, A., SARWAR, B., HERLOCKER, J., AND RIEDL, J. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of Conference of the American Association of Artifical Intelligence* (1999), pp. 439–446.

[45] GRAHAM, P. Better spam filtering. In *Proceedings of the Spam Conference* (2003). Retrieved November 19, 2005, from http://paulgraham.com/better.html.

[46] GRETZEL, U., AND YOO, K. H. Use and impact of online travel reviews. In *International Conference on Information and Communication Technologies in Tourism (ENTER)* (2008), pp. 35–46.

[47] GRISHMAN, R. Information extraction: techniques and challenges. In *Information Extraction (International Summer School SCIE-97)* (1997).

[48] GYENESEI, A. A fuzzy approach for mining quantitative association rules. TUCS Technical Report 336, Turku Centre for Computer Science, 2000.

[49] GYENESEI, A. Interestingness measures for fuzzy association rules. In *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery* (2001), pp. 152–164.

[50] HAN, J., AND FU, Y. Mining multiple-level association rules in large databases. *IEEE Transactions on Knowledge and Data Engineering 11*, 5 (1999), 798–804.

[51] HAN, J., AND KAMBER, M. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2000.

[52] HATZIVASSILOGLOU, V., AND MCKEOWN, K. R. Predicting the semantic orientation of adjectives. In *Proceedings of the 8th Conference on European Chapter of the Association for Computational Linguistics* (1997), pp. 174–181.

[53] HEARST, M. A. Untangling text data mining. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics* (1999), pp. 3–10. See also: http://mappa.mundi.net/trip-m/hearst/.

[54] HERLOCKER, J., KONSTAN, J., AND RIEDL, J. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information Retrieval 5* (2002), 287–310.

[55] HERLOCKER, J. L., KONSTAN, J. A., TERVEEN, L. G., AND RIEDL, J. T. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS) 22*, 1 (2004), 5–53.

[56] HIPP, J., GUNTZER, U., AND NAKHAEIZADEH, G. Algorithms for association rule mining – a general survey and comparison. *ACM SIGKDD Explorations Newsletter 2*, 1 (2000), 58–64.

[57] HOFMANN, T. Probabilistic latent semantic indexing. In *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval* (1999), pp. 50–57.

[58] HOFMANN, T., AND PUZICHA, J. Latent class models for collaborative filtering. In *Proceedings of the International Joint Conference in Artificial Intelligence* (1999), pp. 688–693.

[59] HONG, T.-P., AND LEE, C.-Y. Learning fuzzy knowledge from training examples. In *Proceedings of the 7th ACM Conference on Information and Knowledge Management* (1998), pp. 161–166.

[60] HSU, W.-L., AND LANG, S.-D. Classification algorithms for NETNEWS articles. In *Proceedings of the 8th International Conference on Information and Knowledge Management* (Kansas City, MO, USA, 1999), pp. 114–121.

[61] HU, M., AND LIU, B. Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2004), pp. 168–177.

[62] HU, M., AND LIU, B. Mining opinion features in customer reviews. In *Proceedings of the 19th National Conference on Artificial Intelligence* (2004), pp. 755–760.

[63] HUANG, Z., ZENG, D., AND CHEN, H. A comparative study of recommendation algorithms in e-commerce applications. *IEEE Intelligent Systems 22*, 5 (2007), 68–78.

[64] JACKSON, P., AND MOULINIER, I. *Natural Language Processing for Online Applications: Text Retrieval, Extract and Categorization*. John Benjamins Publishing Company.

[65] JINDAL, N., AND LIU, B. Identifying comparative sentences in text documents. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (2006), pp. 244–251.

[66] JOACHIMS, T. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods - Support Vector Learning* (1999), B. Scholkopf, C. Burges, and A. Smola, Eds., MIT Press, pp. 41–56.

[67] KAJI, N., AND KITSUREGAWA, M. Building lexicon for sentiment analysis from massive collection of HTML documents. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (2007), pp. 1075–1083.

[68] KAMPS, J., MARX, M., MOKKEN, R., AND DE RIJKE, M. Using WordNet to measure semantic orientations of adjectives. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC)* (2004), pp. 1115–1118.

