Relative Preference-based Recommender Systems

By

Shaowu Liu

BIT (Hons)

Submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

Deakin University

June, 2016

DEAKIN UNIVERSITY
ACCESS TO THESIS - A

I am the author of the thesis entitled

Relative Preference-based Recommender Systems

submitted for the degree of

Doctor of Philosophy

This thesis may be made available for consultation, loan and limited copying in accordance with the Copyright Act 1968.

'I certify that I am the student named below and that the information provided in the form is correct'

Full Name: .....Shaowu Liu.....
(Please Print)

Signed: ..... Signature Redacted by Library **.....**

Date: .....24/06/2016.....

DEAKIN UNIVERSITY
CANDIDATE DECLARATION

I certify the following about the thesis entitled (10 word maximum)

_____Relative Preference-based Recommender Systems_____

_____

submitted for the degree of _____DOCTOR OF PHILOSOPHY_____

a. I am the creator of all or part of the whole work(s) (including content and layout) and that where reference is made to the work of others, due acknowledgment is given.

b. The work(s) are not in any way a violation or infringement of any copyright, trademark, patent, or other rights whatsoever of any person.

c. That if the work(s) have been commissioned, sponsored or supported by any organisation, I have fulfilled all of the obligations required by such contract or agreement.

d. That any material in the thesis which has been accepted for a degree or diploma by any university or institution is identified in the text.

e. All research integrity requirements have been complied with.

'I certify that I am the student named below and that the information provided in the form is correct'

Full Name: .....Shaowu Liu.....
(Please Print)

Signed: ..... Signature Redacted by Library **.....**

Date:
.....24/06/2016.....

To my unborn baby...

Table of Contents

Table of Contents v

List of Tables viii

List of Figures ix

Acknowledgements x

Publication List xi

Abstract xiii

1 Introduction 1
- 1.1 Background and Motivation . . . 1
- 1.2 Research Objectives . . . 4
- 1.3 Overview of the Proposed Methodology . . . 5
- 1.4 Thesis Outline . . . 7

2 Literature Review 9
- 2.1 Notation and Problem Formulation . . . 9
- 2.2 Recommendation Techniques . . . 12
  - 2.2.1 Content-based Recommender Systems . . . 12
  - 2.2.2 Collaborative Filtering . . . 14
  - 2.2.3 Context-Aware Recommendation . . . 16
  - 2.2.4 Graph-based Recommendation . . . 17
  - 2.2.5 Trust-based Recommendation . . . 18
- 2.3 Relative Preference-based Recommender Systems . . . 18
  - 2.3.1 Notation and Problem Statement . . . 20
  - 2.3.2 Memory-based Models . . . 24

2.3.3 Model-based Models . . . 25
2.4 Evaluation Metrics . . . 28
2.4.1 Accuracy Metrics . . . 29
2.4.2 Diversity . . . 30
2.4.3 Coverage . . . 30
2.4.4 Stability . . . 31
2.4.5 Metrics for Relative Preference-based Models . . . 31
2.5 Summary . . . 32

3 Ordinal Random Fields for Recommender Systems 34
3.1 Introduction . . . 34
3.2 Preliminaries . . . 37
3.2.1 Matrix Factorization . . . 39
3.2.2 Ordinal Matrix Factorization . . . 40
3.2.3 Summary . . . 41
3.3 Ordinal Random Fields . . . 42
3.3.1 Markov Random Fields . . . 43
3.3.2 ORF: Unifying MRF and OMF . . . 45
3.3.3 Feature Design . . . 46
3.3.4 Parameter Estimation . . . 47
3.3.5 Preference Prediction . . . 48
3.4 Experiment and Analysis . . . 49
3.4.1 Experimental Settings . . . 49
3.4.2 Result and Analysis . . . 52
3.5 Summary . . . 57

4 Preferecen Relation-based Recommender System 58
4.1 Introduction . . . 58
4.2 Preliminaries . . . 61
4.2.1 Preference Relation . . . 62
4.2.2 Problem Statement . . . 63
4.2.3 Related Work . . . 64
4.3 Methodology . . . 66
4.3.1 Preference Relation-based Matrix Factorization . . . 67
4.3.2 Markov Random Fields . . . 70
4.3.3 Conditional Random Fields . . . 73
4.3.4 PrefCRF: Unifying PrefNMF and CRF . . . 76
4.4 Experiment and Analysis . . . 82
4.4.1 Results and Analysis . . . 85

4.5 Summary 96

5 Learning from Heterogeneous Data Sources for Improved Top-N Recommendation 97
5.1 Introduction 97
5.2 Preliminaries 99
5.2.1 Heterogeneous Sources 99
5.2.2 Preference Relation 100
5.2.3 Related Work 102
5.3 Preference Relation-based Conditional Random Fields 104
5.3.1 Problem Statement 104
5.3.2 Preference Relation-based Matrix Factorization 105
5.3.3 Conditional Random Fields 106
5.3.4 Ordinal Logistic Regression 109
5.3.5 PrefCRF: Unifying PrefNMF and CRF 110
5.4 Experiment and Analysis 115
5.4.1 Experimental Settings 115
5.4.2 Results and Analysis 118
5.5 Summary 125

6 Conclusion and Future Work 127
6.1 Contributions 127
6.2 Future Work 128

Bibliography 130

List of Tables

2.1 Notations used in rating-based RecSys 11
2.2 Capabilities of existing methods 19
2.3 Notations used in Relative Preference-based RecSys 23

3.1 Summary of Major Notations (Chapter 3) 38
3.2 Performance of ORF model 53
3.3 Paired t-test for the *ORF* and the *OMF*. 53

4.1 Summary of Major Notations (Chapter 4) 65
4.2 Capabilities of PR-based methods 67
4.3 Results on *MovieLens*-1M dataset 86
4.4 Results on *EachMovie* dataset 87
4.5 Paired t-test for *PrefMRF* and *PrefNMF*. 93

5.1 Performance of PrefCRF on *MovieLens*-1M dataset 120
5.2 Results over ten runs on *Amazon* dataset. 121
5.3 Results over ten runs on *EachMovie* dataset. 122
5.4 NDCG@10 on *MovieLens*-20M dataset. 123

List of Figures

2.1 An overview of Recommender System 13

3.1 Example of undirected graphs 44
3.2 Impact of Minimum Correlation Threshold 54
3.3 Impact of Minimum Correlation Threshold 55
3.4 Impact of Regularization Coefficient 56

4.1 Example of MRF graphs 72
4.2 Example of CRF graphs 74
4.3 Performance of different position T on MovieLens-1M dataset (Sparse). 89
4.4 Performance of different position T on MovieLens-1M dataset (Dense). 90
4.5 Performance of different position T on EachMovie dataset (Sparse). . 91
4.6 Performance of different position T on EachMovie dataset (Dense). . 92
4.7 Impact of Sparsity Levels. 94
4.8 Impact of Parameters (MovieLens-1M) 95

5.1 Flow from user preferences to PR 101
5.2 Undirected graphs for users u and v 108
5.3 Varying sparsity on *MovieLens*-1M dataset. 121
5.4 Varying position T on *EachMovie* dataset. 123
5.5 Varying biases on *MovieLens*-20M dataset. 124
5.6 Varying parameters on *MovieLens*-1M. 125

Acknowledgements

I would like to thank Dr. Gang Li, my supervisor, for his many suggestions and constant support during this research.

I am also thankful to my associate supervisors Prof. Gleb Beliakov and A/Prof. Ping Xiong for useful suggestions and friendly encouragement. I would like to thank my research mates and friends: Dr. Jia Rong, Dr. Huy Quan Vu, Dr. Truyen Tran, Dr. Veelasha Moonsamy, Dr. Yongli Ren, Dr Tianqing Zhu, and so many more. I am also thankful to the support staff, in particular Ms. Lauren Browne, Ms. Judy Chow for their administrative help.

Of course, I am grateful to the patience and love from my parents, who gave me birth at the first place and raised me up. I am also thankful for my parents in law for their support throughout my research studies. Last but not the least, I would like express my sincere gratitude to my beautiful wife, Manchun Li, for her patience and love. Without her everlasting encouragement and understanding, this work would never have come into existence.

Melbourne, Australia Shaowu Liu
June 2016

Publication List

Refereed Journal Articles

1. **Shaowu Liu**, Gang Li, Truyen Tran, and Yuan Jiang. Preference Relation-based Markov Random Fields for Recommender Systems. **Machine Learning**. (**ERA 2010 ranking A***)

2. Veelasha Moonsamy, Jia Rong, **Shaowu Liu**, Mining Permission Patterns for Contrasting Clean and Malicious Android Applications. **Future Generation Computer Systems**, 2014, 36: 122–132. (**ERA 2010 ranking A**)

3. **Shaowu Liu**, Rob Law, Jia Rong, Gang Li, and John Hall. Analyzing Changes in Hotel Customers' Expectations by Trip Mode. International Journal of Hospitality Management, 2013, 34: 359–371. (**ERA 2010 ranking A**)

4. Gleb Beliakov, Gang Li, **Shaowu Liu**, Parallel bucket sorting on Graphics Processing Units based on convex optimization. Optimization, 2015, 64(4): 1033-1055. (**Impact Factor: 0.936**)

5. Huy Quan Vu, Gang Li, Nadezda S. Sukhorukova, Gleb Beliakov, **Shaowu Liu**, Carole Philippe, Helene Amiel, Adrien Ugon. K-complex Detection using a Hybrid-Synergic Machine Learning method. **IEEE Transactions on Systems, Man, and Cybernetics Part C: Application and Reviews (IEEE TSMCC)**, 2012, 42(6): 1478–1490. (**Impact Factor: 2.171**)

6. Huy Quan Vu, **Shaowu Liu**, Xinghua Yang, Zhi Li, Yongli Ren. Identifying Microphone from Noisy Recordings by Using Representative Instance One Class-Classification Approach. Journal of Networks (JNW), 2012, 7(6): 908-917. (**ERA 2010 ranking A**)

Refereed Conference Papers

1. **Shaowu Liu**, Gang Li, Truyen Tran, Yuan Jiang. Preference Relation-based Markov Random Fields. In Proceedings of the 7th Asian Conference on Machine Learning (**ACML 2015**), pp. 157-172, Journal of Machine Learning Research: workshop and conference proceedings, 2015.

2. **Shaowu Liu**, Truyen Tran, Gang Li, Yuan Jiang. Ordinal Random Fields for Recommender Systems. In Proceedings of the 6th Asian Conference on Machine Learning (**ACML 2014**), pp. 283-298, Journal of Machine Learning Research: workshop and conference proceedings, 2014.

3. Veelasha Moonsamy, Jia Rong, **Shaowu Liu**, Gang Li, Lynn Batten. Contrasting Permission Patterns between Clean and Malicious Android Applications. In Proceedings of the 9th International Conference on Security and Privacy in Communication Networks (**SecureComm 2013**).

4. Huy Quan Vu, **Shaowu Liu**, Zhi Li, Gang Li. Microphone Identification using One Class-Classification Approach. In Proceedings of the 2nd Workshop on Applications and Techniques in Information Security (**ATIS 2011**).

Book Chapter

1. Gleb Beliakov and **Shaowu Liu**. Parallel Monotone Spline Interpolation and Approximation on GPUs. Designing Scientific Applications on GPUs, CRC Press, Taylor and Francis Group, 2013, 295–310.

Abstract

Recommender systems aim to recommend users with some of their potentially interesting items by exploiting various information, especially the absolute ratings. Nevertheless, recent literature has suggested that rating-based systems are less reliable comparing to those based on relative preferences, i.e., "which one is better?" instead of "what do you think of this one?" However, a problem of these emerging relative preference-based models is that they consider either the second order interactions, such as similarities between users, or the higher order interactions, such as latent factors. This limitation reduces the performance of relative preference-based systems as the two types of interactions are complementary. On the other hand, due to the change of input format, existing relative preference-based systems do not consider side information such as user profiles and item content, which can be helpful to further improve the performance. Furthermore, the potential of relative preference-based systems to merge heterogeneous data sets was not identified in literature, which can help alleviate the cold-start problem of having limited information for new users or items.

In this thesis, we tackle these three issues. We propose a novel model to exploit the ordinal properties possessed by ratings, where both the second and higher order interactions are considered. In this model, ratings are no longer considered as numbers, but a sequence of ordinal labels. The proposed model used *Markov Random Fields* to combine two types of interactions.

Another type of relative preference is *Preference Relation* (PR), i.e., comparisons of items. For PR-based systems, we proposed a modified version of *Markov*

Random Fields which accepts PR instead of ordinal preferences, by converting PR into user-wise preferences, and then into ordinal distributions through ordinal logistic regression. This process produces the first PR-based recommender system that captures both types of interactions. For incorporating side information, we extended the *Markov Random Fields* to *Conditional Random Fields*, in which the users profiles and item content are considered by designing new features.

Despite of improving existing PR-based systems, we also identified a great potential of such systems to merge heterogeneous data sets. Specifically, data sets in different format, such as *5-star ratings*, *binary ratings*, *page views*, and *mouse clicks* can all be converted into PR format and used by PR-based systems. This observation makes it possible to alleviate the cold-start problem by generating a much denser data set, which could not be done for rating-based systems.

To evaluate the performance of proposed models, we conducted experiments on different public data sets against the state-of-the-art relative preference-based models measured by different metrics. The results presented in the experiment sections of each chapter show statistically significant improvement over existing models. The main contributions of this research are proposing the first relative preference-based models that can capture both types of interactions, and using PR-based models to alleviate the cold-start problem.

Keywords: Recommender Systems, Preference Relation, Collaborative Filtering, Relative Preference, Ordinal Preference

Chapter 1

Introduction

1.1 Background and Motivation

Recommender Systems (RecSys) aim to suggest items (*books*, *movies*, *tourism attractions*, etc.) that are potentially to be liked by the user. To identify the appropriate items, RecSys use various sources of information, such as historical ratings given by users [38] or content of items [5]. RecSys were originally designed for users with insufficient personal experience or with limited knowledge on the items. However, with the rapid expansion of Web 2.0 and e-commerce, overwhelming number of items are offered, and now every user can be benefited from RecSys [66].

Over the last decade there have been rapid advances in RecSys, from both academia and industry [7, 20, 37, 44, 49, 76]. One of the most important events in RecSys was the one million *Netflix Prize* [7] launched in 2006, which sought for RecSys that outperform *Netflix* company's own RecSys. The dataset released in this competition contains historical ratings on movies given by individuals. The *Netflix* dataset, together with other datasets released by companies such as *Amazon* and *Yahoo!* have become the popular benchmark datasets in this field. Due to the extensive use of

these datasets, which contain ratings, most RecSys to date are designed to exploit ratings [40, 41, 69].

However, user feedbacks are not always expressed in form of absolute ratings, and it is often expensive to collect such explicit feedbacks. Furthermore, studies [12, 42] have reported that absolute ratings may not be completely trustworthy. For example, the rating 4 out of 5 may in general indicate high quality, but it can mean just OK for critics. In fact, users' quantitative judgment can be affected by a number of irrelevant factors such as the *mood* when rating, and in psychology this is called misattribution of memory [71].

While users are not good at making consistent quantitative judgment, the *relative preferences* such as *ordinal preferences* [42,46,79,82] and *preference relations* (PR) [12, 18, 19] have been considered as more consistent form of feedbacks across like-minded users. For example, by measuring the relative order between items, the *PR* is usually less variant to irrelevant factors: a user in bad *mood* may give lower ratings to all items but the relative orderings between items remain the same. Being a more reliable type of user preferences, *PR* is also easier to collect comparing to ratings as it can be inferred from implicit feedbacks. For example, the *PR* between two items can be inferred by comparing their *ratings*, *page views*, *played counts*, *mouse clicks*, etc. This property is important as not all users are willing to rate their preferences, where collecting feedbacks implicitly delivers a more user-friendly recommender system. In addition, as the ultimate goal of RecSys, obtaining the ranking of items by itself is to obtain the relative preferences, a more natural input than absolute ratings [42, 81].

Despite of its potential, the newly emerged relative preference-based RecSys provides less features comparing to the well-established rating-based RecSys. Meanwhile,

relative preference-based RecSys provides an alternative view of user preferences, thus can be used to resolve issues of rating-based RecSys. Currently, relative preference-based RecSys still faces the following unresolved issues:

- *Different Structures in User Preferences*: Existing recommendation techniques can be largely divided into two forms: memory-based [65,69] and model-based [38]. Memory-based approaches focus on capturing the second-order interactions between similar users [65] or items [69]. This type of information is called *Local Structure* (LS) of user preferences. On the other hand, model-based approaches focus on discovering the weaker but higher-order interactions among all users and items. This type of information is called *Global Structure* (GS) of user preferences. Previous studies have suggested that these two types of structure are complementary since they address different aspects of the preferences [38, 46, 79]. However, there is yet no relative preference-based RecSys that can capture both *LS* and *GS*.

- *Side Information of User Preferences*: While existing recommendation techniques focus on exploiting user preferences, *side information* such as item content and user attributes [79] are also shown to be useful in improving recommendation quality. However, due to the change of input format, there is yet no relative preference-based RecSys can incorporate side information.

- *User Preferences from Heterogeneous Sources*: Last decade has seen a growing trend towards creating and managing more profiles in *Online Social Networks*. User are now providing feedbacks on different platforms in different formats, such as *5-star ratings*, *thumbs up/down*, as well as implicitly as *mouse clicks*.

These rich, but *heterogeneous*, user preferences provide an opportunity of alleviate the cold-start problem [72]. However, existing recommendation techniques usually assume the user preferences are in the same format, and therefore are unable to exploit these heterogeneous user preferences.

The first two issues have constrained the potentials of relative preference-based RecSys, while the third issue is faced by all existing recommendation techniques. This thesis aims to address these issues to make relative preference-based ReSys more effective and applicable.

1.2 Research Objectives

The objective of this work is to overcome the aforementioned weaknesses of existing relative preference-based RecSys, as well as resolving the heterogeneous data sources issue of traditional RecSys. More specifically, the research objectives of this work are:

- *Learning Local and Global Structures*: Capturing both the local and global structures of user preferences have been done in rating-based recommendation techniques [38]. However, existing approaches are not directly applicable to relative preference-based RecSys as the format of input has changed. Recent advances in *Markov Random Fields*-based RecSys [79] have made it possible to capture both structures in a principled way by utilizing the flexibility of graphical models. This thesis will investigate how the two structures can be compiled into a single model in a probabilistic manner.

- *Incorporating Side Information*: How to incorporate side information such as

item content and user attributes is a problem for relative preference-based RecSys. In fact, no existing relative preference-based RecSys has attempted this task as these models are designed particularly for user preferences and have no flexibility to incorporate side information in a proper way. On the other hand, *Conditional Random Fields* [79], as an the extended version of *Markov Random Fields*, can easily incorporate side information in a probabilistic manner. However, it remains unknown how *Conditional Random Fields* can accept relative preferences as input. This thesis will investigate how to design a relative preference-based *Conditional Random Fields* model.

- *Learning from Heterogeneous Data Sources*: How to unify user preferences in different formats has been a problem for traditional rating-based RecSys. The main difficulty is that there is no suitable method to convert user preferences among formats without introducing noises. Furthermore, some conversions are impractical, such as converting *mouse clicks* to 5-*star ratings*. Fortunately, the relative preference provides a unified interface for all kinds of user preferences, where both *mouse clicks* and 5-*star ratings* can be converted into pairwise preference relations. In this thesis, we will investigate how to learn from heterogeneous data sources using relative preference-based RecSys.

1.3 Overview of the Proposed Methodology

Firstly, to address the problem of learning both local and global structures, we propose two *Markov Random Fields*-based models to capture and unify both the *LS* and *GS* information. Specifically, the proposed model employs *Markov Random Fields*

(MRF) to investigate the *LS* information while the *Ordinal Matrix Factorization* (OMF) captures the *GS* information. In this way, we take advantages of both the representational power of the *MRF* and the ease of modeling ordinal preferences by the *OMF*. Experimental result on public datasets demonstrates that the proposed model can capture both types of interactions, resulting in improved recommendation accuracy. On the other hand, when the input format is pairwise preference relation, the *Preference Relation-based Markov Random Fields* model is proposed to deal with the input format of pairwise comparisons of items.

Secondly, to address the problem of incorporating side information, we extended the proposed *Markov Random Fields*-based models to *Conditional Random Fields*-based models, in which the side information are modeled as global observations of the graphical models. We performed experiments on public datasets and demonstrate that side information has been properly incorporated, and significantly improved recommendation performance has been achieved and validated by statistical tests.

Finally, to address the problem unifying information from heterogeneous data sources, we employed the several models to convert and exploit user preferences of different formats. Specifically, all types of user preferences are converted into the unified preference relation format and modeled by our proposed models. Experiment results on public datasets demonstrate that our solutions to unifying data from heterogeneous sources have successfully minimized the noises information introduced, resulting improved recommendation quality, especially in cold-start cases where each data source provides a limited amount of data.

1.4 Thesis Outline

This section presents the overall organization of this thesis. As the objective of this thesis is to address the problems of relative preference-based RecSys, the content of each chapter is organized as follows:

- ***Chapter 2*** presents a comprehensive survey on recommender systems in general with a focus on relative preference-based RecSys. Specifically, the relevant concepts, assumptions, and emerging research issues in this area will be discussed. Efforts have been made to identify current and future issues of relative preference-based RecSys.

- ***Chapter 3*** focuses on resolving the issue of learning from both the local and global structures in ordinal user preferences. This chapter specifically investigates the scenario of using ordinal type of preference as input. An *Ordinal Random Fields* (ORF) method is proposed to capture and unify both types of structures in a principled way. Experiments on multiple public datasets are conducted to show that the proposed method effectively improves the performance of recommendation by utilizing both types of structures.

- ***Chapter 4*** proposes a novel *Preference Relation-based Markov Random Fields* model to address the issue of learning from both the local and global structures in preference relations. This chapter also proposes a *Preference Relation-based Conditional Random Fields* model, which incorporates side information of users and items. The proposed model does not rely on ratings but pairwise comparisons of items, thus offers better reliability and can be applied to a wider range of applications. To validate its performance, we conducted experiments on several

public datasets together with side information, and performance improvements have been confirmed statistically.

- ***Chapter 5*** addresses the issue of learning from multiple heterogeneous data sources. This chapter identifies and formalizes the heterogeneous data sources problem, and proposes the preference relation-based method to unify heterogeneous data. With consideration of multiple data sources, the proposed method can reduce the effect of cold-start problem where each data source provides limited amount of data. With the help of the proposed method, implicit user preferences such as *page views* and *mouse clicks* can be easily exploited to alleviate the cold-start problem.

- ***Chapter 6*** summarizes the contributions of this thesis, as well as discusses some possible extensions and directions of future research.

To maintain readability, some essential concepts, definitions, and motivations are recounted in each chapter to make it self-contained. For basic concepts of recommender systems, readers may refer to the recommender system handbook [66].

Chapter 2

Literature Review

This chapter is devoted to provides an extensive literature review on recommender systems by racing the trends and directions of current research. We chronologically review contributions along each research direction regarding recommender systems with a focus on relative preference-based RecSys. Specifically, Section 2.1 introduces the basic notations and related concepts of recommender systems. Section 2.2 reviews popular recommendation techniques along with the latest developments. Section 2.3 focuses on relative preference-based RecSys, and will introduce the recent developments on this emerging topic. Section 2.4 presents evaluation metrics that are used to evaluate recommendation performance.

2.1 Notation and Problem Formulation

RecSys use historical data to predict future interest in *items* by *users*. Two objects are involved in RecSys: *items* and *users*. Let $\mathcal{U} = \{u_1, u_2, ..., u_m\}$ denote a set of m *users*, and $\mathcal{T} = \{t_1, t_2, ..., t_n\}$ denote a set of n *items*, such as *books*, *movies*, etc. The interest of user $u \in \mathcal{U}$ in item $t \in \mathcal{T}$ is encoded as the preference $r_{u,t} \in R$, where

R captures the known preferences for all users $\mathcal{U}$. A typical form of preferences is the *ratings* (e.g. $1-5$ stars), though many other forms exist, such as *like/dislike*, *clicked/not clicked*, etc.

Definition 1 (Recommender System). Given item collection $\mathcal{T}$ and known preferences R of all users $\mathcal{U}$, RecSys aims to identify the item $\widehat{t} \in \mathcal{T}$ that maximizes the preference $r_{u_a,t}$ of the *active user* $u_a \in \mathcal{U}$ [1]:

$$\widehat{t} = \arg\max_{t \in \mathcal{T}}(r_{u_a,t}) \tag{2.1.1}$$

This definition often implies that individual items are suggested to the individual users, however, real-world applications may require suggesting a set of items and/or to a group of users. To handle such cases, Definition 1 needs to be extended, however, it remains a challenging task to making recommendations to groups of users or recommending a set of items. For ease of reference, notations used by rating-based RecSys are summarized in Table 2.1.