[69] KIM, B. M., LI, Q., KIM, J. W., AND KIM, J. A new collaborative recommender system addressing three problems. In *Proceedings of the 8th Pacific Rim International Conference on Artificial Intelligence* (2004), pp. 495–504.

[70] KIM, B. M., LI, Q., PARK, C. S., KIM, S. G., AND KIM, J. Y. A new approach for combining content-based and collaborative filters. *Journal of Intelligent Information Systems 27*, 1 (2006), 79–91.

[71] KIM, C., AND KIM, J. A recommendation algorithm using multi-level association rules. In *Proceedings of the IEEE/WIC International Conference on Web Intelligence* (2003), pp. 524–527.

[72] KIM, S.-M., AND HOVY, E. Determining the sentiment of opinions. In *Proceedings of Conference on Computational Linguistics* (2004), pp. 1367–1373.

[73] KLEMETTINEN, M., MANNILA, H., RONKAINEN, R., H., T., AND VERKAMO, A. I. Finding interesting rules from large sets of discovered association rules. In *Proceedings of the 3rd International Conference on Information and Knowledge Management* (1994), pp. 401–407.

[74] KOHAVI, R., BECKER, B., AND SOMMERFIELD, D. Improving simple Bayes. In *Proceedings of European Conference on Machine Learning* (1997), pp. 78–87.

[75] KONSTAN, J. A., MILLER, B. N., MALTZ, D., HERLOCKER, J. L., GORDON, L. R., AND RIEDL, J. GroupLens: Applying collaborative filtering to usenet news. *Communications of the ACM 40*, 3 (1997), 77–87.

[76] LAM, X. N., VU, T., LE, T. D., AND DUONG, A. D. Addressing cold-start problem in recommendation systems. In *Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication* (2008), pp. 208–211.

[77] LAMONT, J. Business intelligence: The text analysis strategy. KMWorld, October 2006. Retrieved April 18, 2008, from: http://www.kmworld.com/Articles/Editorial/Feature/Business-Intelligence-The-text-analysis-strategy-18526.aspx.

[78] LEINO, J., AND RAIHA, K.-J. Case amazon: ratings and reviews as part of recommendations. In *Proceedings of the 2007 ACM Conference on Recommender Systems* (2007), pp. 137–140.

[79] LEUNG, C. W. K., AND CHAN, S. C. F. Sentiment analysis of product reviews. *J. Wang (Eds), Encyclopedia of Data Warehousing and Mining - Second Edition, Information Science Reference* (2008), 1794–1799.

[80] LEUNG, C. W. K., CHAN, S. C. F., AND CHUNG, F. L. A collaborative filtering framework based on fuzzy association rules and multiple-level similarity. *Knowledge and Information Systems 10*, 3 (2006), 357–381.

[81] LEUNG, C. W. K., CHAN, S. C. F., AND CHUNG, F. L. Integrating collaborative filtering and sentiment analysis: A rating inference approach.

In *Proceedings of The ECAI 2006 Workshop on Recommender Systems* (2006), pp. 62–66.

[82] LEUNG, C. W. K., CHAN, S. C. F., AND CHUNG, F. L. Applying cross-level association rule mining to cold-start recommendations. In *Proceedings of the IEEE/WIC/ACM WI-IAT Workshop on Web Personalization and Recommender Systems* (2007), pp. 133–136.

[83] LEUNG, C. W. K., CHAN, S. C. F., AND CHUNG, F. L. Evaluation of a rating inference approach to utilizing textual reviews for collaborative recommendation. In *Cooperative Internet Computing* (2008), World Scientific Publisher.

[84] LEUNG, C. W. K., CHAN, S. C. F., AND CHUNG, F. L. A probabilistic rating inference framework for mining user preferences from reviews. *submitted to World Wide Web - Internet and Web Information Systems (WWWJ)* (under revision).

[85] LI, J., AND CERCONE, N. Discovering and ranking important rules. In *Proceedings of IEEE International Conference on Granular Computing* (2005), pp. 506–511.

[86] LI, J., AND CERCONE, N. A rough set based model to rank the importance of association rules. In *Proceedings of International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing* (2005), pp. 109–118.