Over the last decade, the development on RecSys has been carried out along two research lines: *Recommendation Techniques* and *Evaluation Metrics*. Works on the *recommendation techniques* focus on *how to generate recommendations based on various information sources*, ranging from the item *content*, the known preferences, to more recent sources such as the *context* [2] and the *social trust* [28].

After the recommendations have been generated, the next task is to *evaluate the quality of recommendations* using *evaluation metrics*. Evaluating common *machine learning* tasks such as *classification* are in general less difficult as the ground truth is available to assess the predictions. *Accuracy metrics* such as *mean absolute error* (MAE) are often employed to assess the performance of *machine learning* tasks as well as the RecSys. However, it becomes tricky in RecSys where the ground truth

Table 2.1: Notations used in rating-based RecSys

Notations	Mathematical Meanings
$\mathcal{U}$	set of all users $\mathcal{U} = \{u_1, u_2, ..., u_n\}$
$\mathcal{T}$	set of all items $\mathcal{T} = \{t_1, t_2, ..., t_m\}$
$\mathcal{G}$	a user group, $\mathcal{G} \subseteq \mathcal{U}$
$\mathcal{K}$	an item package, $\mathcal{K} \subseteq \mathcal{T}$
R	available preferences data of all users
$r_{u,t}$	the preference of user u on item t
$R(u_x)$	the set of items rated by user u_x
$S(t_x)$	the set of users rated item t_x
T_{xy}	the set of items co-rated by user u_x and u_y
$r_{\mathcal{G},\mathcal{K}}$	the group $\mathcal{G}$'s preference on package $\mathcal{K}$
$I_r(\mathcal{G}, \mathcal{K})$	the *inter-relevance* between group $\mathcal{G}$ and package $\mathcal{K}$
$I_h(\mathcal{K})$	the aggregation of *inherence* properties of package $\mathcal{K}$
$\lvert\cdot\rvert$	the cardinality of the set
$\widehat{\cdot}$	the prediction, e.g. $\widehat{t}$ is the predicted item t
$\bar{\cdot}$	the arithmetic mean

(user's satisfaction) may not be well represented by the preferences data such as ratings. For example, a user rated 5-star for the movie `Titanic` but he/she may not want to watch `Titanic Extended Version`, but a RecSys focuses on ratings may still consider `Titanic Extended Version` as a 5-star recommendation. For this reason, research of RecSys has gone beyond the *accuracy metrics* to many novel metrics such as *diversity* [11], *novelty* [25], etc. Figure 2.1 provides an overview of *recommendation techniques* and *evaluation metrics* in RecSys.

2.2 Recommendation Techniques

Recommendation techniques aims to identify the right item for the user, where two fundamental approaches are *Content-based* methods [62] and *Collaborative Filtering* methods [65]. Conventionally, *Content-based* methods generate recommendations by exploiting regularities in the item *content*, while *Collaborative Filtering* methods generate recommendations based on available preferences data of users. More recent approaches are exploiting extra information such as the *context* [2] and the *social trust* [28]. In this section, we briefly review these recommendation techniques.

2.2.1 Content-based Recommender Systems

Content-based methods generate recommendations for the active user u_a based on the contents of related items, where other users' information is not utilized. The basic idea is to identify the unrated items that are similar to the active user's highly rated items.

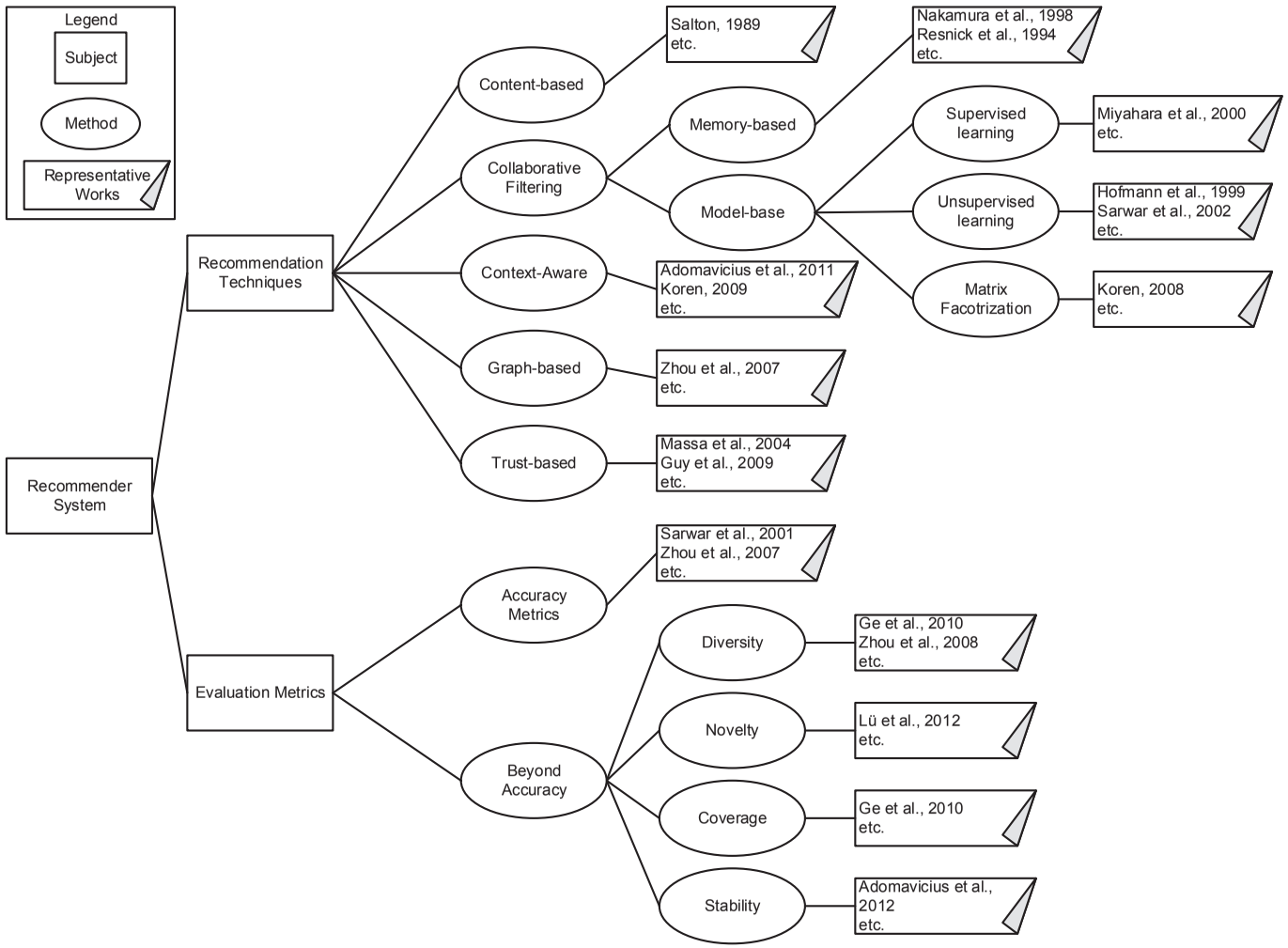


Figure 2.1: An overview of Recommender System

The prediction of active user's preference $r_{u_a,t}$ on unrated item t is calculated based on known preferences of items similar to t. The *similarity* between two items is measured by comparing the *content* of the items. For example, two movies can be compared in terms of the *actors*, *directors*, *genres*, etc [47]. For text-based items, the features can be represented by *keywords* using *term frequency/inverse document frequency* (TF-IDF) [67]. Given the features, the similarity can be calculated using standard metrics such as the *cosine* distance.

Despite of the simplicity, content-based methods have three limitations. Firstly, it can be difficult to define features or extract content from some types of items, such as *audio*, *videos*, and *pictures*. Secondly, the user will always be recommended with items that are highly similar to the items he/she liked, which leads to the lacking of *diversity* [11]. Finally, it is difficult to identify items for new users or users with few ratings, and this is referred to as the *cold-start* problem [72].

2.2.2 Collaborative Filtering

Collaborative Filtering (CF) looks for items highly rated by users *similar* to the active user. CF methods can be classified into two classes: *memory-based* methods and *model-based* methods.

Memory-based Methods In *memory-based* methods [59, 65], the preference predictions are based on the entire collection of known preferences. The idea is that similar users should rate the same movie similarly. The preference $r_{u_a,t}$ of unrated item t for active user u_a is calculated based on the preference $r_{u_j,t}$ from every user $u_j \in U$ who is similar to the active user u_a. The *similarity* between two users is defined by comparing their known preferences, and two

popular measures are *Pearson Correlation Coefficient* (PCC) [65] and *Vector Space Similarity* (VS) [1]:

Pearson Correlation Coefficient (PCC)

$$sim(u_x, u_y) = \frac{\sum_{t_i \in T_{xy}} (r_{xi} - \bar{r}_x)(r_{yi} - \bar{r}_y)}{\sqrt{\sum_{t_i \in T_{xy}} (r_{xi} - \bar{r}_x)^2 \sum_{t_i \in T_{xy}} (r_{yi} - \bar{r}_y)^2}} \tag{2.2.1}$$

Vector Space Similarity (VS)

$$sim(u_x, u_y) = cos(\mathbf{u_x}, \mathbf{u_y}) = \frac{\sum_{t_i \in T_{xy}} r_{xi} r_{yi}}{\sqrt{\sum_{t_i \in T_{xy}} r_{xi}^2 \sum_{t_i \in T_{xy}} r_{yi}^2}} \tag{2.2.2}$$

where $T_{xy} = \{t_i \in T | r_{xi} \neq \emptyset, r_{yi} \neq \emptyset\}$ denotes the set of items co-rated by both u_x and u_y.

Model-based Methods In contrast to *memory-based* methods, *model-based* methods [57,68] will construct a model from the known preferences, and make future recommendations based on the model. *Model-based* methods often take more training time than *memory-based* methods, however, they are more efficient in generating recommendations. According to the type of model, *model-based* methods can be further divided into three classes: *supervised learning-based* [57], *unsupervised learning-based* [31, 70], and *matrix factorization-based* [9, 38].

Similar to *content-based* methods, CF also suffers from *cold-start* problem. In addition, new items rated by a small number of users will have a low chance to be recommended. However, CF has been one of the most popular recommendation techniques for to its efficiency and high quality recommendations.

2.2.3 Context-Aware Recommendation

Both content-based methods and CF focus on the preferences, however, users' interests could also be affected by the *context*. For example, whether a user would like a movie not only depends on the user's taste, but also the *context* such as *when*, *where*, and *with whom*.

One of the first considered *context* is *temporal* information. In 2001, Zimdars et al. [86] treated CF as a uni-variate time series problem, where a user's next preference is predicted based on the previous preference. However, *temporal* information did not attract much attention until the successful of *timeSVD++* method [39] in *Netflix Progress Prize* competition. The *timeSVD++* method predicts preference of active user u_a on item i at time t as:

$$\hat{r}_{ai}(t) = \mu + b_i(t) + b_a(t) + q_i^T \left(p_a(t) + |R(u_a)|^{\frac{1}{2}} \sum_{j \in R(u_a)} y_j \right), \tag{2.2.3}$$

where μ denotes the overall average preference, $b_i(t)$ is the item's bias at time t, $b_a(t)$ is the user's bias at time t, $R(u_a)$ is the set of items rated by user u_a, q_i and y_j are item-factor vectors, and $p_a(t)$ is the user-factor vector at time t, $p_a(t)$, q_i, and y_j are in a joint latent factor space as used in matrix factorization techniques [38]. In this formulation, temporal information is modeled by the time-based bias, and this makes it superior to other competitors.

Recently, it has been recognized that *temporal* is not the only important *context* and various kinds of *context* can be exploited to improve the recommendation quality, and this kind of RecSys is referred to as *Context-Aware Recommender Systems* [2].

2.2.4 Graph-based Recommendation

Graph-based methods consider the recommendation task as a *link prediction* problem of *bipartite graph* [85]. On *bitpartite* graph, users $\mathcal{U}$ and items $\mathcal{T}$ are represented by two sets of nodes, and each pair of $< user, item >$ can be connected with an edge. These edges represent users' interests in items, and making recommendations is the same as connecting the missing edges.

A representative work is the *Network-Based Inference* (NBI) proposed by Zhou et al. [85] which generates recommendations based on the *resource-allocation process*. To make predictions for user u_a, NBI first initializes the network as:

$$AC(u_a, t_i) = \begin{cases} 1 & \text{if } t_i \in R(u_a) \\ 0 & \text{otherwise} \end{cases} \tag{2.2.4}$$

where $R(u_a)$ denotes the set of items rated by user u_a, and $AC(u_a, t_i)$ is the *Allocated Resource* (AC) to node of item t_i that represents user's interests. In this initialization, 1 is assigned to every item t_i if rated by user u_a, and 0 otherwise. After this initialization for all users, the allocated resources will be redistributed among all items in the following two steps and the item with the most AC at the final stage will be recommended to the active user u_a.

Spreading Step In spreading step, all initially allocated resources will flow from all items $\mathcal{T}$ to all users $\mathcal{U}$. The resources flow to each user u_x is calculated as:

$$AC(u_x) = \sum_{i=1}^{|\mathcal{T}|} \frac{c_{xi} AC(u_x, t_i)}{|S(t_i)|} \tag{2.2.5}$$

where $S(t_i)$ denotes the set of users who rated item t_i, $AC(u_x, t_i)$ denotes the initial resources allocated to item t_i by user u_x, and c_{xi} is 1 if u_x rated item t_i and 0 otherwise.

Redistribution Step In this step, the resources will flow back to items $\mathcal{T}$ from users $\mathcal{U}$. The final resources allocated to each item t_i is calculated as:

$$\widehat{AC(t_i)} = \sum_{x=1}^{|\mathcal{U}|} \frac{c_{xi} AC(u_i)}{|\mathcal{T}_x|} \tag{2.2.6}$$

The item with most resources $\widehat{AC(t_i)}$ allocated will be recommended to the active user u_a.

2.2.5 Trust-based Recommendation

Similarity in typical recommendation methods is often defined by standard metrics such as *cosine*. However, instead of finding recommendations from *similar* users, it is also reasonable to find recommendations from *familiar* users [28]. Intuitively, an item liked by the user's good friend has the potential to be liked by the user.

Recent developments in social networks have further revealed the social trust relationships among users, and Massa and Avesani [53] termed this kind of recommender systems as *Trust-Aware Recommendation*. Empirical results from Guy's work [28] indicated that *familiarity*-based methods can be superior to *similarity*-based methods. Despite of the performance comparison, the key advantage of *trust-aware* methods is that it provides a promising approach to *cold-start* problem [27, 28].

2.3 Relative Preference-based Recommender Systems

User preferences can be modeled in three types: *pointwise*, *pairwise*, and *listwise*. Though RecSys is not limited to *pointwise* absolute ratings, the recommendation task is usually considered as a rating prediction problem [38, 40, 69, 78]. Recently, a considerable literature [12, 19, 45, 64, 74] has grown up around the theme of *relative* preferences, especially the *pairwise PR*. Meanwhile, recommendation task is also shifting from rating prediction to item ranking [56, 74, 83] in which the ranking itself is also *relative* preferences.

The use of *relative* preferences has been widely studied in the field of *Information Retrieval* for *learning to rank* tasks [21,22,35]. Recently, *PR*-based [12,19,45,64] and *listwise*-based [74] RecSys have been proposed. Among them, the *PR*-based approach is the most popular, which can be further categorized as *memory-based* methods [12] that capture *local structure* and *model-based* methods [19, 45, 64] that capture *global structure*. We summarize the capabilities of the existing methods in Table 2.2.

Table 2.2: Capabilities of existing methods

Method	Input	Output	LS	GS
Pointwise Memory-based	Ratings	Ratings	✓	
Pointwise Model-based	Ratings	Ratings		✓
Pointwise Hybrid	Ratings	Ratings	✓	✓
Pairwise Memory-based	Preference Relations	Item Rankings	✓	
Pairwise Model-based	Preference Relations	Item Rankings		✓

2.3.1 Notation and Problem Statement

Preference Relation

A *preference relation* (PR) encodes user preferences in form of pairwise ordering between items. This representation is a useful alternative to absolute ratings for three reasons.

Firstly, *PR* is more consistent across like-minded users [12, 19] as it is invariant to many irrelevant factors, such as *mood*. Secondly, *PR* is a more natural and direct input for Top-N recommendation, as both the input and the output are relative preferences. Finally, and perhaps most importantly, *PR* can be obtained implicitly rather than asking the users explicitly. For example, the *PR* over two *Web pages* can be inferred by the *stayed time*, and consequently applies to the displayed items. This property is important as not all users are willing to rate their preferences, where collecting feedbacks implicitly delivers a more user-friendly RecSys. In addition, *PR*-based RecSys provides an opportunity to utilize the vast amount of implicit data that have already been collected over the years, such as *activity logs*. With these potential benefits, we shall take a closer look at the *PR*, and investigate how they can be utilized in RecSys.

We formally define the *PR* as follows. Let $\mathcal{U} = \{u\}^n$ and $\mathcal{I} = \{i\}^m$ denote the set of n users and m items, respectively. The preference of a user $u \in \mathcal{U}$ between items i and j is encoded as π_{uij}, which indicates the strength of user u's *PR* for the ordered item pair (i, j). A higher value of π_{uij} indicates a stronger preference on the first item over the second item.

Definition 2 (Preference Relation). The preference relation is defined as

$$\pi_{uij} = \begin{cases} (\frac{2}{3}, 1] & \text{if } i \succ j \text{ (u prefers i over j)} \\ [\frac{1}{3}, \frac{2}{3}] & \text{if } i \simeq j \text{ (i and j are equally preferable to u)} \\ [0, \frac{1}{3}) & \text{if } i \prec j \text{ (u prefers j over i)} \end{cases} \tag{2.3.1}$$

where $\pi_{uij} \in [0, 1]$ and $\pi_{uij} = 1 - \pi_{uji}$.

This definition is similar to [19], however, we allocate an interval for each preference category, i.e., *preferred*, *equally preferred*, and *less preferred*. Indeed, each preference category can be further break down into more intervals.

Similar to [12], the *PR* can be converted into user-wise preferences over items.

Definition 3 (User-wise Preference). The user-wise preference is defined as

$$p_{ui} = \frac{\sum_{j \in \mathcal{I}_u} [\![\pi_{uij} > \frac{2}{3}]\!] - \sum_{j \in \mathcal{I}_u} [\![\pi_{uij} < \frac{1}{3}]\!]}{|\Pi_{ui}|} \tag{2.3.2}$$

where $[\![\cdot]\!]$ gives 1 for *true* and 0 for *false*, and Π_{ui} is the set of user u's PR related to item i.

The user-wise preference p_{ui} falls in the interval $[-1, 1]$, where -1 and 1 indicate that item i is the least or the most preferred item for user u, respectively. The user-wise preference measures the relative position of an item for a particular user, which is different from absolute ratings.

Preference relation has been widely studied in the field of *Information Retrieval* [14, 21, 22, 35]. Nevertheless, *PR*-based RecSys have only emerged recently [12, 19, 45, 64].

Problem Statement

Generally, the task of *PR*-based RecSys is to take *PR* as input and output Top-N recommendations. Specifically, let $\pi_{uij} \in \Pi$ encode the *PR* of each user $u \in \mathcal{U}$.

Each π_{uij} is defined over an ordered item pair $(i,\ j)$, denoting $i \prec j$, $i \simeq j$, or $i \succ j$ as described in Eq. 2.3.1. The goal is to estimate the value of each unknown $\pi_{uij} \in \Pi_{unknown}$, such that $\hat{\pi}_{uij}$ approximates π_{uij}. This can be considered as an optimization task performs directly on the *PR*:

$$\hat{\pi}_{uij} = \underset{\hat{\pi}_{uij} \in [0,1]}{\arg\min} (\pi_{uij} - \hat{\pi}_{uij})^2 \tag{2.3.3}$$

However, it can be easier to estimate the $\hat{\pi}_{uij}$ by the difference between the two user-wise preferences p_{ui} and p_{uj}, i.e., $\hat{\pi}_{uij} = \phi(\hat{p}_{ui} - \hat{p}_{uj})$, where $\phi(\cdot)$ is a function that bounds the value into $[0, 1]$ and ensures $\phi(0) = 0.5$. For example, the *inverse-logit* function $\phi(x) = \frac{e^x}{1+e^x}$ can be used when user-wise preferences involve large values. Therefore, the objective of this paper is to solve the following optimization problem:

$$(\hat{p}_{ui}, \hat{p}_{uj}) = \underset{\hat{p}_{ui}, \hat{p}_{uj}}{\arg\min} (\pi_{uij} - \phi(\hat{p}_{ui} - \hat{p}_{uj}))^2 \tag{2.3.4}$$

which optimizes the user-wise preferences directly, and Top-N recommendations can be obtained by simply sorting the estimated user-wise preferences.

Let us consider an instance space $X = \{x_i\}$ (e.g. items) and a finite set of labels (e.g. ratings) $Y = \{y_i | i = 1, 2, ..., k\}$. One task of preference learning is to find a *label ranking* for any instance, e.g., determine the most likely rating for an item. The other task is to find an *object ranking*, e.g., to determine the ranking of items.

For ease of reference, notations used in Relative Preference-based RecSys are summarized in Table 2.3. The letters u, v, a, b represent *users*, and the letters i, j, k, l represent *items*.

Table 2.3: Notations used in Relative Preference-based RecSys

Notations	Mathematical Meanings
$\mathcal{U}$	the set of users
$\mathcal{I}$	the set of items
Π	the set of preference relations
p_{ui}	the user-wise preference of user u on item i
$\mathcal{G}$	an undirected graph encodes relations of user-wise preferences
$\mathcal{V}$	the set of vertices each represents a user-wise preference
$\mathcal{E}$	the set of edges each connects two vertices
f_{uv}	the correlation feature between users u and v
f_{ij}	the correlation feature between items i and j
w_{uv}	the weight associated to the user-user correlation feature f_{uv}
w_{ij}	the weight associated to the item-item correlation feature f_{ij}
$Q(p_{ui} \mid u, i)$	the ordinal distribution
$\mathbf{o}$	the side information, e.g., user attributes and item content

2.3.2 Memory-based Models

A memory-based model is proposed in [12] to take preference relations as input and compute similarities between users. The proposed model has the following three steps:

Collecting User Profiles

When preference relations are employed, four values are possible for user preferences:

- $i \succ j$ indicates item i is preferred over item j
- $i \prec j$ indicates item j is preferred over item i
- $i \approx j$ indicates item i and j are equally preferable

Each value correspond to one question answered by a user. However, there are too many possible questions which cannot be asked. Therefore, a decision must be made to decide which subset of questions to ask.

Computing Similarities Between Users

Let I_u be the set of preference relations of user u, and $f_{u_1,u_2}(i,j)$ indicating whether two users u_1 and u_2 agree on their preference on the two items i and j. Given an item pair (i,j), the function $f_{u_1,u_2}(i,j)$ gives the value 1 if the two users have the same preference, and 0 otherwise. The similarity measure between two users u_1 and u_2 is then defined as:

$$\begin{aligned} cos_{\succeq}(u_1,u_2) &= \frac{\sum_{(i,j)\in I_1\cap I_2} f_{u_1,u_2}(i,j)}{\sqrt{\sum_{(i,j)\in I_1} f_{u_1,u_1}(i,j)} \cdot \sqrt{\sum_{(i,j)\in I_2} f_{u_2,u_2}(i,j)}} \\ &= \frac{\sum_{(i,j)\in I_1\cap I_2} f_{u_1,u_2}(i,j)}{\sqrt{|I_1| \cdot |I_2|}} \end{aligned} \tag{2.3.5}$$

where the numerator represents the number preferences that both users agreed, and the denominator normalizes the result.

Making Recommendations

To make recommendations, the preference relations are first converted into user-wise preferences. Denote:

- $c^{+}_{u,i}$: the number of preference relations that i is preferred
- $c^{=}_{u,i}$: the number of preference relations that i is equally preferred to others
- $c^{-}_{u,i}$: the number of preference relations that i is less preferred

Then the user-wise preference of item i by user u is defined as:

$$p_{ui} = \frac{c^{-}_{u,i} - c^{+}_{u,i}}{c^{-}_{u,i} + c^{+}_{u,i} + c^{=}_{u,i}} \tag{2.3.6}$$

With the user-wise preferences computed, the preference over an unknown item j can be predicted by:

$$p_{uj} = \frac{\sum_{v \in N_u} sim(u,v) \cdot p_{vj}}{\sum_{v \in N_u} sim(u,v)} \tag{2.3.7}$$

where N_u is the set of users that have similar profiles to user u.

2.3.3 Model-based Models

Ordinal Matrix Factorization

The ordinal nature of preferences has been overlooked in *RecSys* literature, until recently *Ordinal Matrix Factorization* (OMF) [32, 42, 61, 82] has emerged to explore the ordinal properties of ratings.

In general, *OMF* aims to generate an ordinal distribution $Q(r_{ui}|u,i)$ over all possible rating values for each user/item pair. Predicting the rating for user u on item i is then equivalent to identifying the rating with the greatest mass in the ordinal distribution $Q(r_{ui}|u,i)$. While traditional *RecSys* approaches make only a point estimate, the *OMF* produces a full distribution and each prediction is associated with a probability as a *confidence* measure.

Typical *OMF* approaches assume the existence of a *latent utility* x_{ui} that captures how much the user u is interested in the item i. The latent utility x_{ui} can be defined in different ways [32, 42, 61, 82], but under the same framework of *Random Utility Models* [55]

$$x_{ui} = \mu_{ui} + \epsilon_{ui} \tag{2.3.8}$$

where μ_{ui} is an internal score represents the interaction between the user u and the item i. The ϵ_{ui} is the random noise normally assumed to follow the logistic distribution in practice [42]. The latent utility x_{ui} is then generated from a logistic distribution centred at μ_{ui} with the scale parameter s_{ui} proportional to the standard deviation

$$x_{ui} \sim Logi(\mu_{ui}, s_{ui}) \tag{2.3.9}$$

In collaborative filtering, the user-item interaction is often captured by *MF* techniques, thereby the internal score μ_{ui} can be substituted with the *MF* term $b_{ui} + \mathbf{p}_u^T \mathbf{q}_i$

$$x_{ui} = b_{ui} + \mathbf{p}_u^T \mathbf{q}_i + \epsilon_{ui} \tag{2.3.10}$$

where $\mathbf{p}_u$ and $\mathbf{q}_i$ are, respectively, the latent feature vectors of the user u and the item i. Modelling the latent utility with *MF* reflects the name *OMF*.

Despite how the latent utility is modelled, an *ordinal assumption* is required to convert the numerical utility into ordinal values. A common approach is the *ordinal*

logistic regression originally described by McCullagh [54], which assumes that the rating is chosen based on the interval to which the utility belongs

$$r_{ui} = l \text{ if } x_{ui} \in (\theta_{l-1}, \theta_l] \text{ for } l < L \text{ and } r_{ui} = L \text{ if } x_{ui} > \theta_{L-1} \tag{2.3.11}$$

where L is the number of ordinal levels and θ_l are the threshold values of interest. Other assumptions [51] are also possible but McCullagh's model is by far the most popular. The probability of receiving a rating l is therefore

$$Q(r_{ui} = l|u, i) = \int_{\theta_{l-1}}^{\theta_l} P(x_{ui}|\theta) = F(\theta_l) - F(\theta_{l-1}) \tag{2.3.12}$$

where $F(\theta_l)$ is the cumulative logistic distribution evaluated at θ_l

$$F(x_{ui} \leq l|\theta_l) = \frac{1}{1 + \exp(-\frac{\theta_{uil} - \mu_{ui}}{s_{ui}})} \tag{2.3.13}$$

where the thresholds θ_l can be parameterised to depend on user or item. This paper employs the user-specific thresholds parameterisation described in [42]. Therefore a set of thresholds $\{\theta_{ul}\}_{l=1}^{L}$ is defined for each user u to replace the thresholds θ_{uil} in Eq. 2.3.13.

Given the learned ordinal distribution $Q(r_{ui}|u, i)$, not only the ratings can be predicted but also the *confidence* for each prediction.

Preference Relation-based Matrix Factorization

Matrix Factorization (MF) [41] is a popular approach to RecSys that has mainly been applied to absolute ratings. Recently, the *PrefNMF* [19] model was proposed to adopt *PR* input for *MF* models. The *PrefNMF* model discovers the latent factor space shared between users and items, where the latent factors describe both the *taste* of users and the *characteristics* of items. The attractiveness of an item to a user is then measured by the inner product of their latent feature vectors.

Formally, each user u is associated with a latent feature vector $\mathbf{u}_u \in \mathbb{R}^k$ and each item i is associated with a latent feature vector $\mathbf{v}_i \in \mathbb{R}^k$, where k is the dimension of the latent factor space. The attractiveness of items i and j to the user u are $\mathbf{u}_u^\top \mathbf{v}_i$ and $\mathbf{u}_u^\top \mathbf{v}_j$, respectively. When $\mathbf{u}_u^\top \mathbf{v}_i > \mathbf{u}_u^\top \mathbf{v}_j$ the item i is said to be more preferable to the user u than the item j, i.e., $i \succ j$. The strength of this preference relation π_{uij} can be estimated by $\mathbf{u}_u^\top(\mathbf{v}_i - \mathbf{v}_j)$, and the *inverse-logit* function is applied to ensure $\hat{\pi}_{uij} \in [0, 1]$:

$$\hat{\pi}_{uij} = \frac{e^{\mathbf{u}_u^\top(\mathbf{v}_i - \mathbf{v}_j)}}{1 + e^{\mathbf{u}_u^\top(\mathbf{v}_i - \mathbf{v}_j)}} \tag{2.3.14}$$

The latent feature vectors $\mathbf{u}_u$ and $\mathbf{v}_i$ are learned by minimizing regularized squared error with respect to the set of all known preference relations Π:

$$\min_{\mathbf{u}_u, \mathbf{v}_i \in \mathbb{R}^k} \sum_{\pi_{uij} \in \Pi \wedge (i<j)} (\pi_{uij} - \hat{\pi}_{uij})^2 + \lambda(\|\mathbf{u}_u\|^2 + \|\mathbf{v}_i\|^2) \tag{2.3.15}$$

where λ is the regularization coefficient. The optimization can be done with *Stochastic Gradient Descent* for the favor of speed on sparse data, or with *Alternating Least Squares* for the favor of parallelization on dense data.

2.4 Evaluation Metrics

Classic problems such as classification often have some agreed evaluation metrics. However, recommendation techniques are evaluated in many different ways depending on the form of recommendation as well as the goal of recommendation. For example when predicting a user's rating on a movie, accuracy metrics are often used to measure how close the predicted rating is to the true rating. On the other hand, if the RecSys predicts a ranking of items for a user, then other metrics will be required to measure

the correctness, diversity, etc. In this section, we describe some commonly used evaluation metrics for both rating and ranking based RecSys.

2.4.1 Accuracy Metrics

To measure the recommendation quality, various accuracy metrics can be used. Two popular metrics are *Mean Absolute Error* (MAE) and *Root Mean Squared Error* (RMSE), which measure how close the prediction is to the ground truth Let $\sum_a |R(u_a)|$ be the number of unrated items by user u_a, and $\hat{r}_i$ be the predicted rating of item t_i, the definition of MAE and RMSE are as follows:

$$MAE = \frac{\sum_{a,i} |\hat{R}_{x,i} - R_{a,i}|}{\sum_a |R(u_a)|} \tag{2.4.1}$$

$$RMSE = \sqrt{\frac{\sum_{a,i} |\hat{R}_{a,i} - R_{a,i}|}{\sum_a |R(u_a)|}} \tag{2.4.2}$$

MAE and RMSE are the commonly used metric in literature [68, 69] as well as in various competitions [7]. However, the prediction accuracy can also be measured in terms of correlations between the predicted and the ground truth. Different correlation measures exist and a popular one is *Pearson Correlation Coefficient* (PCC) defined as:

$$PCC = \frac{\sum_a (\hat{R}_a - \overline{\hat{R}})(R_a - \overline{R})}{\sqrt{\sum_a (\hat{R}_a - \overline{\hat{R}})^2} \cdot \sqrt{\sum_a (R_a - \overline{R})}} \tag{2.4.3}$$

Other accuracy metrics are also developed, such as *Accuracy/Precision* [29], and *Area Under ROC Curve* (AUC) [85].

2.4.2 Diversity

Traditionally, the evaluation of RecSys is mainly based on accuracy metrics such as RMSE. However, the accuracy metrics can not evaluate the some properties of the items other than the preferences, such as *Serendipity* [25], *Diversity* [11], etc.

One diversity metric is *Personalization*, in which the uniqueness of each user's recommendation list is measured. Personalization refers the inter-user diversity [84]

$$Personalization = \frac{2}{m(m-1)} \sum_{x \neq y} \left(1 - \frac{|L_k(u_x) \cap L_k(u_y)|}{|L_k(u_x)|}\right), \tag{2.4.4}$$

where m is the number of users, and $(1 - \frac{|L_k(u_x) \cap L_k(u_y)|}{|L_k(u_x)|})$ is the *Hamming* distance between recommendation lists $L_k(u_x)$ and $L_k(u_y)$.

2.4.3 Coverage

Coverage refers to the percentage of items of all items a RecSys can recommend. This metric is based on the observation that some items may not have the chance to be recommended to any user, which reduces the *coverage* of the system.

Let N be the number of top places to be considered, L_d be the number of distinct items in all top-N recommendation lists, and L be the number of distinct items in all recommendation lists. The N-dependent *coverage* is defined as [25]:

$$Coverage(N) = L_d/L \tag{2.4.5}$$

A low *coverage* means the RecSys can only make recommendations on a small number of distinct items, in other words, it always recommends the popular items. It can be shown that RecSys with high *coverage* implies higher *diversity* [48].

2.4.4 Stability

Stability measures consistency of recommendations for the same user [3]. The recommendations generated by a stable RecSys should be similar after some new preferences are added. For example, the first recommendation of an unstable RecSys predicts movie A as 5-star and movie B as 1-star. Then the user watched movie A and rated it as 5-star. With this new preference added to the preferences data, the unstable RecSys then generates the second recommendation that predicts movie B as 5-star. The 5-star movie B which was 1-star, may lead to user confusion and lower the trust of the RecSys. The *Stability* property has been studied in detail in [4].

With various evaluation metrics available, the choice highly depends on the goal of the RecSys. In general, accuracy metrics such as MAE and RMSE are standard metrics for benchmark, and other metrics such as *diversity* and *novelty* are used to fulfill some additional requirements. Unlike other machine learning tasks which have agreed metrics, the evaluation of RecSys has gained great research interests and new metrics are keeping emerged.

2.4.5 Metrics for Relative Preference-based Models

Traditional recommender systems aim to optimize *RMSE* or *MAE* which emphasizes on absolute ratings. However, the ultimate goal of recommender systems is usually to obtain the ranking of items [42], where good performance on *RMSE* or *MAE* may not be translated into good ranking results [42]. Therefore, we employ two evaluation metrics: *Normalized Cumulative Discounted Gain@T* (NDCG@T) [34] which is popular in academia, and *Mean Average Precision@T* (MAP@T) [13] which

is popular in contests [1]. Among them, the NDCG@T metric is defined as

$$\text{NDCG@}T = \frac{1}{K(T)} \sum_{t=1}^{T} \frac{2^{r_t} - 1}{\log_2 (t+1)} \tag{2.4.6}$$

where r_t is the relevance judgment of the item at position t, and $K(T)$ is the normalization constant. The MAP@T metric is defined as

$$\text{MAP@}T = \frac{1}{|\mathcal{U}_{test}|} \sum_{u \in \mathcal{U}_{test}} \sum_{t=1}^{T} \frac{P_u(t)}{min(m_u, t)} \tag{2.4.7}$$

where m_u is the number relevant items to user u, and $P_u(t)$ is user u's precision at position t. Both metrics are normalized to $[0, 1]$, and a higher value indicates better performance.

These metrics, together with other ranking-based metrics, require a set of relevant items to be defined in the test set such that the predicted rankings can be evaluated against. The relevant items can be defined in different ways. In this paper, we follow the same selection criteria used in the related work [12, 38] to consider items with the highest ratings as relevant.

2.5 Summary

Numerous recommendation techniques have been developed over the last decade, ranging from the basic ones of content-based methods to the recent ones of context-based methods. These recommendation techniques performed well in real-world applications such as *Amazon*, *MovieLens*, and *Netflix*. However, due to the extensive use of these rating-based datasets, existing models are specifically designed for the ratings format, whereas vast amount of implicit feedback such as log files has been

[1] KDD Cup 2012 and Facebook Recruiting Competition

stored but not utilized. The new emerged relative preference-based models provide a solution to make use of such implicit feedback, but lacks of modeling abilities comparing to the well-established rating-based models. In the remaining chapters of this thesis, we identify and tackle weaknesses of relative preference-based models to make them more effective and applicable.

Chapter 3

Ordinal Random Fields for Recommender Systems

3.1 Introduction

Recommender Systems (RecSys) aim to suggest items that are potentially of interest to users, where the items can be virtually anything such as *movies* and *attractions for travel*. To identify the appropriate items, RecSys use various sources of information including *item content* [5] and *user preferences* [41]. By far, *Collaborative Filtering* [41, 69] is one of the most popular RecSys techniques, which exploits user preferences especially the *numerical preferences*.

However, numerical preferences are often difficult to collect as users may find it easier to tell which item is preferable to others, rather than expressing the precise degree of liking. Furthermore, researchers argued that numerical preferences may not be completely trustworthy [12, 42]. For example, the internal scales of users can be different, where the rating 4 out of 5 generally indicates high quality, but it is possible to be just fine for critical users. While users are not good at making consistent quantitative judgment, *ordinal preferences* are considered to be more consistent across

like-minded users [18].

Ordinal preferences is an alternative view of user preferences, in which the relative orders between items are measured. To adopt ordinal preferences, substantial research efforts have been made over the past five years [42, 61, 73, 82]. While most data collections are still dominated by numerical preferences, the shift from numerical to ordinal is a slow process. Instead of going solely ordinal preferences in a sudden, most existing ordinal approaches begin with exploiting the ordinal properties possessed by numerical preferences. Among them, *Ordinal Matrix Factorization* (OMF) has been suggested as an effective method in recent developments [32,42,61,82]. In contrast to the numerical approaches, *OMF* makes weaker assumptions as the user preferences are no longer required to be interpreted as numbers, instead, only the ordering of items matters.

Despite of its effectiveness in modeling ordinal properties, *OMF* is incapable of exploiting the *local structure* described as follows. Typical collaborative filtering methods discover two types of information: the *neighborhoods* and the *latent factors*, which we refer to as the *local* and the *global structures* of the preferences:

Local Structure The *local structure* (LS) refers to the second-order interactions between similar users or items. This type of information is often used by *neighborhood-based* collaborative filtering, in which the predictions are made by looking at the neighborhood of users [65] or items [69]. Though the majority of preferences will be ignored in making predictions, *LS*-based approaches are effective when the users/items correlations are highly localized.

Global Structure The *global structure* (GS) refers to the weaker but higher-order interactions among all users and items. This type of information is often used by

latent factor models such as *SVD* [41] and *LDA* [52], which aim at discovering the *latent factor spaces* in the preferences. *GS*-based approaches are often competitive in terms of accuracy as well as computational efficiency.

Exiting literature has suggested that the *LS* and the *GS* are complementary since they address different aspects of the preferences [38,79]. In 2008, a unified framework has been proposed by Koren [38] to capture both structures, but only for numerical preferences. To the best of our knowledge, there is yet no method for the *OMF* to capture both the *LS* and the *GS*.

Recent advances in *Probabilistic Graphical Models*, especially the *Markov Random Fields* (MRF), have provided methods of building *RecSys* capable of exploiting both the *LS* and the *GS* [79]. However, there has been little attempt to address the ordinal preferences issue due to the complication of modeling ordinal preferences with the *MRF*.

This chapter aims to develop a unified model in which the *OMF* and the *MRF* are seamlessly combined to take advantages of both the representational power of the *MRF* and the ease of modeling ordinal preferences by the *OMF*. The proposed *Ordinal Random Fields* (ORF) model is not designed for a particular *OMF* but can incorporate any *OMF* model that produces ordinal distributions such as those in [32, 42,61,82]. While this work primarily focuses on exploiting the *LS*, the representational power of the *ORF* is by no mean limited to this. For example, the *MRF* employed in *ORF* can be extended to *Conditional Random Fields* (CRF) [43,78] to fuse auxiliary information such as the *item content* [5] and *social relations* [50]. These information has been shown helpful in making better recommendations [6,50], and becomes even more valuable when the preferences data are highly sparse. Besides the extensibility,

the *ORF* inherits other advantages of the probabilistic graphical models as well, such as supporting missing data by its nature, and disciplined learning and inferences techniques.

The remaining part of this chapter is organized as follows. Section 3.2 reviews the basic concepts of the *Matrix Factorization* and the *OMF* which form the basis of this work. Section 3.3 is devoted to the proposed *ORF* model. In Section 3.4, experimental results of the proposed *ORF* model are presented. Finally, Section 3.5 concludes this chapter by summarizing the main contributions and future works.

3.2 Preliminaries

RecSys usually predict *users*' future interest in *items*. Let $\mathcal{U}$ and $\mathcal{I}$, denote the set of all *users* and the set of all *items*, respectively. The interest of the user $u \in \mathcal{U}$ in the item $i \in \mathcal{I}$ is encoded as the preference $r_{ui} \in R$, where the rating matrix R contains all known preferences.

Definition 4 (Recommender System). RecSys aims to identify the item $\hat{i} \in \mathcal{I}$ that maximizes the interest of the *target user* $u \in \mathcal{U}$ [1]

$$\hat{i} = \arg\max_{i \in \mathcal{I}}(r_{ui}) \tag{3.2.1}$$

In the rest of this section, we briefly review two *RecSys* approaches: *Matrix Factorization* and *Ordinal Matrix Factorization* that form a basis of this work For ease of reference, notations used throughout this chapter are summarized in Table 3.1, and the term *preference* and *rating* will be used interchangeably.

Table 3.1: Summary of Major Notations (Chapter 3)

Notations	Mathematical Meanings
$\mathcal{U}$	the set of all users
$\mathcal{I}$	the set of all items
R	the set of known preferences
$\mathcal{G}$	an undirected graph which encodes relations of preferences
$\mathcal{V}$	the set of vertices each represents a preference
$\mathcal{E}$	the set of edges each connects two vertices
$\mathbf{r}_u$	the set of all preferences by user u
f_{ij}	the correlation feature between items i and j
w_{ij}	the weight associated to the correlation feature f_{ij}
L	the number of rating levels, and the ratings are integers from 1 to L

3.2.1 Matrix Factorization

Matrix Factorization (MF) [41] is a popular and accurate approach to *RecSys*. This approach discovers the latent factor spaces shared between users and items, where the latent factors can be used to describe both the *taste* of users and the *characteristics* of items. The attractiveness of an item to a user is then measured by the inner product of their latent feature vectors.

Formally, each user u is associated with a latent feature vector $\mathbf{p}_u \in \mathbb{R}^k$ and each item i is associated with a latent feature vector $\mathbf{q}_i \in \mathbb{R}^k$, where k is the number of factors. The aim of *MF* is then to estimate $\hat{r}_{ui} = b_{ui} + \mathbf{p}_u^T \mathbf{q}_i$ such that $\hat{r}_{ui} \simeq r_{ui}$. The bias term $b_{ui} = \mu + b_u + b_i$ takes the biases into consideration, where μ is the overall average rating, b_u is the user bias, and b_i is the item bias. The latent feature vectors are learned by minimizing regularized squared error with respect to all known preferences

$$\min_{\mathbf{p}_u, \mathbf{q}_i \in \mathbb{R}^k} \sum_{(u,i) \in R} (r_{ui} - b_{ui} - \mathbf{p}_u^T \mathbf{q}_i)^2 + \lambda(\|\mathbf{p}_u\|^2 + \|\mathbf{q}_i\|^2) \tag{3.2.2}$$

where λ is the regularization coefficient. The optimization can be done with *Stochastic Gradient Descent* for the favor of speed on sparse data, or with *Alternating Least Squares* for the favor of parallelization on dense data.

Comparing to *neighbor-based* approaches [69], *MF-based* approaches [38, 40] have shown advantages in terms of accuracy and computational efficiency. Nevertheless, all of these approaches treat the preferences as numerical and are incapable of exploiting ordinal preferences.

3.2.2 Ordinal Matrix Factorization

The ordinal nature of preferences has been overlooked in *RecSys* literature, until recently *Ordinal Matrix Factorization* (OMF) [32, 42, 61, 82] has emerged to explore the ordinal properties of ratings.

In general, *OMF* aims to generate an ordinal distribution $Q(r_{ui}|u, i)$ over all possible rating values for each user/item pair. Predicting the rating for user u on item i is then equivalent to identifying the rating with the greatest mass in the ordinal distribution $Q(r_{ui}|u, i)$. While traditional *RecSys* approaches make only a point estimate, the *OMF* produces a full distribution and each prediction is associated with a probability as a *confidence* measure.

Typical *OMF* approaches assume the existence of a *latent utility* x_{ui} that captures how much the user u is interested in the item i. The latent utility x_{ui} can be defined in different ways [32, 42, 61, 82], but under the same framework of *Random Utility Models* [55]

$$x_{ui} = \mu_{ui} + \epsilon_{ui} \tag{3.2.3}$$

where μ_{ui} is an internal score represents the interaction between the user u and the item i. The ϵ_{ui} is the random noise normally assumed to follow the logistic distribution in practice [42]. The latent utility x_{ui} is then generated from a logistic distribution centered at μ_{ui} with the scale parameter s_{ui} proportional to the standard deviation

$$x_{ui} \sim Logi(\mu_{ui}, s_{ui}) \tag{3.2.4}$$

In collaborative filtering, the user-item interaction is often captured by *MF* techniques, thereby the internal score μ_{ui} can be substituted with the *MF* term $b_{ui} + \mathbf{p}_u^T \mathbf{q}_i$

$$x_{ui} = b_{ui} + \mathbf{p}_u^T \mathbf{q}_i + \epsilon_{ui} \tag{3.2.5}$$

where $\mathbf{p}_u$ and $\mathbf{q}_i$ are, respectively, the latent feature vectors of the user u and the item i. Modelling the latent utility with *MF* reflects the name *OMF*.

Despite how the latent utility is modeled, an *ordinal assumption* is required to convert the numerical utility into ordinal values. A common approach is the *ordinal logistic regression* originally described by McCullagh [54], which assumes that the rating is chosen based on the interval to which the utility belongs

$$r_{ui} = l \text{ if } x_{ui} \in (\theta_{l-1}, \theta_l] \text{ for } l < L \text{ and } r_{ui} = L \text{ if } x_{ui} > \theta_{L-1} \tag{3.2.6}$$

where L is the number of ordinal levels and θ_l are the threshold values of interest. Other assumptions [51] are also possible but McCullagh's model is by far the most popular. The probability of receiving a rating l is therefore

$$Q(r_{ui} = l|u, i) = \int_{\theta_{l-1}}^{\theta_l} P(x_{ui}|\theta) = F(\theta_l) - F(\theta_{l-1}) \tag{3.2.7}$$

where $F(\theta_l)$ is the cumulative logistic distribution evaluated at θ_l

$$F(x_{ui} \leq l|\theta_l) = \frac{1}{1 + \exp(-\frac{\theta_{uil} - \mu_{ui}}{s_{ui}})} \tag{3.2.8}$$

where the thresholds θ_l can be parameterized to depend on user or item. This paper employs the user-specific thresholds parameterization described in [42]. Therefore a set of thresholds $\{\theta_{ul}\}_{l=1}^{L}$ is defined for each user u to replace the thresholds θ_{uil} in Eq. 3.2.8.

Given the learned ordinal distribution $Q(r_{ui}|u, i)$, not only the ratings can be predicted but also the *confidence* for each prediction.

3.2.3 Summary

Matrix Factorization has been one of the most popular *RecSys* approaches, which primarily focuses on numerical preferences such as ratings. Nevertheless, the nature

of user preferences is often ordinal, and the importance of modeling ordinal properties has been recognized in recent works on *OMF* [32, 42, 61, 82]. Although the *OMF* enables the modeling of ordinal properties, the employment of *MF* makes it only focuses on the higher-order interactions (the *GS*) regardless of the localized interactions (the *LS*), whereas both information are valuable [38, 79]. Furthermore, the *OMF* by its nature cannot model auxiliary information such as *content* [5] directly.

The powerful representation of *Markov Random Fields* (MRF) offers an opportunity to take advantages from all of these information, and have been developed in recent works [78, 79]. Nevertheless, exploiting the ordinal properties is not an easy task for *MRF* [79], therefore the strengths of the *OMF* and the *MRF* are nicely complementary. This observation leads to a naturally extension of unifying these two approaches, and motivates the present work.

3.3 Ordinal Random Fields

In this section, we propose the *Ordinal Random Fields* (ORF) to model the ordinal properties and capture both the *LS* and the *GS*. Here we exploit the *LS* of the item-item correlations only, while the user-user correlations can be modeled in a similar manner. The rest of this section introduces the concept of the *Markov Random Fields* followed a detailed discussion of the *ORF* including its feature design, parameter estimation, and predictions.