[87] LI, J., PATTARAINTAKORN, P., AND CERCONE, N. Rule evaluations, attributes, and rough sets: Extension and a case study. *Transactions on Rough Sets (Lecture Notes in Computer Science) 6* (2007), 152–171.

[88] LI, J., TANG, B., AND CERCONE, N. Applying association rules for interesting recommendations using rule templates. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)* (2004), pp. 166–170.

[89] LIN, W., ALVAREZ, S. A., AND RUIZ, C. Efficient adaptive-support association rule mining for recommender systems. *Data Mining and Knowledge Discovery 6*, 1 (2002), 83–105.

[90] LINDEN, G., SMITH, B., AND YORK, J. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing 7*, 1 (2003), 76–80.

[91] LIU, B., HU, M., AND CHENG, J. Opinion Observer: Analyzing and comparing opinions on the web. In *Proceedings of the 14th International Conference on World Wide Web* (2005), pp. 342–351.

[92] LIU, H. MontyLingua: An end-to-end natural language processor with common sense, 2004. Available online at http://web.media.mit.edu/%7Ehugo/montylingua.

[93] LIU, J., YAO, J., AND WU, G. Sentiment classification using information extraction technique. In *Advances in Intelligent Data Analysis VI* (2005), pp. 216–227.

[94] LORENZI, F., AND RICCI, F. Case-based recommender systems: a unifying view. In *IJCAI-05 Workshop on Intelligent Techniques for Web Personalization* (2005), B. Mobasher and S. Anand, Eds., pp. 89–113.

[95] LORENZI, F., AND RICCI, F. Case-based recommender systems: a unifying view. In *Intelligent Techniques for Web Personalization* (2005), B. Mobasher and S. Anand, Eds., pp. 89–113.

[96] MAEDCHE, A., AND STAAB, S. Ontology learning for the semantic web. *IEEE Intelligent Systems 16*, 2 (2001), 72–79.

[97] MALTZ, D., AND EHRLICH, K. Pointing the way: Active collaborative filtering. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems* (1995), pp. 202–209.

[98] MANDREOLI, F., MARTOGLIA, R., AND TIBERIO, P. Text clustering as a mining task. In *Text Mining and Its Application to Intelligence, CRM and Knowledge Management*, A. Zanasi, Ed. Southampton, UK: WIT Press, 2005, pp. 75–108.

[99] MANNING, C. D., RAGHAVAN, P., AND SCHUTZE, H. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[100] MCCALLUM, A., AND NIGAM, K. A comparison of event models for Naive Bayes text classification. In *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization* (1998).

[101] MCJONES, P. EachMovie collaborative filtering data set, 1997. retired as of October, 2004.

[102] MCNEE, S. M. *Meeting User Information Needs in Recommender Systems*. PhD thesis, University of Minnesota, 2006.

[103] MELVILLE, P., MOONEY, R. J., AND NAGARAJAN, R. Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the 18th National Conference on Artificial Intelligence* (2002), pp. 187–192.

[104] MIDDLETON, S. E., ALANI, H., AND DE ROURE, D. C. Exploiting synergy between ontologies and recommender systems. In *Proceedings of the WWW2002 International Workshop on the Semantic Web* (2002).

[105] MIDDLETON, S. E., R., S. N., AND DE ROURE, D. C. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems 22*, 1 (2004), 54–88.

[106] MILLER, B. N. *Toward a Personal Recommender System*. PhD thesis, University of Minnesota, 2003.

[107] MILLER, G., BECKWITH, R., FELLBAUM, C., GROSS, D., AND MILLER, K. Introduction to WordNet: An online lexical database. *International Journal of Lexicography (Special Issue) 3*, 4 (1990), 235–312.

[108] MISHNE, G., AND GLANCE, N. Predicting movie sales from blogger sentiment. In *Proceedings of AAAI-CAAW-06, the Spring Symposia on Computational Approaches to Analyzing Weblogs* (2006).

[109] MOBASHER, B., JIN, X., AND ZHOU, Y. Semantically enhanced collaborative filtering on the web. In *Proceedings of the First European Web Mining Forum* (2003), pp. 57–76.

[110] MOONEY, R., AND ROY, L. Content-based book recommending using learning for text categorization. In *Proceedings of the 5th ACM Conference on Digital Libraries* (2000), pp. 195–204.

[111] N., Z. C., LAUSEN, G., AND SCHMIDT-THIEME, L. Taxonomy-driven computation of product recommendations. In *Proceedings of the 2004 ACM CIKM International Conference on Information and Knowledge Management* (2004), pp. 406–415.

[112] OKANOHARA, D., AND TSUJII, J. Assigning polarity scores to reviews using machine learning techniques. In *Proceedings of the Second International Joint Conference on Natural Language Processing* (2005), pp. 314–325.

[113] PANG, B., AND LEE, L. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computation Linguistics* (2004), pp. 271–278.

[114] PANG, B., AND LEE, L. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computation Linguistics* (2005), pp. 115–124.

[115] PANG, B., LEE, L., AND VAITHYANATHAN, S. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (2002), pp. 79–86.

[116] PARK, J. S., CHEN, M.-S., AND YU, P. S. An effective hash-based algorithm for mining association rules. In *Proceedings of ACM SIGMOD International Conference on Management of Data* (1995), pp. 175–186.

[117] PARK, S. T., PENNOCK, D., MADANI, O., GOOD, N., AND DECOSTE, D. Naive filterbots for robust cold-start recommendations. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2006), pp. 699–705.

[118] PAWLAK, Z. *Rough Sets: Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, 1992.

[119] PEI, J., HAN, J., AND MAO, R. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *Proceedings of ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery* (2000), pp. 21–30.

[120] POPESCU, A., AND ETZIONI, O. Extracting product features and opinions from reviews. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing* (2005), pp. 339–346.

[121] POPESCUL, A., UNGAR, L. H., PENNOCK, D. M., AND LAWRENCE, S. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence* (2001), pp. 437–444.

[122] PORTER, M. F. An algorithm for suffix stripping. *Program 14*, 3 (1980), 130–137.

[123] PROVOST, F., AND FAWCETT, T. Robust classification for imprecise environments. *Machine Learning 42*, 3 (2001), 203–231.

[124] RADEV, D., FAN, W., AND ZHANG, Z. WebInEssence: A personalized web-based multi-document summarization and recommendation system. In *Proceedings of the NAACL Workshop on Automatic Summarization* (Pittsburgh, PA, June 2001).

[125] RASHID, A., ALBERT, I., COSLEY, D., LAM, S., MCNEE, S., KONSTAN, J., AND RIEDL, J. Getting to know you: Learning new user preferences in recommender systems. In *Proceedings of the International Conference on Intelligent User Interfaces* (2002), pp. 127–134.

[126] RENZ, I., AND FRANKE, J. Text mining. In *Text Mining: Theoretical Aspects and Applications*. 2003.

[127] RESNICK, P., IACOVOU, N., SUCHAK, M., BERGSTORM, P., AND RIEDL, J. GroupLens: An open architecture for collaborative filtering

of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work* (1994), ACM, pp. 175–186.

[128] RICCI, F. Travel recommender systems. *IEEE Intelligent Systems 17*, 6 (2002), 55–57.

[129] RICCI, F., CAVADA, D., MIRZADEH, N., AND VENTURINI, A. *Case-Based Travel Recommendations*. Travel Destination Recommendation Systems: Behavioral Foundations and Applications. CAB Publishing, 2006, ch. 6.

[130] RICCI, F., AND DEL MISSIER, F. Supporting travel decision making through personalized recommendation. In *Designing Personalized User Experiences for eCommerce* (2004), C.-M. Karat, J. Blom, and J. Karat, Eds., Kluwer Academic Publisher, pp. 221–251.

[131] RICCI, F., AND WIETSMA, R. T. A. Product reviews in travel decision making. In *Proceedings of the International Conference on Information and Communication Technologies in Tourism (ENTER)* (2006), pp. 296–307.

[132] RIFKIN, R., AND KLAUTAU, A. In defense of one-vs-all classification. *Journal of Machine Learning Research 5* (2004), 101–141.

[133] RILOFF, E., AND WIEBE, J. Learning extraction patterns for subjective expressions. In *Proceedings of Conference on Empirical Methods in Natural Language Processing* (2003), pp. 105–112.

[134] ROBB, D. Text mining tools take on unstructured data. Computer World, 2004. Retrieved November 24, 2005, from: http://www.computerworld.com/printthis/2004/0,4814,93968,00.html.