3.3.1 Markov Random Fields

Markov Random Fields (MRF) [16, 78] models a set of random variables having Markov property with respect to an undirected graph $\mathcal{G}$. The undirected graph $\mathcal{G}$ consists a set of vertices $\mathcal{V}$ connected by a set of edges $\mathcal{E}$ without orientation, where two vertices are neighborhood of each other when connected. Each vertex in $\mathcal{V}$ encodes a random variable, and the Markov property implies that a variable is conditionally independent of other variables given its neighborhoods.

In this work, we use *MRF* to model user preferences and their relations respect to a set of undirected graphs. Specifically for each user u, there is a graph $\mathcal{G}_u$ with a set of vertices $\mathcal{V}_u$ and a set of edges $\mathcal{E}_u$. Each vertex in $\mathcal{V}_u$ represents a preference r_{ui} of user u on item i, and each edge in $\mathcal{E}_u$ captures a relation between two preferences by the same user.

As we consider only the item-item correlations in this work, two preferences are connected by an edge if and only if they are given by the same user. Fig. 3.1 shows an example of two graphs for users u and v. Note that vertices of different graphs are not connected directly, however, the edges between the same pair of items are associated to the same item-item correlation. For example, the edge between r_{ui} and r_{uj} and the edge between r_{vi} and r_{vj} are associated to the same item-item correlation between items i and j (see the green dashed line in Fig. 3.1).

Formally, let $\mathcal{I}(u)$ be the set of all items rated by user u and $\mathbf{r}_u = \{r_{ui} | i \in \mathcal{I}(u)\}$ be the joint set of all preferences (the variables) related to user u, then the *MRF* defines a distribution $P(\mathbf{r}_u)$ over the graph $\mathcal{G}_u$:

$$P(\mathbf{r}_u) = \frac{1}{Z_u} \Psi(\mathbf{r}_u) \tag{3.3.1}$$

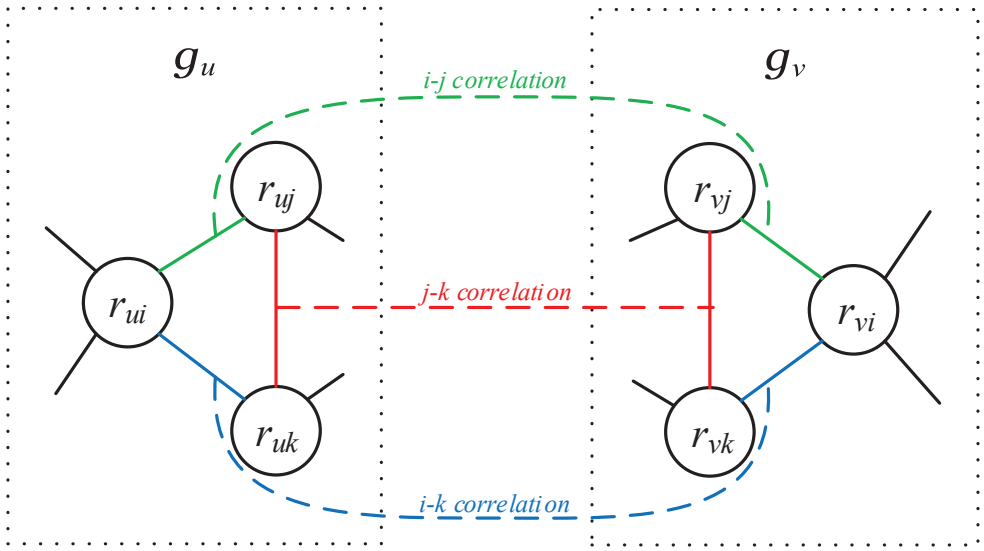


Figure 3.1: Example of undirected graphs for users u and v

$$\Psi(\mathbf{r}_u) = \prod_{(ui,uj)\in\mathcal{E}_u} \psi_{ij}(r_{ui}, r_{uj}) \tag{3.3.2}$$

where Z_u is the normalization term that ensures $\sum_{\mathbf{r}_u} P(\mathbf{r}_u) = 1$, and $\psi(\cdot)$ is a positive function known as *potential*.

The potential $\psi_{ij}(r_{ui}, r_{uj})$ captures the correlation between items i and j

$$\psi_{ij}(r_{ui}, r_{uj}) = \exp\{w_{ij} f_{ij}(r_{ui}, r_{uj})\} \tag{3.3.3}$$

where $f_{ij}(\cdot)$ is the feature function and w_{ij} is the corresponding weight. The correlation features capture the *LS*, while the weights realize the importance of each correlation feature. In *ORF*, the weights also control the relative importance between the *LS* and the *GS*. With the weights estimated from data, the unknown preference r_{ui} can be predicted as

$$\hat{r}_{ui} = \arg\max_{r_{ui}} P(r_{ui}|\mathbf{r}_u) \tag{3.3.4}$$

where $P(r_{ui}|\mathbf{r}_u)$ serves as the confidence measure of the prediction.

3.3.2 ORF: Unifying MRF and OMF

The standard *MRF* approach captures the *LS* by modeling item-item correlations under the framework of probabilistic graphical models. However, it employs the log-linear modeling as shown in Eq. 3.3.3, and therefore does not enable a simple treatment of ordinal preferences. *OMF*, on the other hand, can nicely model the ordinal preferences in a probabilistic way but is weak in capturing the *LS*. The complementary between these two techniques calls for the unified *ORF* model to take all of the advantages.

Essentially, the proposed *ORF* model promotes the agreement between the *GS* discovered by the *OMF* and the *LS* discovered by the *MRF*. More specifically, the *ORF* model combines the item-item correlations (Eq. 3.3.3) and the point-wise ordinal distribution $Q(r_{ui}|u,i)$ obtained from the *OMF* (Eq. 3.2.7)

$$P(\mathbf{r}_u) \propto \Psi_u(\mathbf{r}_u) \prod_{r_{ui} \in \mathbf{r}_u} Q(r_{ui}|u,i) \tag{3.3.5}$$

where $\Psi_u(\mathbf{r}_u)$ is the potential function capturing the interaction among items, and $\mathbf{r}_u$ is the set of preferences from user u.

The potential function $\Psi_u(\mathbf{r}_u)$ can be further factorized into pairwise potentials based on Eq. 3.3.3 and Eq. 3.3.2:

$$\Psi_u(\mathbf{r}_u) = \exp\left(\sum_{r_{ui}, r_{uj} \in \mathbf{r}_u} w_{ij} f_{ij}(r_{ui}, r_{uj}) \right) \tag{3.3.6}$$

where $f_{ij}(\cdot)$ is the correlation feature between items i and j to be defined shortly in Section 3.3.3, and w_{ij} is the corresponding weight controls the relative importance of each correlation feature (LS) to the ordinal distribution (GS). Put all together, the

joint distribution $P(\mathbf{r}_u)$ for each user u can be modelled as

$$P(\mathbf{r}_u) \propto \exp\left(\sum_{r_{ui},r_{uj}\in\mathbf{r}_u} w_{ij} f_{ij}(r_{ui}, r_{uj})\right) \prod_{r_{ui}\in\mathbf{r}_u} Q(r_{ui}|u,i) \tag{3.3.7}$$

where there is a graph for each user but the weights are optimised by all users.

In fact, the user-user correlations can also be captured as

$$P(R) \propto \prod_i \Psi_i(\mathbf{r}_i) \prod_u \Psi_u(\mathbf{r}_u) \prod_{u,i} Q(r_{ui}|u,i) \tag{3.3.8}$$

but we limit our discussion to item-item correlations in this paper.

3.3.3 Feature Design

A feature is essentially a function f of $n > 1$ arguments that maps the (n-dimensional) input onto the unit interval $f : \mathbb{R}^n \to [0, 1]$, where the input can be ratings or auxiliary information such as *content* [78].

The item-item correlation is captured by the following feature

$$f(r_{ui}, r_{uj}) = g(|(r_{ui} - \bar{r}_i) - (r_{uj} - \bar{r}_j)|) \tag{3.3.9}$$

where $g(\alpha) = 1 - \alpha/(L-1)$ does normalization, and $\bar{r}_i$ and $\bar{r}_j$ are the average ratings for items i and j, respectively. This correlation feature captures the intuition that correlated items should receive similar ratings by the same user after offsetting the goodness of each item.

Though this work focuses on the item-item correlations, the feature for user-user correlations can be designed in a similar manner:

$$f(r_{ui}, r_{vi}) = g(|(r_{ui} - \bar{r}_u) - (r_{vi} - \bar{r}_v)|) \tag{3.3.10}$$

where $\bar{r}_u$ and $\bar{r}_v$ are the global average ratings for users u and v respectively.

Although the user and item bias have been modeled by the underlying *OMF*, the *ORF* itself can also model the bias with *identity features* for item i and for user u

$$f_i(r_{ui}, i) = g(|r_{ui} - \bar{r}_i|),\ f_u(r_{ui}, u) = g(|r_{ui} - \bar{r}_u|) \tag{3.3.11}$$

Indeed, auxiliary information such as *content* [5] and *social relations* [50] can also be modeled by designing corresponding features. That being said, the *ORF* is a generic framework with great extensibility to integrate multiple sub-components such as neighborhood, content, and ordinal ratings.

Nevertheless, this work focuses on the item-item correlation features only. Since one correlation feature exists for each possible pair of co-rated items, the number of correlation features can be large, and this makes the estimation slow to converge and less robust. Therefore we only keep the correlation features if strong correlation exists between two items i and j. Specifically, the *strong correlation features* are extracted based on the Pearson correlation and a user-specified *minimum correlation threshold*.

3.3.4 Parameter Estimation

In general, *MRF* models cannot be determined by standard maximum likelihood approaches, instead, approximation techniques such as *Markov Chain Monte Carlo* (MCMC) [26] and *Pseudo-likelihood* [8] are often used in practice. The *pseudo-likelihood* leads to exact computation of the loss function and its gradient with respect to parameters, and thus faster. The MCMC-based methods may, on the other hand, lead to better estimation given enough time. As the experiments involve different settings and large number of features, this study employs the *pseudo-likelihood* technique to perform efficient parameter estimation by maximizing the regularized sum

of log local likelihoods

$$\mathcal{L}(\mathbf{w}) = \sum_{r_{ui} \in R} \log P(r_{ui}|\mathbf{r}_u \backslash r_{ui}) - \frac{1}{2\sigma^2} \sum_{u \in \mathcal{U}} \mathbf{w}_u^T \mathbf{w}_u \tag{3.3.12}$$

where σ is the regularization coefficient, and $\mathbf{w}_u$ is the subset of weights related to user u.

The local likelihood is defined as

$$P(r_{ui}|\mathbf{r}_u \backslash r_{ui}) = \frac{1}{Z_{ui}} Q(r_{ui}|u,i) \exp\left(\sum_{r_{uj} \in \mathbf{r}_u \backslash r_{ui}} w_{ij} f_{ij}(r_{ui}, r_{uj}) \right) \tag{3.3.13}$$

where Z_{ui} is the normalization term.

$$Z_{ui} = \sum_{r_{ui}=1}^{L} Q(r_{ui}|u,i) \exp\left(\sum_{r_{uj} \in \mathbf{r}_u \backslash r_{ui}} w_{ij} f_{ij}(r_{ui}, r_{uj}) \right) \tag{3.3.14}$$

To optimize the parameters, we use the stochastic gradient ascent procedure that updates the parameters by passing through the set of ratings of each user:

$$\mathbf{w}_u \leftarrow \mathbf{w}_u + \eta \nabla \mathcal{L}(\mathbf{w}_u) \tag{3.3.15}$$

where η is the learning rate. More specifically, for each r_{ui} and its neighbor r_{uj} in the set of ratings $\mathbf{r}_u$ by user u, update the weight w_{ij} using the gradient of the log pseudo-likelihood

$$\frac{\partial log \mathcal{L}}{\partial w_{ij}} = f_{ij}(r_{ui}, r_{uj}) - \sum_{r_{ui}=1}^{L} P(r_{ui}|\mathbf{r}_u \backslash r_{ui}) f_{ij}(r_{ui}, r_{uj}) \tag{3.3.16}$$

3.3.5 Preference Prediction

The prediction of rating r_{ui} is straightforward, which can be done by identifying the rating with the greatest mass in local likelihood:

$$\hat{r}_{ui} = \arg\max_{r_{ui}} P(r_{ui}|\mathbf{r}_u) \tag{3.3.17}$$

where the local likelihood is given by Eq. 3.3.13. Prediction made in this approach identifies the most likely rating from discrete values 1 to L, and the local likelihood serves as a *confidence* measure. For predictions of scalar values, the expectation can be used instead:

$$\hat{r}_{ui} = \sum_{r_{ui}=1}^{L} r_{ui} P(r_{ui}|\mathbf{r}_u) \tag{3.3.18}$$

Finally, Alg. 1 summarizes the learning and prediction procedures for the *ORF*.

3.4 Experiment and Analysis

To study the performance of the proposed *ORF* model, comparisons were made with the following representative algorithms: *a*) *K-Nearest Neighbors* (K-NN) [65, 69], which represents the methods exploiting the *LS*; *b*) *OMF* [42], which exploits the *GS* and ordinal properties; *c*) and finally the *ORF* model, which takes ordinal properties into account and exploits both the *LS* and the *GS*. Details of the experimental settings and results are presented in this section.

3.4.1 Experimental Settings

Datasets Experiments were conducted on two public movie rating datasets: the MovieLens-100K and the MovieLens-1M [1] datasets. The MovieLens-1M dataset contains roughly 1 million ratings by 6040 users on 3900 movies. The MovieLens-100K dataset contains 100K ratings by 943 users on 1682 movies. Ratings are on the $1-5$ scale.

To perform a reliable evaluation, we keep only users who rated at least 30

[1] `http://grouplens.org/datasets/movielens`

Algorithm 1 *Ordinal Random Fields* Algorithm

Input: User preferences R; the ordinal distribution Q from Eq. 3.2.7.

Step 1: Generate strong correlation features: $\mathbf{f}_{strong} \leftarrow \{f_{ij} | Pearson(i,j) \geq minCorr\}$

Step 2: Initialize the weights: $\forall w_{ij} \in \mathbf{w}$, $w_{ij} \leftarrow \mathcal{N}(0, 0.01)$;

Step 3: Repeat

for each $u \in \mathcal{U}$ **do**

for each $r_{ui}, r_{uj} \in \mathbf{r}_u$, $i \neq j$ **do**

if $f_{ij} \in \mathbf{f}_{strong}$ **then**

Compute correlation feature f_{ij} according to Eq. 3.3.9

Compute normalization term Z_{ui} according to Eq. 3.3.14

Compute local likelihood according to Eq. 3.3.13

Compute the gradient for weight w_{ij} according to Eq. 3.3.16

Update w_{ij} with the gradient $w_{ij} \leftarrow w_{ij} + \eta \nabla \mathcal{L}(w_{ij})$

end if

end for

end for

Until stopping criteria met

Predictions:

* Predict most likely rating with confidence measure using Eq. 3.3.18.

* Predict expectation using Eq. 3.3.17

movies, and each dataset is shuffled and split into the disjoint training set, validation set and test set. For each user, 5 ratings are kept in the validation set for tuning the hyper-parameters, 10 ratings are reserved for testing, and the rest for training.

Evaluation Metric The *Mean Absolute Error* (MAE) and the *Root Mean Square Error* (RMSE) are used as the evaluation metric

$$MAE = \frac{1}{|R_{test}|} \sum_{(u,i) \in R_{test}} |\hat{r}_{ui} - r_{ui}|, \; RMSE = \sqrt{\frac{\sum_{(u,i) \in R_{test}} (\hat{r}_{ui} - r_{ui})^2}{|R_{test}|}} \tag{3.4.1}$$

where R_{test} is the test set kept aside until all parameters have been tuned. A smaller *MAE* or *RMSE* value indicates better performance. Although both metrics are used, we consider the *MAE* metric to be more suitable for ordinal preferences. The reason is that it makes more scenes to consider being off by 4 is just twice as bad as being off by 2 when the preferences are ordinal. The *RMSE* metric, on the other hand, can be skewed to methods that are optimized for numerical preferences.

Parameters To perform a fair comparison, we fix the number of latent factors to the typical value of 50 for all algorithms, and all weights are randomly initialized from $\mathcal{N}(0, 0.01)$. The number of neighbors for *K-NN* algorithms is set to 10 and 30. The minimum correlation threshold for the *ORF* is set to reasonable values considering both the prediction performance and computational efficiency. We will also report the effect of varying the minimum correlation threshold.

3.4.2 Result and Analysis

We first compare the performance of the proposed *ORF* model with related algorithms: user-based *K-NN*, item-based *K-NN* and *OMF*, where the *OMF* is the targeted baseline. Then the impact of parameters is investigated for the *ORF* model, in particular the regularization coefficient and the minimum correlation threshold.

Comparison with Other Methods

The comparison results in terms of prediction accuracy are shown in Table 3.2. The *global average* is used only as a benchmark, which uses the average rating as the predictions. The following observations can be made based on the results.

Firstly, the *K-NN* methods, especially the item-based *K-NN*, perform quite well. As the *K-NN* methods exploit only the *LS*, this result indicates the effectiveness of the *LS*. However, the ignorance of the *GS* makes the *K-NN* methods not less generalized and thus highly susceptible to noisy data.

Secondly, the *OMF* fits the data quite well when predicting the most likely ratings for the *MAE* metric. However, it exploits only the *GS* and therefore further improvements are possible by incorporating the *LS* information.

Finally, the *ORF* has made further improvements upon the *OMF* by unifying the modeling of both the *LS* and the *GS*, as well as ordinal properties. Note that the performance of the *ORF* relies on the ordinal distributions generated by the underlying *OMF*, which can be implemented in different ways [32, 42, 61, 82]. In present work, the improvements over the *OMF* are soley based on incorporating the *LS* information.

To confirm the improvements, a paired t-test (two-tailed) with a confidence level of

Table 3.2: For both the *OMF* and the *ORF*, the expectation values (Eq. 3.3.18) are used for *RMSE* and the most likely values (Eq. 3.3.17) are used for *MAE*.

	MovieLens-100K		MovieLens-1M	
Method	RMSE	MAE	RMSE	MAE
Global Ave.	1.1186	0.9430	1.1123	0.9401
UserKNN, K=10	0.9687	0.7584	0.9350	0.7328
ItemKNN, K=10	0.9372	0.7305	0.9032	0.7041
UserKNN, K=30	0.9463	0.7413	0.9149	0.7173
ItemKNN, K=30	0.9315	0.7295	**0.8987**	0.70478
OMF	0.9525	0.7226	0.9144	0.6918
ORF, minCorr=0.4	0.9475	0.7185	0.9117	0.6887
ORF, minCorr=0.3	0.9448	**0.7148**	**0.9093**	**0.6870**

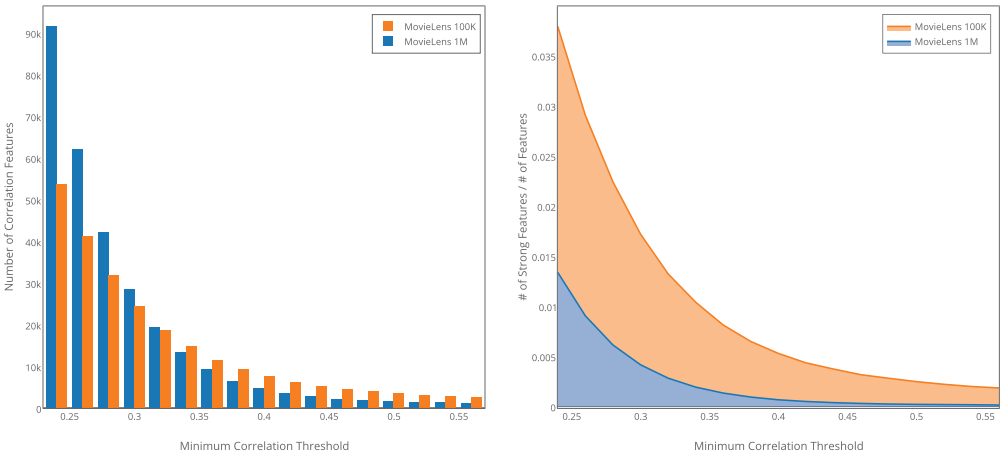
Table 3.3: Paired t-test for the *ORF* and the *OMF*.

	t-test statistics		
Methods	df	t	p-value
ORF vs. OMF on MAE	9	6.0163	0.0002
ORF vs. OMF on RMSE	9	4.8586	0.0009

95% has been applied to the *ORF* and the *OMF*. Results shown in Table 3.3 confirm that the performance of methods with and without capturing the *LS* is statistically significant.

Impact of Minimum Correlation Threshold

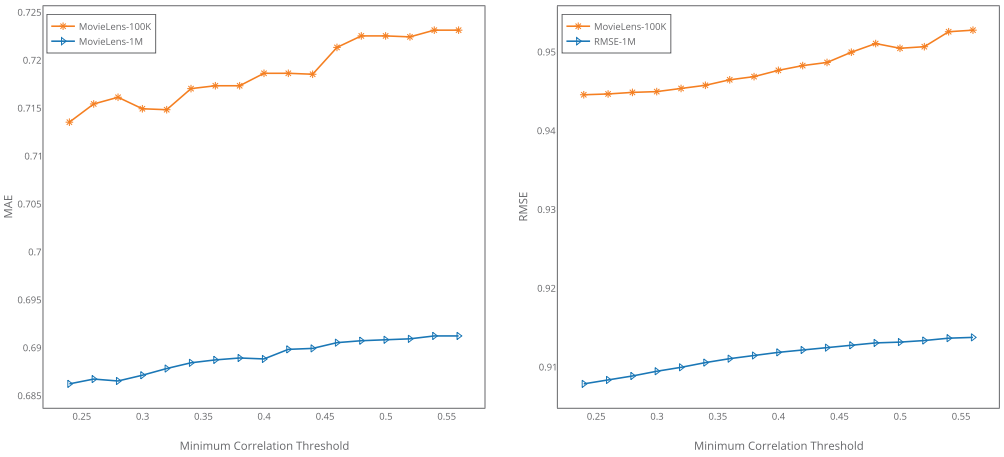
As mentioned in Section 3.3.3, the *ORF* model requires a minimum correlation threshold to control the number of correlation features. The reason is that the number of correlation features can be very large, which makes the model less robust and slow to converge. Specifically, when this threshold goes to minimum (e.g. -1 for Pearson correlation), the potential number of correlation features can be as large as $n^2/2$ where n is the number of items. On the other hand, the number of correlation features goes to zero when the threshold goes to maximum, and the *ORF* reduces to the *OMF*.



(a) Number of Correlation Features (b) Coverage of Correlation Features

Figure 3.2: Impact of Minimum Correlation Threshold on Number of Correlation Features

Fig. 3.2(a) shows the number of correlation features for different minimum correlation thresholds. Given that the MovieLens-100K dataset contains less items comparing the MovieLens-1M dataset, there are even more correlation features remained in the MovieLens-100K dataset when the threshold becomes larger. This observation implies that the items in the MovieLens-100K dataset are more correlated with each other. We also show the coverage of correlation features for both datasets, and the MovieLens-100K has consistently higher coverage of correlation features.



(a) MAE (b) RMSE

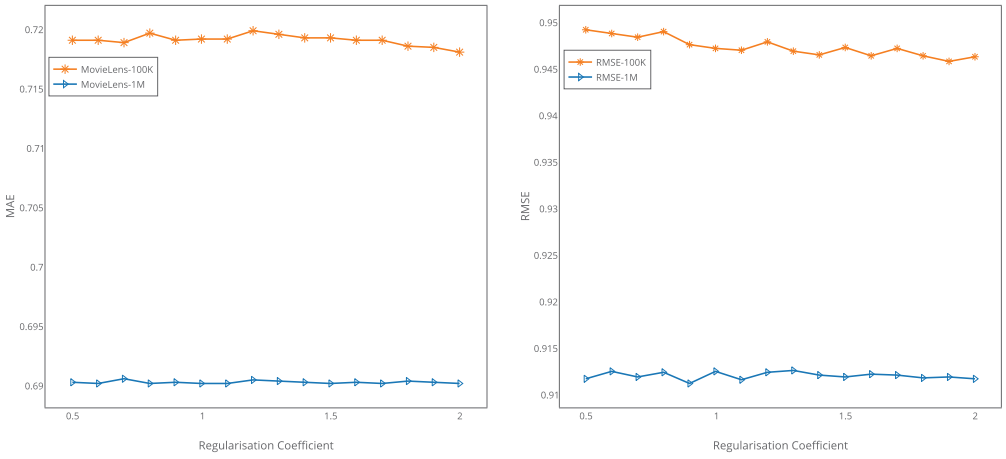
Figure 3.3: Impact of Minimum Correlation Threshold

Having these statistics result, we further examine the impact of the minimum correlation threshold on prediction accuracy, as plotted in Fig. 3.3. It can be observed that the prediction accuracy improves as the minimum correlation threshold becomes smaller. However, we notice that the performance on the smaller MovieLens-100K dataset is not as stable as that on the MovieLens-1M dataset, where the curve of the MovieLens-1M dataset is smoother and shows better monotonicity. One explanation

is that the MovieLens-100K dataset may not have enough data to make robust estimation for large number of weights. However, given adequate data and time, the best prediction performance can be achieved by including all correlation features, i.e., the minimum correlation threshold is set to minimum.