[135] SARWAR, B., KARYPIS, G., J., K., AND RIEDL, J. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM conference on Electronic Commerce* (2000), pp. 158–167.

[136] SARWAR, B., KARYPIS, G., KONSTAN, J., AND RIEDL, J. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International World Wide Web Conference* (2001), pp. 285–295.

[137] SARWAR, B., KONSTAN, J., BORCHERS, A., HERLOCKER, J., MILLER, B., AND RIEDL, J. Using filtering agents to improve prediction quality in the GroupLens research collaborative filtering system. In *Proceedings of the ACM conference on Computer Supported Cooperative Work* (1998), pp. 345–354.

[138] SARWAR, B. M., KARYPIS, G., KONSTAN, J. A., AND RIEDL, J. T. Application of dimensionality reduction in recommender systems a case study. In *Proceedings of the ACM WebKDD Workshop, in conjunction with the ACM-SIGKDD Conference on Knowledge Discovery in Databases* (2000).

[139] SCHAFER, J. B., KONSTAN, J. A., AND RIEDL, J. E-commerce recommendation applications. *Data Mining and Knowledge Discovery 5*, 1–2 (2001), 115–153.

[140] SCHEIN, A., POPESCUL, A., UNGAR, L. H., AND PENNOCK, D. M. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (2002), pp. 253–260.

[141] SCHEIN, A. I., POPESCUL, A., UNGAR, L. H., AND PENNOCK, D. M. CROC: A new evaluation criterion for recommender systems. *M. E. Zurko and A. Greenwald (Eds.), Electronic Commerce Research, Special Issue on World Wide Web Electronic Commerce, Security and Privacy 5*, 1 (2005), 51–74.

[142] SEBASTIANI, F. Machine learning in automated text categorization. *ACM Computing Surveys 34*, 1 (2002), 1–47.

[143] SEBASTIANI, F. *Text Categorization*. WIT Press, Southampton, UK, 2005, ch. Text Mining and Its Application to Intelligence, CRM and Knowledge Management. pages 109–129.

[144] SHARDANAND, U., AND MAES, P. Social information filtering: Algorithms for automating 'Word of Mouth'. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems* (1995), pp. 210–217.

[145] SHIMADA, K., AND ENDO, T. Seeing several stars: A rating inference task for a document containing several evaluation criteria. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)* (2008), pp. 1006–1014.

[146] SILVESTRI, F., BARAGLIA, R., AND PALMERINI, P. Online generation of suggestions for web users. *Journal of Digital Information Management 2*, 2 (2004), 104–108.

[147] SMOLA, A. J., AND SCHOLKOPF, B. A tutorial on support vector regression. Tech. Rep. NC2-TR-1998-030, NeuroCOLT2, Royal Holloway College, University of London, October 1998.

[148] SMYTH, B., AND COTTER, P. A personalized television listings service. *Communications of the ACM 43*, 8 (2000), 107–111.

[149] SNYDER, B., AND BARZILAY, R. Multiple aspect ranking using the good grief algorithm. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics* (2007), pp. 300–307.

[150] SRIKANT, R., AND AGRAWAL, R. Mining generalized association rules. In *Proceedings of the 21st International Conference on Very Large Databases* (1995), pp. 407–419.

[151] TABOADA, M., ANTHONY, C., AND VOLL, K. Methods for creating semantic orientation dictionaries. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)* (2006), pp. 427–432.

[152] TAN, P.-N., KUMAR, V., AND SRIVASTAVA, J. Selecting the right interestingness measure for association patterns. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2002), pp. 32–41.

[153] TERVEEN, L., HILL, W., AMENTO, B., MCDONALD, D., AND CRETER, J. PHOAKS: A system for sharing recommendations. *Communications of the ACM 40*, 3 (1997), 59–62.

[154] TRIPADVISOR. Fact sheet. Retrieved April 11, 2009, from http://www.tripadvisor.com/ PressCenter-c4-Fact%5FSheet.html, 2007.

[155] TSO-SUTTER, K. H. L., MARINHO, L. B., AND SCHMIDT-THIEME, L. Tag-aware recommender systems by fusion of collaborative filtering algorithm. In *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC)* (2008), pp. 1995–1999.