Impact of Regularization Coefficient

While the number of correlation features can be large, the model might be prone to over-fitting. Therefore we investigate the impact of varying the the regularization coefficient. Fig. 3.4 shows the performance of the *ORF* under different regularization



(a) MAE (b) RMSE

Figure 3.4: Impact of Regularization Coefficient

settings. We observe that by varying the regularization coefficient the prediction performance was not affected too much. One possible explanation is that the ordinal distribution employed in the *ORF* is generated by the underlying *OMF* with its own regularization mechanism, whereas the regularization term in the *ORF* controls only

the weights for the second-order item-item correlation features. In other words, the *ORF* model by itself is unlikely to over-fit the data given that the underlying *OMF* model has been properly regularized.

3.5 Summary

In this work we presented the *ORF* model that takes advantages of both the representational power of the *MRF* and the ease of modeling ordinal preferences by the *OMF*. While the standard *OMF* approaches exploit only the *GS*, the *ORF* model is able to capture the *LS* as well. In addition, the *ORF* model defines a uniformed interface for different *OMF* approaches with various internal implementations. Last but not least, the *ORF* model is a generic framework that can be extended to incorporate additional information by designing more features.

A future extension could take the user-user correlations into account as we modeled only the item-item correlations in this work. Incorporating the user-user correlations may further improve the prediction performance. Another future work is to take *auxiliary information* into consideration by replacing the *MRF* with the *Conditional Random Fields* [43]. Fusing *auxiliary information* such as the *item content* and *social relations* could improve the prediction performance especially when the data is highly sparse.

Chapter 4

Preferecen Relation-based Recommender System

4.1 Introduction

RecSys aim to recommend users with some of their potentially interesting items, which can be virtually anything ranging from *movies* to *tourism attractions*. To identify the appropriate items, RecSys attempts to exploit *user preferences* [41] and various *side information* including *content* [5,6], *temporal dynamics* [40], and *social relations* [50]. By far, *Collaborative Filtering* [41] is one of the most popular RecSys techniques, which exploits user preferences, especially in form of explicit *absolute ratings*. Nevertheless, relying on solely *absolute ratings* is prone to the *cold-start* problem [72] where few ratings are known for *cold* users or items. To alleviate the *cold-start* problem, additional information, which is usually heterogeneous [6] and implicit [64] must be acquired and exploited.

Recently, a considerable literature [12, 19, 45, 64, 74] has grown up around the theme of *relative* preferences. The underlying motivation is that *relative* preferences are often easier to collect and more reliable as a measure of user preferences. For

example, it can be easier for users to tell which item is preferable than expressing the precise degree of liking. Furthermore, studies [12, 42] have reported that absolute ratings may not be completely trustworthy. For example, rating 4 out of 5 may in general indicate high quality, but it can mean just OK for critics. In fact, users' quantitative judgment can be affected by irrelevant factors such as the *mood* when rating, and this is called misattribution of memory [71].

While users are not good at making consistent quantitative judgment, the *preference relation* (PR), as a kind of relative preference, has been considered as more consistent across like-minded users [12, 18, 19]. By measuring the relative order between items, the *PR* is usually less variant to irrelevant factors. For example, a user in bad *mood* may give lower ratings to all items but the relative orderings between items remain the same. Being a more reliable type of user preferences, *PR* is also easier to collect comparing to ratings as it can be inferred from implicit feedbacks. For example, the *PR* between two items can be inferred by comparing their *ratings*, *page views*, *played counts*, *mouse clicks*, etc. This property is important as not all users are willing to rate their preferences, where collecting feedbacks implicitly delivers a more user-friendly recommender system. In addition, as the ultimate goal of *RecSys*, obtaining the ranking of items by itself is to obtain the relative preferences, a more natural input than absolute ratings [42, 81].

While the *PR* captures the user preferences in the *pairwise* form, most existing works [42, 46] take the *pointwise* approach to exploiting ordinal properties possessed by absolute ratings. To accept the *PR* as input and output item rankings, *pairwise* approaches to RecSys have recently emerged in two forms: memory-based [12] and model-based [19, 45, 64]. These studies have shown the feasibility of *PR*-based

methods, and demonstrated competitive performance comparing to their underlying models, such as memory-based *K-Nearest Neighbor* (KNN) [12] and model-based *Matrix Factorization* (MF) [19].

However, the limitations of these underlying models have constrained the potentials of their *PR* extensions. More specifically, both *KNN* and *MF* based methods can only capture one type of information at a time, while both the *local* and the *global* information are essential in achieving good performance [38, 46, 79]. We refer these two types of information as the *local* and the *global* structures of the preferences:

Local Structure The *local structure* (LS) refers to the second-order interactions between similar users [65] or items [69]. This type of information is often used by *neighborhood-based* collaborative filtering, in which the predictions are made by looking at the neighborhood of users [65] or items [69]. *LS*-based approaches ignore the majority of preferences in making predictions, but are effective when the users or items correlations are highly localized.

Global Structure The *global structure* (GS) refers to the weaker but higher-order interactions among all users and items [41]. This type of information is often used by *latent factor models* such as *Matrix Factorization* [41], which aim to discover the latent factor space in the preferences. *GS*-based approaches are often competitive in terms of accuracy and computational efficiency [41].

Previous studies have suggested that these two structures are complementary since they address different aspects of the preferences [38, 46, 79]. However, to the best of our knowledge, there is yet no *PR*-based method that can capture both *LS* and *GS*. Another problem of existing *PR*-based methods is that side information such as item content and user attributes can't be easily incorporated, which is critical in

cold-start cases. All the above reasonings lead to the desired model with the following properties: *1*) Accept *PR* as input; *2*) Capture both *LS* and *GS*; *3*) Side information can be easily incorporated; *4*) Output item rankings.

Recent advances in *Markov Random Fields*-based RecSys [16,46,77,79] have made it possible to achieve the above objectives. *MRF*-based RecSys was first developed in [79] to capture both *LS* and *GS*. Later on, it has been extended in [46] to exploit ordinal properties possessed by absolute ratings. Nevertheless, all of these attempts rely on absolute ratings.

This work aims to push the *MRF*-based RecSys one step further by fitting it into the *PR* framework, namely the *Preference Relation-based Markov Random Fields* (PrefMRF) and the *Preference Relation-based Conditional Random Fields* (PrefCRF) when side information is incorporated. The remaining part of this paper is organized as follows. Section 4.2 introduces the concepts of *PR*-based RecSys and formalizes the problem, followed by a review of related work. Section 4.3 is devoted to the proposed *PrefMRF* and *PrefCRF* models. Benchmark results on Top-N recommendation are presented in Section 4.4. Finally, Section 4.5 concludes this work by summarizing the main contributions and envisaging future works.

4.2 Preliminaries

RecSys aim at predicting users' future interest in items, and the recommendation task can be considered as a *preference learning* problem, which aims to construct a predictive preference model from observed preference information [58]. Existing *preference learning* methods are based on different *learning to rank* approaches [23].

Among them, the *pointwise* approach is the choice of most RecSys [38, 69], which exploit absolute ratings, though *pairwise* approach that exploits *PR* has been largely overlooked until recently. The rest of this section describes the basic concepts and formalizes the *PR*-based RecSys followed by a review of related work.

4.2.1 Preference Relation

A *preference relation* (PR) encodes user preferences in form of pairwise ordering between items. This representation is a useful alternative to absolute ratings for three reasons.

Firstly, *PR* is more consistent across like-minded users [12, 19] as it is invariant to many irrelevant factors, such as *mood*. Secondly, *PR* is a more natural and direct input for Top-N recommendation, as both the input and the output are relative preferences. Finally, and perhaps most importantly, *PR* can be obtained implicitly rather than asking the users explicitly. For example, the *PR* over two *Web pages* can be inferred by the *stayed time*, and consequently applies to the displayed items. This property is important as not all users are willing to rate their preferences, where collecting feedbacks implicitly delivers a more user-friendly RecSys. In addition, *PR*-based RecSys provides an opportunity to utilize the vast amount of implicit data that have already been collected over the years, such as *activity logs*. With these potential benefits, we shall take a closer look at the *PR*, and investigate how they can be utilized in RecSys.

We formally define the *PR* as follows. Let $\mathcal{U} = \{u\}^n$ and $\mathcal{I} = \{i\}^m$ denote the set of n users and m items, respectively. The preference of a user $u \in \mathcal{U}$ between items i and j is encoded as π_{uij}, which indicates the strength of user u's *PR* for the ordered

item pair (i, j). A higher value of π_{uij} indicates a stronger preference on the first item over the second item.

Definition 5 (Preference Relation). The preference relation is defined as

$$\pi_{uij} = \begin{cases} (\frac{2}{3}, 1] & \text{if } i \succ j \ (u \text{ prefers } i \text{ over } j) \\ [\frac{1}{3}, \frac{2}{3}] & \text{if } i \simeq j \ (i \text{ and } j \text{ are equally preferable to } u) \\ [0, \frac{1}{3}) & \text{if } i \prec j \ (u \text{ prefers } j \text{ over } i) \end{cases} \tag{4.2.1}$$

where $\pi_{uij} \in [0, 1]$ and $\pi_{uij} = 1 - \pi_{uji}$.

This definition is similar to [19], however, we allocate an interval for each preference category, i.e., *preferred*, *equally preferred*, and *less preferred*. Indeed, each preference category can be further break down into more intervals.

Similar to [12], the *PR* can be converted into user-wise preferences over items.

Definition 6 (User-wise Preference). The user-wise preference is defined as

$$p_{ui} = \frac{\sum_{j \in \mathcal{I}_u} [\![\pi_{uij} > \frac{2}{3}]\!] - \sum_{j \in \mathcal{I}_u} [\![\pi_{uij} < \frac{1}{3}]\!]}{|\Pi_{ui}|} \tag{4.2.2}$$

where $[\![\cdot]\!]$ gives 1 for *true* and 0 for *false*, and Π_{ui} is the set of user u's PR related to item i.

The user-wise preference p_{ui} falls in the interval $[-1, 1]$, where -1 and 1 indicate that item i is the least or the most preferred item for user u, respectively. The user-wise preference measures the relative position of an item for a particular user, which is different from absolute ratings.

4.2.2 Problem Statement

Generally, the task of *PR*-based RecSys is to take *PR* as input and output Top-N recommendations. Specifically, let $\pi_{uij} \in \Pi$ encode the *PR* of each user $u \in \mathcal{U}$.

Each π_{uij} is defined over an ordered item pair $(i,\ j)$, denoting $i \prec j$, $i \simeq j$, or $i \succ j$ as described in Eq. 4.2.1. The goal is to estimate the value of each unknown $\pi_{uij} \in \Pi_{unknown}$, such that $\hat{\pi}_{uij}$ approximates π_{uij}. This can be considered as an optimization task performs directly on the *PR*:

$$\hat{\pi}_{uij} = \underset{\hat{\pi}_{uij} \in [0,1]}{\arg\min} (\pi_{uij} - \hat{\pi}_{uij})^2 \tag{4.2.3}$$

However, it can be easier to estimate the $\hat{\pi}_{uij}$ by the difference between the two user-wise preferences p_{ui} and p_{uj}, i.e., $\hat{\pi}_{uij} = \phi(\hat{p}_{ui} - \hat{p}_{uj})$, where $\phi(\cdot)$ is a function that bounds the value into $[0, 1]$ and ensures $\phi(0) = 0.5$. For example, the *inverse-logit* function $\phi(x) = \frac{e^x}{1+e^x}$ can be used when user-wise preferences involve large values. Therefore, the objective of this paper is to solve the following optimization problem:

$$(\hat{p}_{ui}, \hat{p}_{uj}) = \underset{\hat{p}_{ui}, \hat{p}_{uj}}{\arg\min} (\pi_{uij} - \phi(\hat{p}_{ui} - \hat{p}_{uj}))^2 \tag{4.2.4}$$

which optimizes the user-wise preferences directly, and Top-N recommendations can be obtained by simply sorting the estimated user-wise preferences.

For ease of reference, notations used throughout this chapter are summarized in Table 4.1. The letters u, v, a, b represent *users*, and the letters i, j, k, l represent *items*.

4.2.3 Related Work

User preferences can be modeled in three types: *pointwise*, *pairwise*, and *listwise*. Though RecSys is not limited to the *pointwise* absolute ratings, the recommendation task is usually considered as a rating prediction problem. Recently, a considerable literature [12, 17, 19, 24, 36, 45, 64, 74] has grown up around the theme of *relative* preferences, especially the *pairwise PR*. Meanwhile, recommendation task is also shifting

Table 4.1: Summary of Major Notations (Chapter 4)

Notations	Mathematical Meanings
$\mathcal{U}$	the set of users
$\mathcal{I}$	the set of items
Π	the set of preference relations
p_{ui}	the user-wise preference of user u on item i
$\mathcal{G}$	an undirected graph encodes relations of user-wise preferences
$\mathcal{V}$	the set of vertices each represents a user-wise preference
$\mathcal{E}$	the set of edges each connects two vertices
f_{uv}	the correlation feature between users u and v
f_{ij}	the correlation feature between items i and j
w_{uv}	the weight associated to the user-user correlation feature f_{uv}
w_{ij}	the weight associated to the item-item correlation feature f_{ij}
$Q(p_{ui} \mid u, i)$	the ordinal distribution produced by PrefNMF
$\mathbf{o}$	the side information, e.g., user attributes and item content

from rating prediction to item ranking [56, 74, 83], in which the ranking itself is also *relative* preferences. Preference relation has been widely studied in the field of

The use of *relative* preferences has been widely studied in the field of *Information Retrieval* [14, 21, 22, 35, 64] for *learning to rank* tasks. Recently, *PR*-based [12, 19, 45, 64] and *listwise*-based [74] RecSys have been proposed. Among them, the *PR*-based approach is the most popular, which can be further categorized as *memory-based* methods [12] that capture *local structure* and *model-based* methods [19, 45, 64] that capture *global structure*. To the best of our knowledge, there is yet no *PR*-based method that can capture both *LS* and *GS*.

Advances in *Markov Random Fields* (MRF) and its extension *Conditional Random Fields* (CRF) have made it possible to utilize both *LS* and *GS* by taking advantages of MRF's powerful representation capability. Nevertheless, exploiting the *PR* is not an easy task for *MRF* and *CRF* [46, 79]. This observation leads to a natural extension of unifying the MRF models with the *PR*-based models, to complement their strengths. We summarize the capabilities of the existing and our proposed *PrefMRF* and *PrefCRF* models in Table 4.2.

4.3 Methodology

In this section, we propose the *Preference Relation-based Markov Random Fields* (PrefMRF) to model the *PR* and capture both *LS* and *GS*. When side information is taken into consideration, this model extends to *Preference Relation-based Conditional Random Fields* (PrefCRF). In this work, we exploit *LS* in terms of the item-item correlations as well as the user-user correlations. The rest of this section introduces

Table 4.2: Capabilities of PR-based methods

Method	Input	Output	LS	GS	Side Information
Pointwise Memory-based	Ratings	Ratings	✓		
Pointwise Model-based	Ratings	Ratings		✓	
Pointwise Hybrid	Ratings	Ratings	✓	✓	
Pairwise Memory-based	Preference Relations	Item Rankings	✓		
Pairwise Model-based	Preference Relations	Item Rankings		✓	
PrefMRF	**Preference Relations**	**Item Rankings**	✔	✔	
PrefCRF	**Preference Relations**	**Item Rankings**	✔	✔	✔

the concept of the *Preference Relation-based Matrix Factorization*(PrefNMF) [19] that will be our underlying model, and then followed a detailed discussion of the *PrefMRF* and *PrefCRF* on issues including feature design, parameter estimation, and predictions.

4.3.1 Preference Relation-based Matrix Factorization

Matrix Factorization (MF) [41] is a popular approach to *RecSys* that has mainly been applied to absolute ratings. Recently, the *PrefNMF* [19] model was proposed to adopt *PR* input for *MF* models. Like *MF* models, the *PrefNMF* model discovers the latent factor space shared between users and items, where the latent factors describe both the *taste* of users and the *characteristics* of items. The attractiveness of an item to a user is then measured by the inner product of their latent feature vectors.

Point Estimation

The *PrefNMF* model described in [19] provides a point estimation to each preference, i.e., a single value. Formally, each user u is associated with a latent feature vector $\mathbf{u}_u \in \mathbb{R}^k$ and each item i is associated with a latent feature vector $\mathbf{v}_i \in \mathbb{R}^k$, where k is the dimension of the latent factor space. The attractiveness of items i and j to the user u are $\mathbf{u}_u^\top \mathbf{v}_i$ and $\mathbf{u}_u^\top \mathbf{v}_j$, respectively. When $\mathbf{u}_u^\top \mathbf{v}_i > \mathbf{u}_u^\top \mathbf{v}_j$ the item i is said to be more preferable to the user u than the item j, i.e., $i \succ j$. The strength of this preference relation π_{uij} can be estimated by $\mathbf{u}_u^\top(\mathbf{v}_i - \mathbf{v}_j)$, and the *inverse-logit* function is applied to ensure $\hat{\pi}_{uij} \in [0, 1]$:

$$\hat{\pi}_{uij} = \frac{e^{\mathbf{u}_u^\top(\mathbf{v}_i - \mathbf{v}_j)}}{1 + e^{\mathbf{u}_u^\top(\mathbf{v}_i - \mathbf{v}_j)}} \tag{4.3.1}$$

The latent feature vectors $\mathbf{u}_u$ and $\mathbf{v}_i$ are learned by minimizing regularized squared error with respect to the set of all known preference relations Π:

$$\min_{\mathbf{u}_u, \mathbf{v}_i \in \mathbb{R}^k} \sum_{\pi_{uij} \in \Pi \wedge (i<j)} (\pi_{uij} - \hat{\pi}_{uij})^2 + \lambda(\|\mathbf{u}_u\|^2 + \|\mathbf{v}_i\|^2) \tag{4.3.2}$$

where λ is the regularization coefficient. The optimization can be done with *Stochastic Gradient Descent* for the favor of speed on sparse data, or with *Alternating Least Squares* for the favor of parallelization on dense data.

Distribution Estimation

The original *PrefNMF* [19] computes the attractiveness of an item to a user by the product of their latent feature vectors which results a scalar value, where the likelihoods of other possible values remain unknown. However, in order to be combined with *MRF* models, we wish to have the distributions over a set of possible values.

Therefore the *Random Utility Models* [55] and the *Ordinal Logistic Regression* [54] are applied to perform the conversion.

Random Utility Models [55] assume the existence of a *latent utility* $x_{ui} = \mu_{ui} + \epsilon_{ui}$ that captures how much the user u is interested in the item i, where μ_{ui} captures the interest and ϵ_{ui} is the random noise that follows the logistic distribution as in [42]. The latent utility x_{ui} is then generated from a logistic distribution centered at μ_{ui} with the scale parameter s_{ui} proportional to the standard deviation:

$$x_{ui} \sim Logi(\mu_{ui}, s_{ui}) \tag{4.3.3}$$

Recall that *PrefNMF* computes attractiveness of item to user by the product of their latent feature vectors, thereby the internal score μ_{ui} can be substituted with the term $\mathbf{u}_u^\top \mathbf{v}_i$:

$$x_{ui} = \mathbf{u}_u^\top \mathbf{v}_i + \epsilon_{ui} \tag{4.3.4}$$

where $\mathbf{u}_u$ and $\mathbf{v}_i$ are, respectively, the latent feature vectors of the user u and the item i.

The *Ordinal Logistic Regression* [54] is then used to convert the user-wise preferences p_{ui} into ordinal values, which assumes that the preference p_{ui} is chosen based on the interval to which the latent utility belongs:

$$p_{ui} = l \text{ if } x_{ui} \in (\theta_{l-1}, \theta_l] \text{ for } l < L \text{ and } p_{ui} = L \text{ if } x_{ui} > \theta_{L-1} \tag{4.3.5}$$

where L is the number of ordinal levels and θ_l are the threshold values of interest. The probability of receiving a preference l is therefore

$$Q(p_{ui} = l \mid u, i) = \int_{\theta_{l-1}}^{\theta_l} P(x_{ui} \mid \theta)\, \mathrm{d}\theta = F(\theta_l) - F(\theta_{l-1}) \tag{4.3.6}$$

where $F(\theta_l)$ is the cumulative logistic distribution evaluated at θ_l with standard deviation s_{ui}:

$$F(x_{ui} \leq l \mid \theta_l) = \frac{1}{1 + \exp(-\frac{\theta_{uil} - \mu_{ui}}{s_{ui}})} \tag{4.3.7}$$

The thresholds θ_l can be parameterized to depend on user or item. This paper employs the user-specific thresholds parameterization described in [42]. Therefore a set of thresholds $\{\theta_{ul}\}_{l=1}^{L}$ is defined for each user u to replace the thresholds θ_{uil} in Eq. 4.3.7, and is learned from data.

Given the learned ordinal distribution $Q(p_{ui} \mid u, i)$, not only the preferences can be predicted but also the *confidence* for each prediction. The ordinal distribution $Q(p_{ui} \mid u, i)$ captures the *GS* information in a probabilistic form, and will be incorporated into *MRF* and *CRF* in Section 4.3.4. Note that the user preference is quantized into ordinal values in this process.

4.3.2 Markov Random Fields

Markov Random Fields (MRF) [16,78] model a set of random variables having Markov property with respect to an undirected graph $\mathcal{G}$. The undirected graph $\mathcal{G}$ consists a set of vertices $\mathcal{V}$ connected by a set of edges $\mathcal{E}$ without orientation, where two vertices are neighborhood of each other when connected. Each vertex in $\mathcal{V}$ encodes a random variable, and the Markov property implies that a variable is conditionally independent of others given its neighborhoods.

In this work, we use *MRF* to model user-wise preference and their interactions respect to a set of undirected graphs. Specifically for each user u, there is a graph $\mathcal{G}_u$ with a set of vertices $\mathcal{V}_u$ and a set of edges $\mathcal{E}_u$. Each vertex in $\mathcal{V}_u$ represents a preference p_{ui} of user u on the item i. Note that the term *preference* is used instead

of *rating* because in the new model the *preference* is not interpolated as absolute ratings but user-wise ordering of items. Each edge in $\mathcal{E}_u$ captures a relation between two preferences by the same user.

Two preferences are connected by an edge if they are given by the same user or on the same item, corresponding to the item-item and user-user correlations, respectively. Modeling these correlations is actually capturing the *LS* information in the preferences. However, it is not easy to model two types of correlations at the same time as it will result a large graph. Instead, we model the item-item and user-user correlations separately, and merge their predictions. Fig. 4.1 shows an example of four graphs for user u, user v, item i, and item j. Note that vertices of different graphs are not connected directly, however, the weights are estimated across graphs when the edges correspond to the same correlation. For example, the edge between p_{ui} and p_{uj} and the edge between p_{vi} and p_{vj} are associated to the same item-item correlation ψ_{ij} between items i and j.

Formally, let $\mathcal{I}_u$ be the set of all items evaluated by the user u and $\mathcal{U}_i$ be the set of all users rated the item i. Then denote $\mathbf{p}_u = \{p_{ui} \mid i \in \mathcal{I}_u\}$ be the joint set of all preferences (the variables) expressed by user u, and $\mathbf{p}_i = \{p_{ui} \mid u \in \mathcal{U}_i\}$ be the joint set of all preferences (the variables) rated on item i. Under this setting, the *MRF* defines two distributions $P(\mathbf{p}_u)$ and $P(\mathbf{p}_i)$ over the graphs $\mathcal{G}_u$ and $\mathcal{G}_i$, respectively:

$$P(\mathbf{p}_u) = \frac{1}{Z_u}\Psi_u(\mathbf{p}_u) \ , \ P(\mathbf{p}_i) = \frac{1}{Z_i}\Psi_i(\mathbf{p}_i) \tag{4.3.8}$$

$$\Psi_u(\mathbf{p}_u) = \prod_{(ui,uj)\in\mathcal{E}_u} \psi_{ij}(p_{ui}, p_{uj}) \ , \ \Psi_i(\mathbf{p}_i) = \prod_{(ui,vi)\in\mathcal{E}_i} \psi_{uv}(p_{ui}, p_{vi}) \tag{4.3.9}$$

where Z_u and Z_i are the normalization terms that ensure $\sum_{\mathbf{p}_u} P(\mathbf{p}_u) = 1$ and $\sum_{\mathbf{p}_i} P(\mathbf{p}_i) = 1$. The term $\psi(\cdot)$ is a positive function known as *potential*.

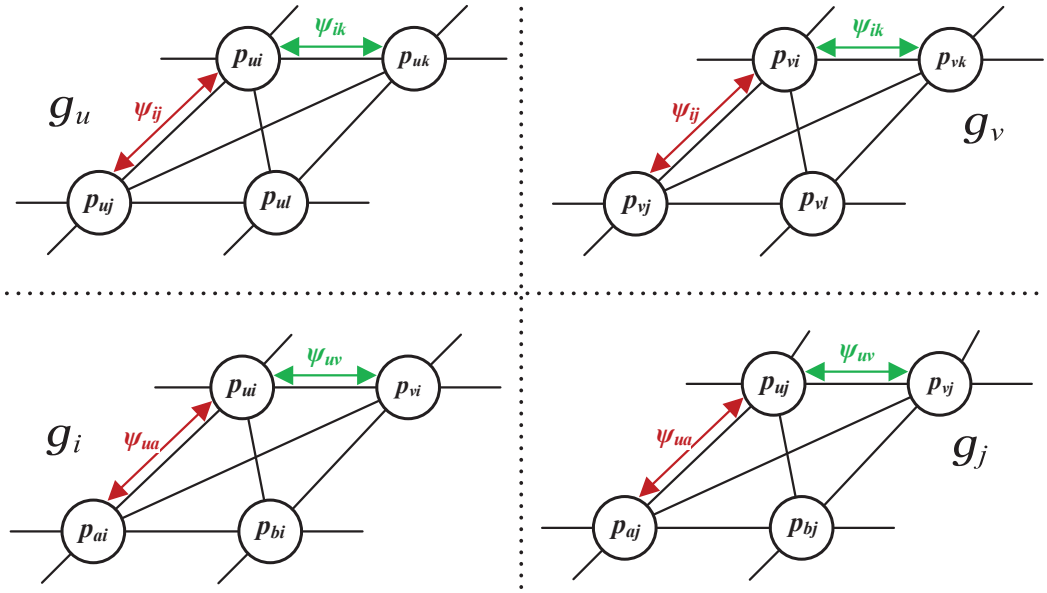


Figure 4.1: Example of MRF graphs. u, v, a, and b are users. i, j, l, and k are items.

The potentials $\psi_{ij}(p_{ui}, p_{uj})$ and $\psi_{uv}(p_{ui}, p_{vi})$ capture the correlation between items i and j and correlation between users u and v, respectively:

$$\psi_{ij}(p_{ui}, p_{uj}) = \exp\{w_{ij} f_{ij}(p_{ui}, p_{uj})\} \tag{4.3.10}$$

$$\psi_{uv}(p_{ui}, p_{vi}) = \exp\{w_{uv} f_{uv}(p_{ui}, p_{vi})\} \tag{4.3.11}$$

where $f_{ij}(\cdot)$ and $f_{uv}(\cdot)$ are the feature functions to be designed shortly in Section 4.3.4, and w_{ij} and w_{uv} are the corresponding weights.

These correlation features capture the *LS* information, while the weights realize the importance of each correlation feature. With the weights estimated from data, the unknown preference p_{ui} can be predicted using item-item correlations:

$$\hat{p}_{ui} = \underset{p_{ui} \in [-1,1]}{\arg\max}\, P(p_{ui} \mid \mathbf{p}_u) \tag{4.3.12}$$

or using user-user correlations:

$$\hat{p}_{ui} = \arg\max_{p_{ui} \in [-1,1]} P(p_{ui} \mid \mathbf{p}_i) \tag{4.3.13}$$

where the confidences of the predictions can be measured by $P(p_{ui} \mid \mathbf{p}_u)$ and $P(p_{ui} \mid \mathbf{p}_i)$.

4.3.3 Conditional Random Fields

Despite of the user preferences, various side information including *content* [5, 6], *temporal dynamics* [40], and *social relations* [50] are also important in making quality recommendations. While there exist methods to incorporate side information, there is yet no *PR*-based method that can achieve this.

One advantage of *Markov Random Fields* is its extensibility, thus side information can be easily incorporated by extending the *MRF* to *Conditional Random Fields* (CRF). In *MRF*, the item-item and user-user correlations are modeled in a set of graphs, where each graph has a set of vertices representing the preferences. To incorporate side information, the *MRF* is extended to *CRF* by conditioning each vertex on a set of global observations $\mathbf{o}$, i.e., the side information in our context. Specifically, each user u is associated with a set of attributes $\{\mathbf{o}_u\}$ such as *gender* and *occupation*. Similarly, each item i is associated with a set of attributes $\{\mathbf{o}_i\}$ such as *genres* of movie. This side information is encoded as the set of global observations $\mathbf{o} = \{\{\mathbf{o}_u\}, \{\mathbf{o}_i\}\}$. The graphs for item-item and user-user correlations conditioned on global observations are illustrated in Fig. 4.2.

Using the same settings as *MRF* in Section 4.3.2, the *CRF* models the conditional

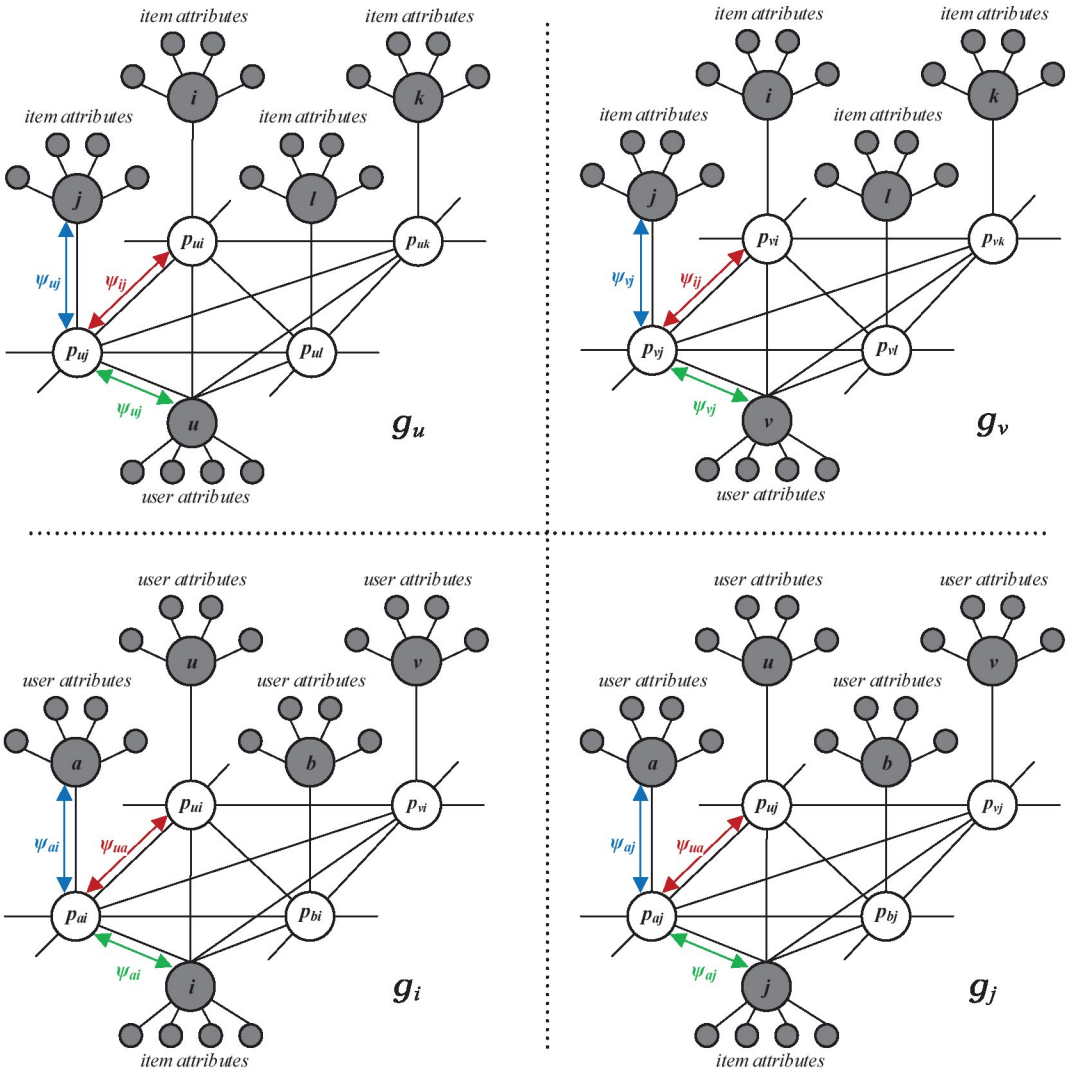


Figure 4.2: Example of CRF graphs. u, v, a, and b are users. i, j, l, and k are items.

distributions $P(\mathbf{p}_u \mid \mathbf{o})$ and $P(\mathbf{p}_i \mid \mathbf{o})$ over the graphs $\mathcal{G}_u$ and $\mathcal{G}_i$, respectively:

$$P(\mathbf{p}_u \mid \mathbf{o}) = \frac{1}{Z_u(\mathbf{o})}\Psi_u(\mathbf{p}_u, \mathbf{o}) \ , \ P(\mathbf{p}_i \mid \mathbf{o}) = \frac{1}{Z_i(\mathbf{o})}\Psi_i(\mathbf{p}_i, \mathbf{o}) \tag{4.3.14}$$

$$\Psi_u(\mathbf{p}_u, \mathbf{o}) = \prod_{(ui)\in\mathcal{V}_u} \psi_{ui}(p_{ui}, \mathbf{o}) \prod_{(ui,uj)\in\mathcal{E}_u} \psi_{ij}(p_{ui}, p_{uj}) \tag{4.3.15}$$

$$\Psi_i(\mathbf{p}_i, \mathbf{o}) = \prod_{(ui)\in\mathcal{V}_i} \psi_{ui}(p_{ui}, \mathbf{o}) \prod_{(ui,vi)\in\mathcal{E}_i} \psi_{uv}(p_{ui}, p_{vi}) \tag{4.3.16}$$

where $Z_u(\mathbf{o})$ and $Z_i(\mathbf{o})$ are the normalization terms ensure $\sum_{\mathbf{p}_u} P(\mathbf{p}_u \mid \mathbf{o}) = 1$ and $\sum_{\mathbf{p}_i} P(\mathbf{p}_i \mid \mathbf{o}) = 1$. The term $\psi(\cdot)$ is a positive function known as *potential*.

The potential $\psi_{ui}(\cdot)$ captures the global observations associated to the user u and the item i:

$$\psi_{ui}(p_{ui}, \mathbf{o}) = \exp\{\mathbf{w}_u^\top \mathbf{f}_u(p_{ui}, \mathbf{o}_i) + \mathbf{w}_i^\top \mathbf{f}_i(p_{ui}, \mathbf{o}_u))\} \tag{4.3.17}$$

The potentials $\psi_{ij}(\cdot)$ and $\psi_{uv}(\cdot)$ capture the item-item and user-user correlations, respectively:

$$\psi_{ij}(p_{ui}, p_{uj}) = \exp\{w_{ij} f_{ij}(p_{ui}, p_{uj})\} \tag{4.3.18}$$

$$\psi_{uv}(p_{ui}, p_{vi}) = \exp\{w_{uv} f_{uv}(p_{ui}, p_{ui})\} \tag{4.3.19}$$

where $\mathbf{f}_u$, $\mathbf{f}_i$, f_{ij}, and f_{uv} are the features to be designed shortly in Section 4.3.4, and $\mathbf{w}_u$, $\mathbf{w}_i$, w_{ij}, and w_{uv} are the corresponding weights realize the importance of each feature. With the weights estimated from data, the unknown preference p_{ui} can be predicted using item-item correlations:

$$\hat{p}_{ui} = \arg\max_{p_{ui}\in[-1,1]} P(p_{ui} \mid \mathbf{p}_u, \mathbf{o}) \tag{4.3.20}$$

or using user-user correlations:

$$\hat{p}_{ui} = \arg\max_{p_{ui}\in[-1,1]} P(p_{ui} \mid \mathbf{p}_i, \mathbf{o}) \tag{4.3.21}$$

where $P(p_{ui} \mid \mathbf{p}_u, \mathbf{o})$ and $P(p_{ui} \mid \mathbf{p}_i, \mathbf{o})$ give the confidence of the predictions.

4.3.4 PrefCRF: Unifying PrefNMF and CRF

The *CRF* model captures the *LS* information by modeling item-item and user-user correlations under the framework of probabilistic graphical models. However, it employs the log-linear modeling as shown in Eq. 4.3.18 and Eq. 4.3.18, and therefore does not enable a simple treatment of *PR*. The *PrefNMF* model, on the other hand, can nicely model the *PR* but is weak in capturing the *LS* and side information. The complementary between these two techniques calls for the unified *PrefCRF* model to take all of the advantages.

Essentially, the proposed *PrefCRF* model promotes the agreement between the *GS* discovered by the *PrefNMF*, the *LS* discovered by the *MRF*, and the side information discovered by the *CRF*. More specifically, the *PrefCRF* model combines the item-item and user-user correlations (Eq. 4.3.15 and Eq. 4.3.16) and the ordinal distributions $Q(p_{ui} \mid u, i)$ over user-wise preferences obtained from Eq. 4.3.6:

$$P(\mathbf{p}_u \mid \mathbf{o}) \propto \Psi_u(\mathbf{p}_u, \mathbf{o}) \prod_{p_{ui} \in \mathbf{p}_u} Q(p_{ui} \mid u, i) \tag{4.3.22}$$

$$P(\mathbf{p}_i \mid \mathbf{o}) \propto \Psi_i(\mathbf{p}_i, \mathbf{o}) \prod_{p_{ui} \in \mathbf{p}_i} Q(p_{ui} \mid u, i) \tag{4.3.23}$$

where Ψ_u is the potential function capturing the interaction among items evaluated by user u, and Ψ_i is the potential function capturing the interaction among users rated item i. Put all together, the joint distribution $P(\mathbf{p}_u)$ for each user u can be modeled as:

$$P(\mathbf{p}_u) \propto \exp\left(\sum_{p_{ui}, p_{uj} \in \mathbf{p}_u} w_{ij} f_{ij}(p_{ui}, p_{uj}) + \sum_{(ui) \in \mathcal{V}_u} \psi_{ui}(p_{ui}, \mathbf{o}) \right) \prod_{p_{ui} \in \mathbf{p}_u} Q(p_{ui} \mid u, i) \tag{4.3.24}$$

and the joint distribution $P(\mathbf{p}_i)$ for each item i can be modeled as:

$$P(\mathbf{p}_i) \propto \exp\left(\sum_{p_{ui},p_{vi}\in\mathbf{p}_i} w_{uv} f_{uv}(p_{ui},p_{vi}) + \sum_{(ui)\in\mathcal{V}_i} \psi_{ui}(p_{ui},\mathbf{o})\right) \prod_{p_{ui}\in\mathbf{p}_i} Q(p_{ui} \mid u,i) \tag{4.3.25}$$

where there is a graph for each user or item but the weights are optimized by all users or all items.

Feature Design

A feature is essentially a function f of $n > 1$ arguments that maps the n-dimensional input into the unit interval $f : \mathbb{R}^n \to [0,1]$. We design the following kinds of features:

Correlation Features The item-item correlation is captured by the feature:

$$f_{ij}(p_{ui},p_{uj}) = g(|(p_{ui}-\bar{p}_i)-(p_{uj}-\bar{p}_j)|) \tag{4.3.26}$$

and the user-user correlation is captured by the feature

$$f_{uv}(p_{ui},p_{vi}) = g(|(p_{ui}-\bar{p}_u)-(p_{vi}-\bar{p}_v)|) \tag{4.3.27}$$

where $g(\alpha)$ normalizes feature values and α plays the role of deviation. The terms $\bar{p}_i$, $\bar{p}_j$, $\bar{p}_i$, and $\bar{p}_j$ are the item or user averages. The item-item correlation feature captures the intuition that correlated items should be ranked similarly by the same user after offsetting the goodness of each item. Similarly, the user-user correlation feature captures the intuition that correlated users should rate the same item similarly.

Attribute Features Each user u and item i has a set of attributes $\mathbf{o}_u$ and $\mathbf{o}_i$, respectively. These attributes are mapped to preferences by the following features:

$$\begin{aligned} \mathbf{f}_i(p_{ui}) &= \mathbf{o}_u g(|(p_{ui} - \bar{p}_i)|) \\ \mathbf{f}_u(p_{ui}) &= \mathbf{o}_i g(|(p_{ui} - \bar{p}_u)|) \end{aligned} \tag{4.3.28}$$

where $\mathbf{f}_i$ models which users like the item i and $\mathbf{f}_u$ models which classes of items the user u likes.

Since one correlation feature exists for each possible pair of co-rated items, the number of correlation features can be large which makes the estimation slow to converge and less robust. Therefore we only keep the correlation features if strong item-item correlation or user-user correlation exists. Specifically, the *strong correlation features* $\mathbf{f}_{\text{strong}}$ are extracted based on the *Pearson Correlation* and a user-specified *minimum correlation threshold*. Note that the correlation is calculated based on the user-wise preferences generated from *PR* thus the rule of using *PR* as input is not violated.

Parameter Estimation

In general, *MRF*-based models cannot be determined by standard maximum likelihood approaches, instead, approximation techniques are often used in practice such as *Pseudo-likelihood* [8] and *Contrastive Divergence* (CD) [30]. The *Pseudo-likelihood* leads to exact computation of the loss function and its gradient with respect to parameters, and thus faster. The CD-based methods may, on the other hand, lead to better estimation given enough time. As the experiments involve different settings and large number of features, this study employs the *Pseudo-likelihood* technique to perform efficient parameter estimation by maximizing the regularized sum of log local

likelihoods:

$$log\mathcal{L}(\mathbf{w}) = \sum_{p_{ui} \in \Pi} \log P(p_{ui} \mid \mathbf{p}_u, \mathbf{o}) - \frac{1}{2\sigma^2}\mathbf{w}^\top \mathbf{w} \tag{4.3.29}$$

where $\mathbf{w}$ are the weights and $1/2\sigma^2$ controls the regularization. To make the notation uncluttered, we write $\mathbf{p}_u$ instead of explicitly as $\mathbf{p}_u \backslash p_{ui}$. In this section we describe the parameter estimation of item-item correlations, where the user-user correlations can be estimated in the same way by replacing items with users.

The local likelihood in Eq. 4.3.29 is defined as:

$$P(p_{ui} \mid \mathbf{p}_u, \mathbf{o}) = \frac{1}{Z_{ui}(\mathbf{o})} Q(p_{ui} \mid u, i)\psi_{ui}(p_{ui}, \mathbf{o}) \prod_{p_{uj} \in \mathbf{p}_u} \psi_{ij}(p_{ui}, p_{uj}) \tag{4.3.30}$$

where $Z_{ui}(\mathbf{o})$ is the normalization term:

$$Z_{ui} = \sum_{p_{ui}=l_{min}}^{l_{max}} Q(p_{ui} \mid u, i)\psi_{ui}(p_{ui}, \mathbf{o}) \prod_{p_{uj} \in \mathbf{p}_u} \psi_{ij}(p_{ui}, p_{uj}) \tag{4.3.31}$$

where l_{min} is the first and l_{max} is the last interval, i.e., 1 and 3 in our settings.

To optimize the parameters, we use the stochastic gradient ascent procedure that updates the parameters by passing through the set of ratings of each user:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta \nabla log\mathcal{L}(\mathbf{w}) \tag{4.3.32}$$

where η is the learning rate. More specifically, for each p_{ui} we update the attribute weights $\mathbf{w}_o = \{\mathbf{w}_u, \mathbf{w}_i\}$ and correlation weight w_{ij} for each neighbor $p_{uj} \in \mathbf{p}_u$ using the gradient of the log pseudo-likelihood

$$\frac{\partial log\mathcal{L}}{\partial \mathbf{w}_o} = \mathbf{f}_o(p_{ui}, \mathbf{o}) - \sum_{p_{ui}=l_{min}}^{l_{max}} P(p_{ui} \mid \mathbf{p}_u, \mathbf{o})\mathbf{f}_o(p_{ui}, \mathbf{o}) - \frac{w_i}{\sigma^2} \tag{4.3.33}$$

$$\frac{\partial log\mathcal{L}}{\partial w_{ij}} = f_{ij}(p_{ui}, p_{uj}) - \sum_{p_{ui}=l_{min}}^{l_{max}} P(p_{ui} \mid \mathbf{p}_u, \mathbf{o}) f_{ij}(p_{ui}, p_{uj}) - \frac{w_{ij}}{\sigma^2} \tag{4.3.34}$$

Item Recommendation

The ultimate goal of *RecSys* is often to rank the items and recommend the Top-N items to the user. To obtain the item rankings, *PrefMRF* estimates distributions over user-wise preferences which can be converted into point estimate: The *PrefCRF* produces distributions over the user-wise preferences, which can be converted into point estimate:

Most Likely Preference The preference can be determined by selecting the preference with the greatest mass in local likelihood:

$$\hat{p}_{ui} = \arg\max_{p_{ui}} P(p_{ui} \mid \mathbf{p}_u, \mathbf{o}) \tag{4.3.35}$$

where the local likelihood is given by Eq. 4.3.30. The local likelihood serves as a *confidence* measure.

Smoothed Expectation When the prediction is not strict to discrete values, the expectation can be used instead:

$$\hat{p}_{ui} = \sum_{p_{ui}=l_{min}}^{l_{max}} p_{ui} P(p_{ui} \mid \mathbf{p}_u, \mathbf{o}) \tag{4.3.36}$$

where l refers to the intervals of user-wise preferences: from least to most preferred. Note that l is limited to the simplest case of 3 intervals in our settings, but more intervals are possible.

The predictions by item-item correlation and user-user correlations can be merged by taking the mean value, and then items can be sorted and ranked accordingly. Finally, Alg. 2 summarizes the learning and prediction procedures for the *PrefCRF*.

Algorithm 2 *PrefCRF* Algorithm

Input: Explict or implicit preferences.

Step 1: Infer PR from preferences.

Step 2: Predict user-wise preferences $\hat{p}_{ui}$ using Eq. 4.2.2.

Step 3: Predict distribution for each $\hat{p}_{ui}$ using Eq. 4.3.6.

Step 4: Repeat

for each $u \in \mathcal{U}$ **do**

for each $p_{ui} \in \mathbf{p}_u$ **do**

Compute normalization term Z_{ui} using Eq. 4.3.31

Compute local likelihood using Eq. 4.3.30

Compute attribute feature $\mathbf{f}_i$ and $\mathbf{f}_u$ using Eq. 4.3.28

Compute gradients for attribute features $\mathbf{f}_o$ using Eq. 4.3.33

Update $\mathbf{w}_o$ with the gradient using Eq. 4.3.32

for each $p_{uj} \in \mathbf{p}_u$, $i \neq j \wedge f_{ij} \in \mathbf{f}_{\text{strong}}$ **do**

Get correlation feature f_{ij} and f_{uv} using Eq. 4.3.26 and Eq. 4.3.27

Get gradient for correlation feature f_{ij} using Eq. 4.3.34

Update w_{ij} with the gradient using Eq. 4.3.32

end for

end for

end for

Until stopping criteria met

Predictions:

* Predict user-wise preferences using Eq. 4.3.36 or Eq. 4.3.35.

* Select Top-N items according to estimated preferences.

Computational Complexity

We perform the computational complexity analysis on the *PrefMRF* and its underlying *PrefNMF* algorithms. Given n users and m items each with d_u and d_i preferences, respectively. Let us temporarily ignore the user-specified latent factors. Then the complexity of both *PrefNMF* and *PrefMRF* is $O(nd_u^2)$. However, in practice few item co-rated by the same user are strong neighbors of each other due to the correlation threshold defined in Section 4.3.4. As a result, the computation time of *PrefMRF* tends to be $O(nd_uc)$ where c is a factor of correlation threshold.

4.4 Experiment and Analysis

Datasets

Ideally, the experiments should be conducted on datasets that contain user preferences in two forms: *PR* and *absolute ratings*. Unfortunately no such a dataset is publicly available at the moment, therefore we choose to compile the rating-based datasets into the form of *PR*. We use the same conversion method as in [19] by comparing the ratings of each ordered pair of items co-rated by the same user. For example, 1 is assigned to the *PR* π_{uij} if $p_{ui} > p_{uj}$; 0 is assigned if $p_{ui} < p_{uj}$, and 0.5 is assigned if $p_{ui} = p_{uj}$.