[156] TURNEY, P. D. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (2002), pp. 417–424.

[157] TURNEY, P. D., AND LITTMAN, M. L. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems 21*, 4 (2003), 315–346.

[158] UNGAR, L. H., AND FOSTER, D. P. Clustering methods for collaborative filtering. In *Proceedings of the AAAI Workshop on Recommendation Systems* (1998).

[159] VAPNIK, V. N. *The Nature of Statistical Learning Theory*. Springer, 1995.

[160] VELARDI, P., NAVIGLI, R., AND MISSIKOFF, M. Integrated approach for web ontology learning and engineering. *IEEE Computer 35*, 11 (2002), 60–63.

[161] WANG, K., HE, Y., AND HAN, J. Mining frequent itemsets using support constraints. In *Proceedings of the 26th International Conference on Very Large Databases* (2000), pp. 43–52.

[162] WANG, K., AND LIU, H. Q. Mining is-part-of association patterns from semistructured data. In *Proceedings of the 9th IFIP 2.6 Working Conference on Database Semantics* (2000).

[163] WATSON, I. *Applying Case-Based Reasoning: Techniques for Enterprise Systems*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA, 1997.

[164] WEISS, S. M., INDURKHYA, N., ZHANG, T., AND DAMERAU, F. *Text Mining: Predictive Methods for Analyzing Unstructured Information*. Springer, New York, USA, 2004.

[165] WIEBE, J., BRUCE, R., BELL, M., MARTIN, M., AND WILSON, T. A corpus study of evaluative and speculative language. In *Proceedings of the 2nd ACL SIGdial Workshop on Discourse and Dialogue* (2001).

[166] WIEBE, J., BRUCE, R., AND O'HARA, T. Development and use of a gold-standard data set for subjectivity classifications. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics* (1999), pp. 246–253.

[167] WIETSMA, R. T. A., AND RICCI, F. Product reviews in mobile decision aid systems. In *Proceedings of the Conference on Information and Communication Technologies in Tourism (ENTER)* (2005), pp. 15–18.

[168] WILSON, T., WIEBE, J., AND HWA, R. Just how mad are you? Finding strong and weak opinion clauses. In *Proceedings of the 19th National Conference on Artificial Intelligence* (2004), pp. 761–769.

[169] WOLVERTON, T. Despite tough year, amazon's bezos keeps his chin up. CNET News.com, November 2000. Retrieved August 18, 2008, from: http://news.cnet.com/2009-1017%5F3-249204-2.html.

[170] YAMANISHI, K., AND LI, H. Mining open answers in questionnaire data. *IEEE Intelligent Systems 17*, 5 (2002), 58–63.

[171] YANG, Y., AND PEDERSEN, J. O. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning (ICML-97)* (1997), D. H. Fisher, Ed., Morgan Kaufmann Publishers, pp. 412–420.

[172] YI, J., NASUKAWA, T., BUNESCU, R., AND NIBLACK, W. Sentiment Analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *Proceedings of the 3rd IEEE International Conference on Data Mining* (2003), pp. 427–434.

[173] ZADEH, L. A. Knowledge representation in fuzzy logic. *IEEE Transactions on Knowledge and Data Engineering 1*, 1 (March 1989), 89–100.

[174] ZAKI, M. J. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering 12*, 3 (2000), 372–390.

[175] ZAKI, M. J., PARTHASARATHY, S., OGIHARA, M., AND LI, W. New algorithms for fast discovery of association rules. Technical Report 651, Computer Science Department, The University of Rochester, 1997.

[176] ZHOU, L., AND CHAOVALIT, P. Ontology-supported polarity mining. *Journal of the American Society for Information Science and Technology 59*, 1 (2008), 98–110.

[177] ZHU, X., AND GOLDBERG, A. B. Kernel regression with order preferences. In *Proceedings of the 22nd National Conference on Artificial Intelligence* (2007), pp. 681–686.

[178] ZIEGLER, C., MCNEE, S. M., KONSTAN, J. A., AND LAUSEN, G. Improving recommendation lists through topic diversification. In *Proceedings of the Fourteenth International World Wide Web Conference* (2005), pp. 22–32.