Experiments were conducted on two datasets: the *MovieLens*-1M [1] and the *EachMovie* [2] datasets. The *MovieLens*-1M dataset contains more than 1 million ratings

[1] `http://grouplens.org/datasets/movielens`
[2] `http://grouplens.org/datasets/eachmovie`

by 6,040 users on 3,900 movies. The *EachMovie* dataset contains 2.8 million ratings by 72,916 users on 1,628 movies. The minimum rating is 1 and we cap the maximum at 5 for both datasets. The impact of side information is studied on the *MovieLens*-1M dataset which provides *gender*, *age*, and *occupation* information about users and *genres* of movies.

For a reliable and fair comparison, each dataset is split into train and test sets, and the following settings are aligned to related work [83]. As the sparsity levels differ between the *MovieLens*-1M and the *EachMovie* datasets, different number of ratings are reserved for training and the rest for testing. Specifically, for each user in the *MovieLens*-1M we randomly select $N = 30, 40, 50, 60$ ratings for training, and put the rest for testing. Some users do not have enough ratings thus were excluded from experiments. The *EachMovie* has less items but much more users comparing to *MovieLens*-1M, therefore it is safe to remove some less active users and we set $N = 70, 80, 90, 100$ to investigate the performance on dense dataset.

Evaluation Metrics

Traditional recommender systems aim to optimize *RMSE* or *MAE* which emphasizes on absolute ratings. However, the ultimate goal of recommender systems is usually to obtain the ranking of items [42], where good performance on *RMSE* or *MAE* may not be translated into good ranking results [42]. Therefore, we employ two evaluation metrics: *Normalized Cumulative Discounted Gain@T* (NDCG@T) [34] which is popular in academia, and *Mean Average Precision@T* (MAP@T) [13] which

is popular in contests [3]. Among them, the NDCG@T metric is defined as

$$\text{NDCG@}T = \frac{1}{K(T)} \sum_{t=1}^{T} \frac{2^{r_t} - 1}{\log_2 (t+1)} \tag{4.4.1}$$

where r_t is the relevance judgment of the item at position t, and $K(T)$ is the normalization constant. The MAP@T metric is defined as

$$\text{MAP@}T = \frac{1}{|\mathcal{U}_{test}|} \sum_{u \in \mathcal{U}_{test}} \sum_{t=1}^{T} \frac{P_u(t)}{min(m_u, t)} \tag{4.4.2}$$

where m_u is the number relevant items to user u, and $P_u(t)$ is user u's precision at position t. Both metrics are normalized to $[0, 1]$, and a higher value indicates better performance.

These metrics, together with other ranking-based metrics, require a set of relevant items to be defined in the test set such that the predicted rankings can be evaluated against. The relevant items can be defined in different ways. For example, for each user we can consider only 5-star items in the testing set as relevant items, or those items above each user's average as relevant items. In this paper, we follow the same selection criteria used in the related work [12, 38] to consider items with the highest ratings as relevant.

Parameter Setting

For a fair comparison, we fix the number of latent factors to 50 for all algorithms, the same as in related work [15]. The number of neighbors for *KNN* algorithms is set to 50. We vary the minimum correlation threshold to examine the performances with different number of features. Different values of regularization coefficient are also tested.

[3] KDD Cup 2012 and Facebook Recruiting Competition

4.4.1 Results and Analysis

We first compare the performance of the proposed *PrefMRF* and *PrefCRF* models with four related models: *KNN*, *NMF*, *PrefKNN*, and *PrefNMF*, where the *PrefNMF* is the targeted model, and then investigate the impact of parameter settings.

Comparison on Top-N Recommendation

Comparison of these algorithms is conducted by measuring the *NDCG* and the *MAP* metrics on Top-N recommendation tasks. Each experiment is repeated ten times with different random seeds and we report the mean results with standard deviations on *MovieLens*-1M dataset in Table 4.3 and *EachMovie* dataset in Table 4.4. Note that only the *MovieLens*-1M dataset has side information which is used by the *PrefCRF* model. The *PrefMRF* as well as other models are based on only preferences data. We also report the *NDCG* and *MAP* values by varying the position T (i.e., how many items to recommend) in Fig. 4.3 and Fig. 4.4 for *MovieLens*-1M dataset and in Fig. 4.5 and Fig. 4.6 for *MovieLens*-1M dataset. The following observations can be made based on the results.

Firstly, the *KNN* and the *PrefKNN* methods didn't perform well on *MovieLens*-1M comparing with *Matrix Factorization* based methods. One possible reason is that predictions are made based only on the neighbors, and as a result too much information has been ignored especially when the dataset is large. However, the performance of *KNN*-based methods has improved on the *EachMovie* dataset as we reserved more ratings for training, i.e., better neighbors can be found for prediction.

Secondly, *PrefNMF* outperforms *NMF* on *MovieLens*-1M dataset which is consistent to the results reported in [19]. However, *PreNMF* does not perform well

Table 4.3: Results over ten runs on *MovieLens*-1M dataset.

	Given 30			
Algorithm	NDCG@5	NDCG@10	MAP@5	MAP@10
UserKNN	0.3969 ± 0.0020	0.4081 ± 0.0029	0.2793 ± 0.0021	0.2744 ± 0.0025
NMF	0.5232 ± 0.0057	0.5195 ± 0.0040	0.3866 ± 0.0055	0.3549 ± 0.0037
PrefKNN	0.3910 ± 0.0044	0.4048 ± 0.0038	0.2745 ± 0.0043	0.2720 ± 0.0037
PrefNMF	0.5729 ± 0.0049	0.5680 ± 0.0041	0.4387 ± 0.0046	0.3992 ± 0.0033
PrefMRF	$\mathbf{0.6020 \pm 0.0050}$	$\mathbf{0.5934 \pm 0.0039}$	$\mathbf{0.4721 \pm 0.0050}$	$\mathbf{0.4244 \pm 0.0036}$
PrefCRF	$\mathbf{0.6316 \pm 0.0076}$	$\mathbf{0.5966 \pm 0.0028}$	$\mathbf{0.6254 \pm 0.0073}$	$\mathbf{0.4245 \pm 0.0028}$
	Given 40			
Algorithm	NDCG@5	NDCG@10	MAP@5	MAP@10
UserKNN	0.4108 ± 0.0040	0.4252 ± 0.0036	0.2936 ± 0.0036	0.2877 ± 0.0034
NMF	0.5323 ± 0.0050	0.5291 ± 0.0034	0.3976 ± 0.0045	0.3631 ± 0.0035
PrefKNN	0.4122 ± 0.0024	0.4283 ± 0.0024	0.2944 ± 0.0023	0.2904 ± 0.0023
PrefNMF	0.5773 ± 0.0037	0.5732 ± 0.0028	0.4437 ± 0.0041	0.4019 ± 0.0032
PrefMRF	$\mathbf{0.6215 \pm 0.0029}$	$\mathbf{0.6140 \pm 0.0023}$	$\mathbf{0.4844 \pm 0.0025}$	$\mathbf{0.4420 \pm 0.0020}$
PrefCRF	$\mathbf{0.6435 \pm 0.0064}$	$\mathbf{0.6092 \pm 0.0023}$	$\mathbf{0.6420 \pm 0.0062}$	$\mathbf{0.4392 \pm 0.0021}$
	Given 50			
Algorithm	NDCG@5	NDCG@10	MAP@5	MAP@10
UserKNN	0.4273 ± 0.0040	0.4424 ± 0.0027	0.3078 ± 0.0038	0.3015 ± 0.0026
NMF	0.5360 ± 0.0041	0.5326 ± 0.0036	0.4010 ± 0.0040	0.3669 ± 0.0025
PrefKNN	0.4326 ± 0.0027	0.4483 ± 0.0030	0.3125 ± 0.0024	0.3070 ± 0.0022
PrefNMF	0.5761 ± 0.0067	0.5745 ± 0.0035	0.4424 ± 0.0064	0.4019 ± 0.0033
PrefMRF	$\mathbf{0.6248 \pm 0.0053}$	$\mathbf{0.6172 \pm 0.0032}$	$\mathbf{0.4896 \pm 0.0053}$	$\mathbf{0.4460 \pm 0.0027}$
PrefCRF	$\mathbf{0.6648 \pm 0.0055}$	$\mathbf{0.6158 \pm 0.0018}$	$\mathbf{0.6580 \pm 0.0059}$	$\mathbf{0.4471 \pm 0.0024}$
	Given 60			
Algorithm	NDCG@5	NDCG@10	MAP@5	MAP@10
UserKNN	0.4480 ± 0.0044	0.4622 ± 0.0035	0.3266 ± 0.0036	0.3163 ± 0.0027
NMF	0.5462 ± 0.0068	0.5409 ± 0.0063	0.4109 ± 0.0069	0.3734 ± 0.0055
PrefKNN	0.4526 ± 0.0062	0.4689 ± 0.0039	0.3301 ± 0.0051	0.3223 ± 0.0033
PrefNMF	0.5756 ± 0.0062	0.5733 ± 0.0048	0.4409 ± 0.0059	0.4007 ± 0.0037
PrefMRF	$\mathbf{0.6422 \pm 0.0037}$	$\mathbf{0.6301 \pm 0.0037}$	$\mathbf{0.5112 \pm 0.0035}$	$\mathbf{0.4600 \pm 0.0026}$
PrefCRF	$\mathbf{0.6772 \pm 0.0074}$	$\mathbf{0.6242 \pm 0.0018}$	$\mathbf{0.6715 \pm 0.0072}$	$\mathbf{0.4536 \pm 0.0016}$

Table 4.4: Results over ten runs on *EachMovie* dataset without side information.

	Given 70			
Algorithm	NDCG@5	NDCG@10	MAP@5	MAP@10
UserKNN	0.7088 ± 0.0020	0.7115 ± 0.0015	0.6012 ± 0.0027	0.5767 ± 0.0017
NMF	0.7581 ± 0.0022	0.7577 ± 0.0017	0.6524 ± 0.0026	0.6225 ± 0.0020
PrefKNN	0.7260 ± 0.0022	0.7307 ± 0.0018	0.6197 ± 0.0020	0.5990 ± 0.0016
PrefNMF	0.7408 ± 0.0033	0.7348 ± 0.0039	0.6330 ± 0.0035	0.5800 ± 0.0038
PrefMRF	$\mathbf{0.8317 \pm 0.0032}$	$\mathbf{0.8245 \pm 0.0029}$	$\mathbf{0.7512 \pm 0.0039}$	$\mathbf{0.6921 \pm 0.0034}$
	Given 80			
Algorithm	NDCG@5	NDCG@10	MAP@5	MAP@10
UserKNN	0.7146 ± 0.0018	0.7168 ± 0.0017	0.6070 ± 0.0021	0.5825 ± 0.0019
NMF	0.7636 ± 0.0021	0.7638 ± 0.0018	0.6583 ± 0.0025	0.6286 ± 0.0018
PrefKNN	0.7337 ± 0.0028	0.7377 ± 0.0018	0.6271 ± 0.0029	0.6057 ± 0.0021
PrefNMF	0.7422 ± 0.0036	0.7319 ± 0.0040	0.6329 ± 0.0039	0.5774 ± 0.0033
PrefMRF	$\mathbf{0.8364 \pm 0.0036}$	$\mathbf{0.8232 \pm 0.0030}$	$\mathbf{0.7553 \pm 0.0038}$	$\mathbf{0.6991 \pm 0.0032}$
	Given 90			
Algorithm	NDCG@5	NDCG@10	MAP@5	MAP@10
UserKNN	0.7191 ± 0.0022	0.7279 ± 0.0028	0.6120 ± 0.0021	0.5933 ± 0.0013
NMF	0.7712 ± 0.0039	0.7692 ± 0.0033	0.6663 ± 0.0043	0.6431 ± 0.0034
PrefKNN	0.7418 ± 0.0028	0.7421 ± 0.0015	0.6357 ± 0.0030	0.6192 ± 0.0020
PrefNMF	0.7456 ± 0.0031	0.7358 ± 0.0038	0.6357 ± 0.0040	0.5819 ± 0.0036
PrefMRF	$\mathbf{0.8394 \pm 0.0035}$	$\mathbf{0.8249 \pm 0.0032}$	$\mathbf{0.7474 \pm 0.0037}$	$\mathbf{0.7046 \pm 0.0032}$
	Given 100			
Algorithm	NDCG@5	NDCG@10	MAP@5	MAP@10
UserKNN	0.7279 ± 0.0028	0.7277 ± 0.0015	0.6238 ± 0.0032	0.5973 ± 0.0021
NMF	0.7741 ± 0.0030	0.7717 ± 0.0028	0.6719 ± 0.0034	0.6411 ± 0.0030
PrefKNN	0.7505 ± 0.0019	0.7511 ± 0.0012	0.6478 ± 0.0020	0.6231 ± 0.0014
PrefNMF	0.7391 ± 0.0033	0.7298 ± 0.0034	0.6318 ± 0.0039	0.5761 ± 0.0039
PrefMRF	$\mathbf{0.8418 \pm 0.0031}$	$\mathbf{0.8277 \pm 0.0030}$	$\mathbf{0.7546 \pm 0.0038}$	$\mathbf{0.7063 \pm 0.0036}$

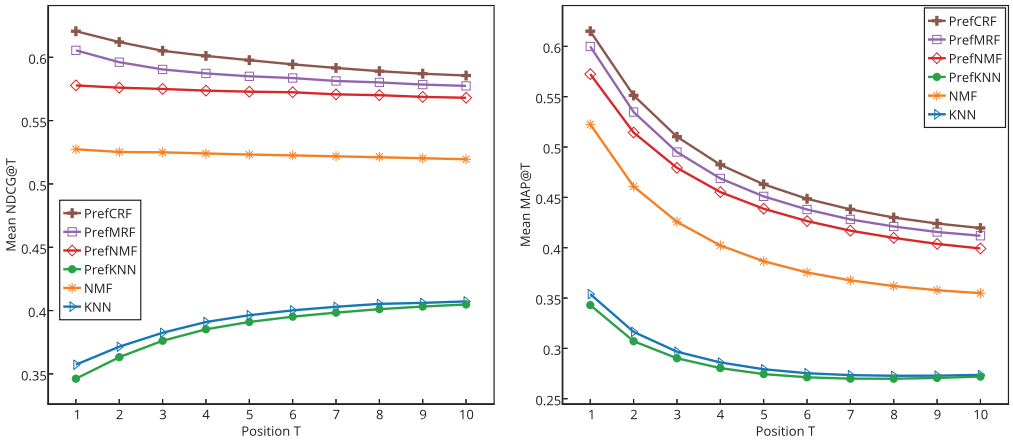
on *EachMovie* where its performance is only slightly better than user-based *KNN*. The reason behind could be the *EachMovie* is much denser than the *MovieLens*-1M dataset, which makes the number of *PR* huge and difficult to tune optimal parameters. Besides, we observe that *PrefNMF* in general only achieves a slight improvement with more training data and even drops a bit with *Given* 60. Similarly for the *EachMovie* dataset. With these observations, it appears that for a given number of users, the *PrefNMF* can be trained reasonably well with fewer data.

Finally, the proposed *PrefMRF* and *PrefCRF* have made further improvement on both datasets upon the *PrefNMF* through capturing both *LS* and *GS*, as well as exploiting side information. From Fig. 4.3 and Fig. 4.4 we can see that the algorithms stabilized around position 10 and *PrefMRF* and *PrefCRF* consistently deliver better performance than others. It should be noted that the performance of *PrefMRF* and *PrefCRF* rely on their underlying model that captures the *GS*. In other words, the performance may vary when the *PrefNMF* is replaced with other alternative methods such as [45].

To confirm the improvements, a paired t-test (two-tailed) with a significance level of 95% has been applied to the best *PrefMRF* and the second best *PrefNMF*. Results shown in Table 4.5 confirm that the performance of models with and without capturing the *LS* is statistically significant.

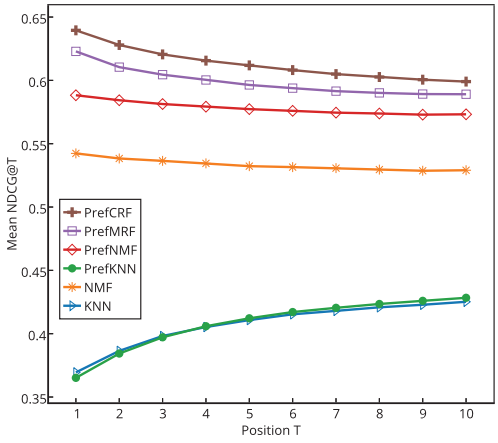
Performance on Various Data Sparsity Levels

To thoroughly examine the performance of these algorithms, we compare their performances under different settings of training set sizes: from *Given* 30 to *Given* 60 on *MovieLens*-1M dataset, and from *Given* 70 to *Given* 100 on *EachMovie* dataset.

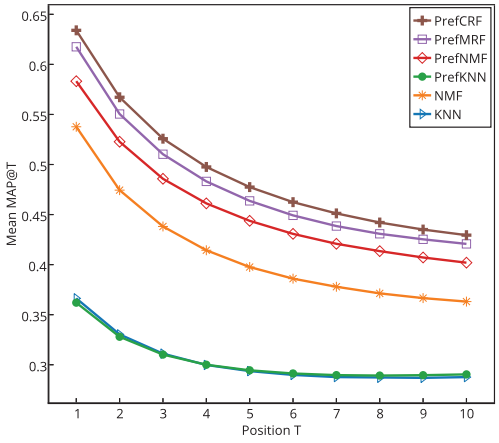


(a) NDCG@T (Given 30)

(b) MAP@T (Given 30)

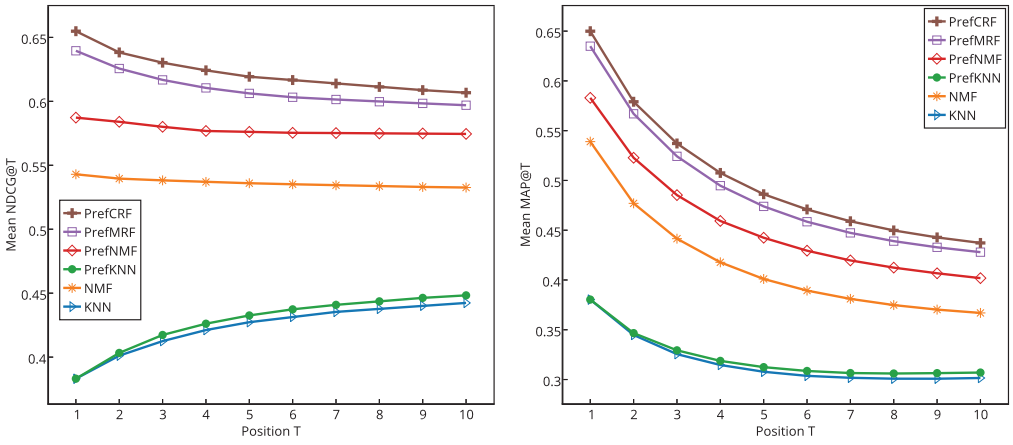


(c) NDCG@T (Given 40)



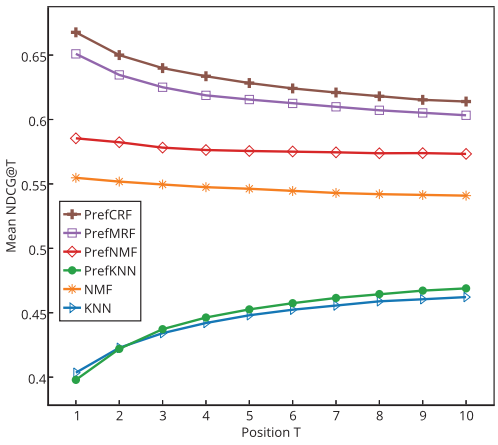
(d) MAP@T (Given 40)

Figure 4.3: Performance of different position T on MovieLens-1M dataset (Sparse).

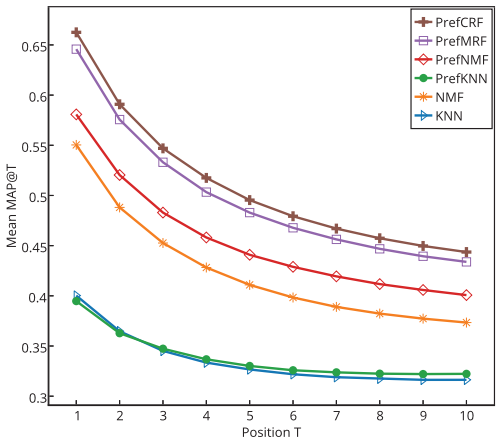


(a) NDCG@T (Given 50)

(b) MAP@T (Given 50)

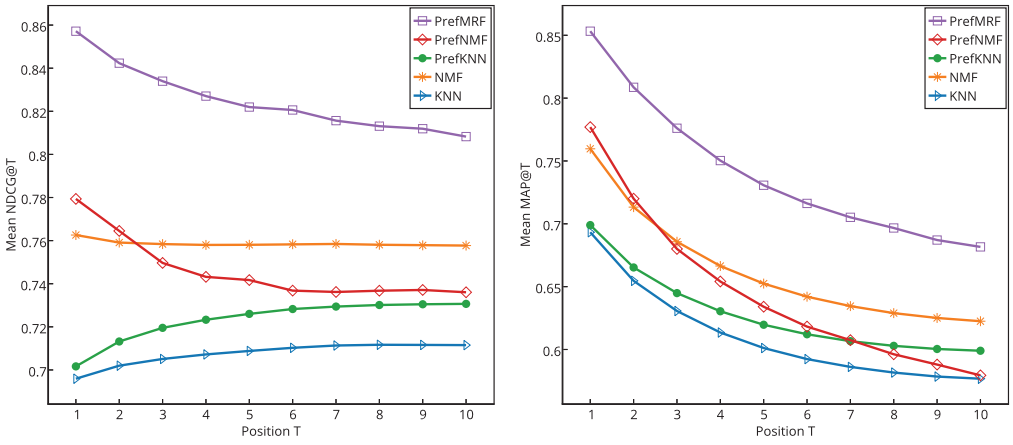


(c) NDCG@T (Given 60)



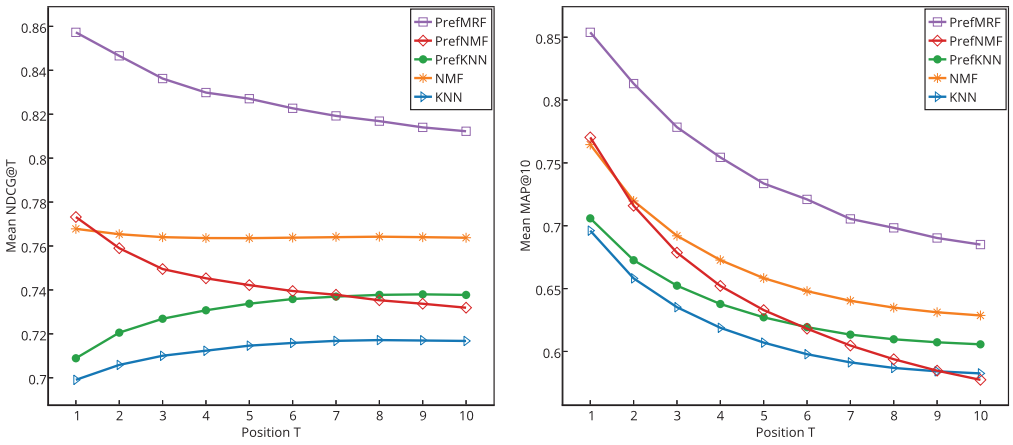
(d) MAP@T (Given 60)

Figure 4.4: Performance of different position T on MovieLens-1M dataset (Dense).



(a) NDCG@T (Given 70)

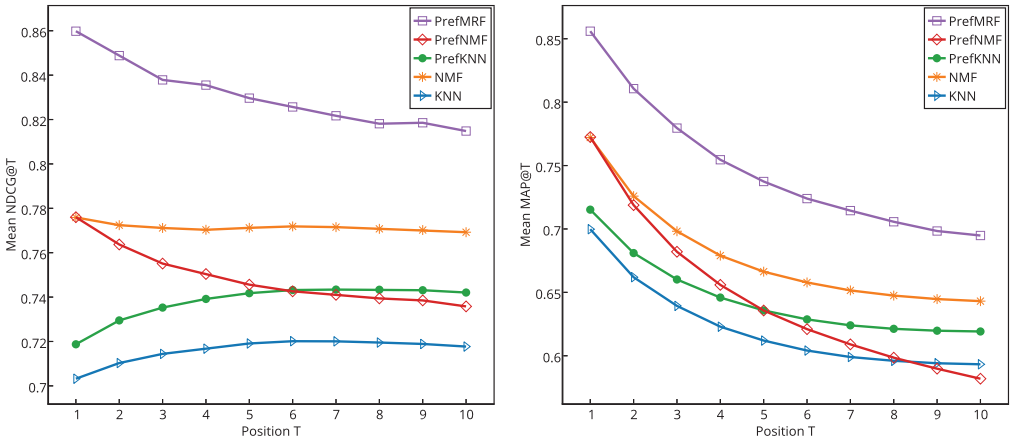
(b) MAP@T (Given 70)



(c) NDCG@T (Given 80)

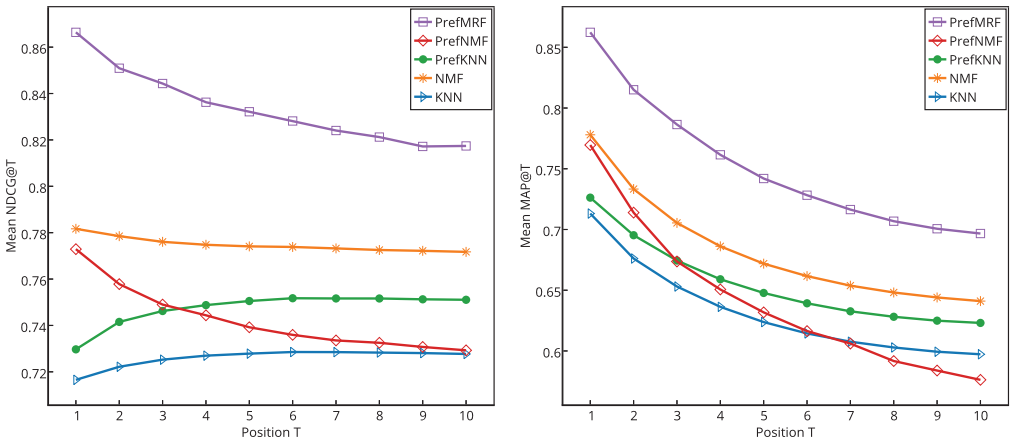
(d) MAP@T (Given 80)

Figure 4.5: Performance of different position T on EachMovie dataset (Sparse).



(a) NDCG@T (Given 90)

(b) MAP@T (Given 90)



(c) NDCG@T (Given 100)

(d) MAP@T (Given 100)

Figure 4.6: Performance of different position T on EachMovie dataset (Dense).

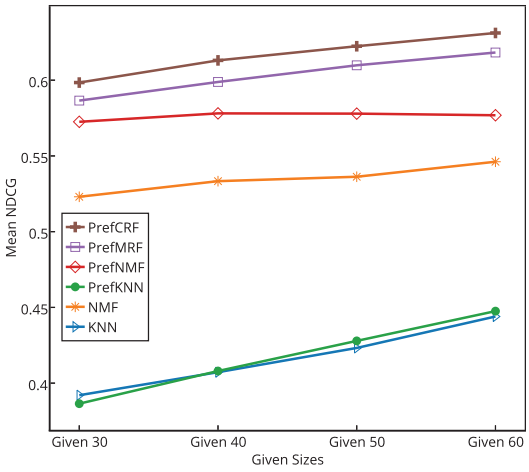
Table 4.5: Paired t-test for *PrefMRF* and *PrefNMF*.

Settings			t-test statistics		
Dataset	Sparsity	Metric	df	t	p-value
MovieLens	*Given* 60	*NDCG*@10	9	16.6218	< 0.00001
MovieLens	*Given* 60	*MAP*@10	9	23.5517	< 0.00001
EachMovie	*Given* 100	*NDCG*@10	9	72.4189	< 0.00001
EachMovie	*Given* 100	*MAP*@10	9	72.1346	< 0.00001

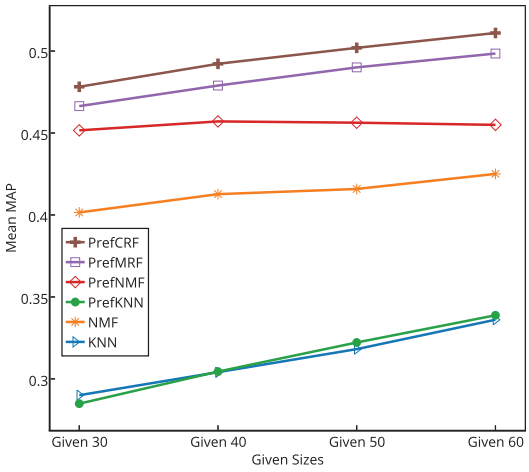
Results are plotted in Fig. 4.7. It can be observed that in general more training data result in better performance. However, *PrefNMF* does not gain much benefit from more data and even perform slightly worse in *Given* 60. The *PrefMRF* on the other hand consistently gains performance from more data as the *LS* information can be better captured.

Impact of Minimum Correlation Threshold

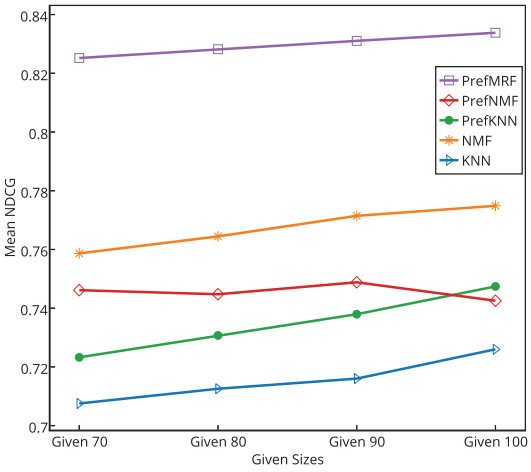
As described in Section 4.3.4, a *minimum correlation threshold* is required to control the number of features in the *PrefMRF* model. By default, each pair of co-rated items has a feature which results in a large number of features. However, many of these features are useless if the item-item correlation are weak. To make the model more robust and with faster convergence, a *minimum correlation threshold* is applied to remove weak features. Specifically, the feature is removed if two items has a correlation measured by *Pearson* correlation less than the threshold. Results are



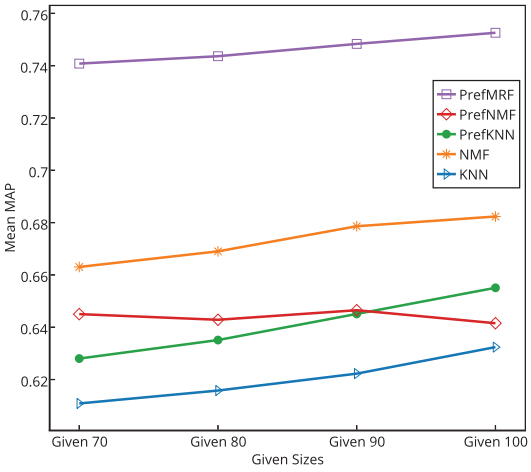
(a) Mean NDCG by training set sizes on MovieLens-1M dataset.



(b) Mean MAP by training set sizes on MovieLens-1M dataset.



(c) Mean NDCG by training set sizes on EachMovie dataset.

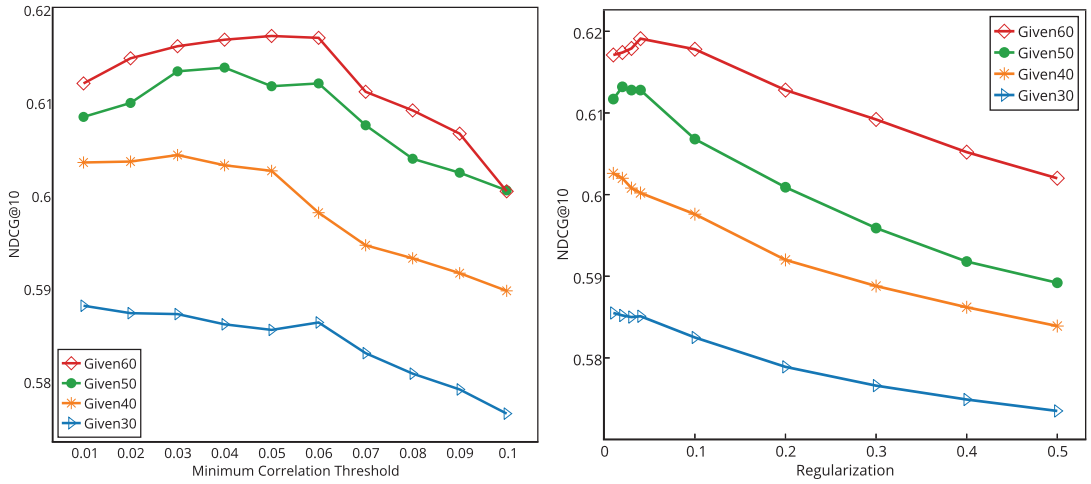


(d) Mean MAP by training set sizes on EachMovie dataset.

Figure 4.7: Impact of Sparsity Levels.

plotted in Fig. 4.8(a).

It can be observed that a smaller correlation threshold delivers better performance, however, the number of features will also increase. To balance the performance and computation time, it is wise to select a moderate level of threshold depending on the dataset.



(a) NDCG@10 by Threshold

(b) NDCG@10 by Regularization

Figure 4.8: Impact of Parameters (MovieLens-1M)

Impact of Regularization Coefficient

As the number of features in *PrefMRF* can be large, the model might be prone to over-fitting. Therefore, we investigate the impact of regularization settings as plotted in Fig. 4.8(b).

We observe that the performance is better when a small regularization penalty applies. In other words, the *PrefMRF* can generalize reasonable well without too much regularization. This can be explained as the weights of item-item correlations

are not user-specific but shared by all users, thus they cannot over-fit every user perfectly.

4.5 Summary

In this chapter we presented the *PrefMRF* model, which takes advantages of both the representational power of the *MRF* and the ease of modeling preference relations by the *PrefNMF*. To the best of our knowledge, there was no *PR*-based method that can capture both *LS* and *GS*, until the *PrefMRF* model is proposed in this work. In addition, side information can be easily incorporated by extending the *PrefMRF* model to the *PrefCRF* model. Experiment results on public datasets demonstrate that both types of interactions have been properly captured by *PrefMRF*, and significantly improved Top-N recommendation performance has been achieved. In addition, the *PrefMRF* model provides a generic interface for unifying various *PR*-based methods other than the *PrefNMF* used in this paper. In other words, any *PR*-based method that captures the *GS* should be able to take advantages from *PrefMRF* to capture *LS* as well.

For future work, we would like to work on several directions. First, the computation efficiency of *PR*-based matrix factorization needs to be improved given that the number of preference relations is usually much larger than absolute ratings. This is feasible as each user has his/her own set of preference relations, thus the learning process can be parallelized. Secondly, it would be interesting to see how *PR*-based methods perform on real implicit preferences dataset, such as *page views* and *mouse clicks*.

Chapter 5

Learning from Heterogeneous Data Sources for Improved Top-N Recommendation

5.1 Introduction

RecSys aim to recommend users with some of their potentially interesting items, which can be virtually anything ranging from *movies* to *tourism attractions*. To identify the appropriate items, RecSys attempts to exploit *user preferences* [41] and various *side information* [6]. However, the *cold-start* problem [72] raises when little information is known for *cold* users or items, e.g., a newly registered user. To alleviate the *cold-start* problem, additional information, which is usually heterogeneous, must be acquired.

The last decade has seen a growing trend towards creating and managing more profiles in *Online Social Networks* (OSN), such as *Facebook*, *LinkedIn*, and *Netflix* etc. In light of this trend, it becomes possible to alleviate the *cold-start* problem by learning user preferences from heterogeneous data sources, e.g., a *cold user* of *Netflix* may have been used *Facebook* for a while. Nevertheless, user preferences from heterogeneous sources are heterogeneous whereas existing recommendation techniques

require the user preferences to be homogeneous. For example, user preferences are expressed as 5-star ratings for *Netflix* [1], 6-star ratings for *EachMovie* [2], and binary ratings for *Facebook*. Sometimes the user preferences may not even be expressed as ratings but as implicit feedbacks such as *page views* and *clicks*. Moreover, user preferences collected from heterogeneous sources may have different biases, as the user preferences not only reflect the quality of the items but also the quality of service providers. When the user preferences are heterogeneous and with biases, existing recommendation techniques cannot be directly applied. To the best of our knowledge, no previous work has considered the heterogeneous sources problem, in which the user preferences are heterogeneous with biases.

In this work, we identify that the *preference relations* (PR), which measures the relative ordering between items, could be a key to the heterogeneous sources problem. With the assistance of *PR*, user preferences from heterogeneous sources can be fused seamlessly. For example, user preferences expressed as 5-star ratings, binary ratings, and *page views* can not be directly fused in general. However, all those user preferences can be deduced into the *PR* format by performing pairwise comparison on items. Once the user preferences are represented in *PR*, a direct merge can be performed. In fact, converting user preferences into *PR* not only provides a method to merge heterogeneous data but also reduces the biases that come with heterogeneity, i.e., the relative ordering of items is resistant to biases. In addition to collecting more user preferences, another method to alleviate the *cold-start* problem is to utilize the heterogeneous side information, such as attributes like *age* or *occupation* of users and items.

[1] `http://www.netflixprize.com`

[2] `http://grouplens.org/datasets/eachmovie`

This chapter aims to propose the *Preference Relation-based Conditional Random Fields* (PrefCRF) model to learn from heterogeneous data sources as well as the heterogeneous side information The remaining part of this paper is organized as follows. Section 5.2 introduces the basic concepts of learning from heterogeneous sources and preference relations, followed by a review of related work. Section 5.3 is devoted to the proposed *PrefCRF* model. Benchmark results on Top-N recommendation are presented in Section 5.4. Finally, Section 5.5 concludes this chapter by summarizing the main contributions and envisaging future works.

5.2 Preliminaries

This section briefly summarizes necessary background related to the *heterogeneous sources* problem and the *preference relations* that form the basis of our solution.

5.2.1 Heterogeneous Sources

User preferences are usually assumed to come from a single *homogeneous source*. This assumption is becoming invalid given the rapid development of *OSN*, in which users maintain multiple profiles and the form of preferences diverges. We define two sources as heterogeneous if their preferences are *1*) in different forms, e.g., *ratings* and *clicks*;

2) in different scales, e.g., *5-star* scale and *6-star* scale;

3) or biased differently due to factors irrelevant to the items' quality, e.g., quality of the service providers. Based on this definition, not only the physically separated

sources are heterogeneous but a source changed significantly is also considered heterogeneous to itself.

In general, user preferences from heterogeneous sources cannot be merged directly as they may be in different forms. Even if their forms are the same, the scales could be different, where a force casting may change the meaning of preferences. In case that the scales are the same, biases are still introduced by the sources which make the recommendations inaccurate.

5.2.2 Preference Relation

Preference relation (PR) encodes user preferences in the form of *relative* ordering between items, which is a useful alternative representation to *absolute* ratings as suggested in recent works [12, 19]. In fact, existing preferences such as ratings or other types of preferences can be easily represented as PR and then merged into a single dataset as shown in Fig. 5.1. . This property is particularly useful for the *cold-start* problem but has been overlooked in literature.

We formally define the PR as follows. Let $\mathcal{U} = \{u\}^n$ and $\mathcal{I} = \{i\}^m$ denote the set of n users and m items, respectively. The PR of a user $u \in \mathcal{U}$ between items i and j is encoded as π_{uij}, which indicates the strength of user u's preference relation for the ordered item pair (i, j). A higher value of π_{uij} indicates a stronger preference to the first item over the second item.

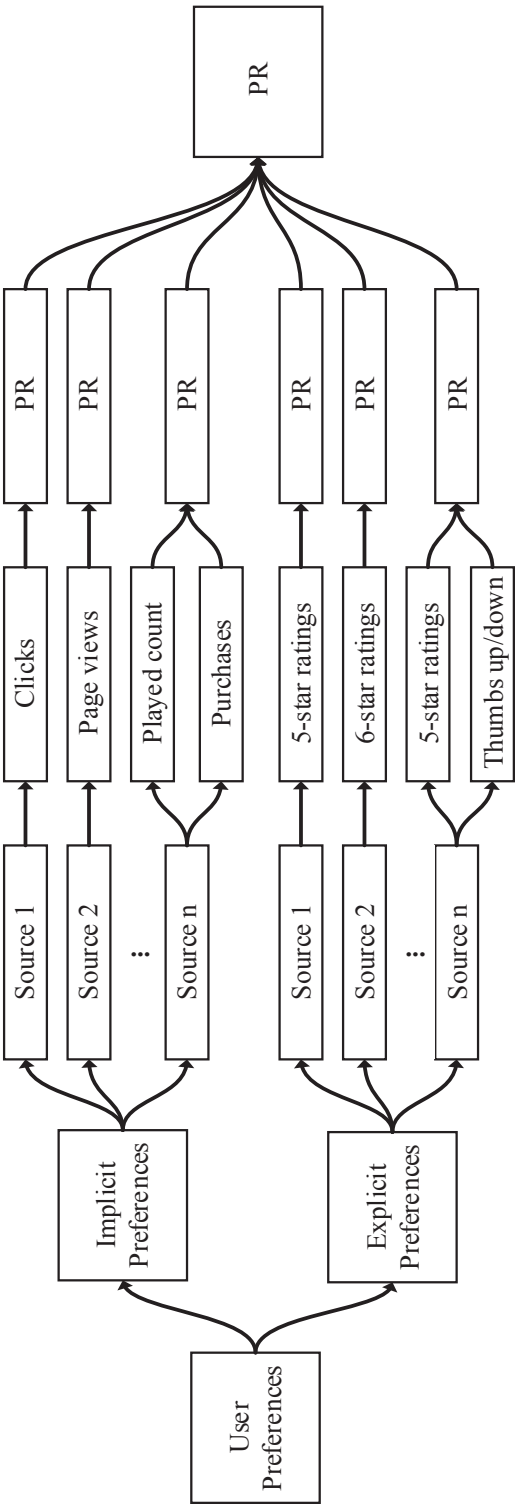


Figure 5.1: Flow from user preferences to PR

Definition 7 (Preference Relation). The preference relation is defined as

$$\pi_{uij} = \begin{cases} (\frac{2}{3}, 1] & \text{if } i \succ j \text{ (u prefers i over j)} \\ [\frac{1}{3}, \frac{2}{3}] & \text{if } i \simeq j \text{ (i and j are equally preferable)} \\ [0, \frac{1}{3}) & \text{if } i \prec j \text{ (u prefers j over i)} \end{cases} \tag{5.2.1}$$

where $\pi_{uij} \in [0, 1]$ and $\pi_{uij} = 1 - \pi_{uji}$.

An interval is allocated for each preference category, i.e., *preferred*, *equally preferred*, and *less preferred*. Indeed, each preference category can be further break down into more intervals, though here in this paper we consider the minimal case of 3 intervals.

Similar to [12], the *PR* can be converted into *user-wise preferences* over items which encode the ranking of items evaluated by a particular user.

Definition 8 (User-wise Preference). The user-wise preference is defined as

$$p_{ui} = \frac{\sum_{j \in \mathcal{I}_u \setminus i} [\![\pi_{uij} > \frac{2}{3}]\!] - \sum_{j \in \mathcal{I}_u \setminus i} [\![\pi_{uij} < \frac{1}{3}]\!]}{|\Pi_{ui}|} \tag{5.2.2}$$

where $\mathcal{I}_u$ is the set of items related to user u, $[\![\cdot]\!]$ gives 1 for *true* and 0 for *false*, and Π_{ui} is the set of user u's PR related to item i.

The user-wise preference p_{ui} falls in the interval $[-1, 1]$, where 1 and -1 indicate that item i is the most and the least preferred item for u, respectively.

5.2.3 Related Work

Preference Relation (PR) has been widely studied in the field of *Information Retrieval* [33]. Nevertheless, *PR*-based RecSys have only emerged recently [12,19,45,64].

User preferences are usually categorized into three classes: *pointwise*, *pairwise*, and *listwise*. The *pointwise* preferences correspond to the *absolute* ratings widely used in RecSys, the *pairwise* preferences correspond to the *PR* presented in this work, and the *listwise* preferences is indeed the output of Top-N RecSys.

Though RecSys is not limited to *absolute* ratings, the recommendation task is usually considered as a rating prediction problem. Recently, a considerable literature [12, 19, 45, 64, 74] has grown up around the theme of *relative* preferences. Meanwhile, recommendation task is also shifting from rating prediction to item ranking [74, 83], in which the ranking itself is also *relative* preferences. Recently, *pairwise preference relation-based* [12, 19, 45, 64] and *listwise*-based [74] RecSys have been proposed. Among them, the *pairwise* approach is the most popular, which can be further categorized as *memory-based* methods [12] and *model-based* methods [19, 45, 64]. The *pairwise PR*-based RecSys has the potential to unifying heterogeneous user preferences, and this property, to the best of our knowledge, is first recognized in the present paper. Though the *PR*-based RecSys has the potential to alleviate the cold-start problem, it does not utilize the *side information*.

Recent advances in *PR*-based RecSys [12, 19, 45, 64] and *Conditional Random Fields* (CRF) [78] have made it possible to unify the heterogeneous preferences into the unified *PR* format as well as modeling side information. This observation leads to a natural extension presented in this work to unify the *CRF*-based method with the *PR*-based methods, to complement their strengths.

There exist two similar research topics that should be distinguished from our work. The first one is the *cross-domain* RecSys [60] which considers heterogeneous items, i.e., mixing *movies* and *books*, while our work considers the heterogeneous preferences

associated to the same type of items. The other topic is the *mixture of experts* [75], in which the non-personalized expert opinions from different sources are weighted into the personalized recommendations, while our work considers heterogeneous preferences of the same user distributed across various sources.

5.3 Preference Relation-based Conditional Random Fields

In this section, we propose the *Preference Relation-based Conditional Random Fields* (PrefCRF) to model both the heterogeneous preferences and the side information. The rest of this section defines the *PR*-based RecSys problem, and introduces the concept of the *PrefNMF* [19] that forms our underlying model, followed by a detailed description of the *PrefCRF* and discussion on issues such as feature design, parameter estimation, and predictions.

5.3.1 Problem Statement

Generally, the task of *PR*-based RecSys is to take *PR* as input and output Top-N recommendations. Specifically, let $\pi_{uij} \in \Pi$ encode the *PR* of each user $u \in \mathcal{U}$, and each π_{uij} is defined over an ordered item pair (i, j), denoting $i \prec j$, $i \simeq j$, or $i \succ j$. The main task towards Top-N recommendations is to estimate the value of each unknown $\pi_{uij} \in \Pi_{unknown}$, such that $\hat{\pi}_{uij}$ approximates π_{uij}. This can be considered

as an optimization task that performs directly on the *PR*

$$\hat{\pi}_{uij} = \underset{\hat{\pi}_{uij} \in [0,1]}{\arg\min} (\pi_{uij} - \hat{\pi}_{uij})^2 \tag{5.3.1}$$

However, it can be easier to estimate the $\hat{\pi}_{uij}$ by the difference between two user-wise preferences p_{ui} and p_{uj}, i.e., $\hat{\pi}_{uij} = \phi(\hat{p}_{ui} - \hat{p}_{uj})$, where $\phi(\cdot)$ is a function that bounds the value into $[0, 1]$ and ensures $\phi(0) = 0.5$. For example, the *inverse-logit* function $\phi(x) = \frac{e^x}{1+e^x}$ can be used when user-wise preferences involve large values. The objective of this paper is then to solve the following optimization problem

$$(\hat{p}_{ui}, \hat{p}_{uj}) = \underset{\hat{p}_{ui}, \hat{p}_{uj}}{\arg\min} (\pi_{uij} - \phi(\hat{p}_{ui} - \hat{p}_{uj}))^2 \tag{5.3.2}$$

which optimizes the user-wise preferences directly, and Top-N recommendations can be obtained by simply sorting the estimated user-wise preferences.

5.3.2 Preference Relation-based Matrix Factorization

Matrix Factorization (MF) [41] is a popular *RecSys* approach that has mainly been applied to absolute ratings. Recently, the *PrefNMF* [19] model was proposed to accommodate *PR* input for *MF* models. Like traditional *MF* models, the *PrefNMF* model discovers the latent factor space shared between users and items, where the latent factors describe both the *taste* of users and the *characteristics* of items. The attractiveness of an item to a user is then measured by the inner product of their latent feature vectors.

Formally, each user u is associated with a latent feature vector $\mathbf{u}_u \in \mathbb{R}^k$, and each item i is associated with a latent feature vector $\mathbf{v}_i \in \mathbb{R}^k$, where k is the dimension of the latent factor space. The attractiveness of items i and j to user u are $\mathbf{u}_u^\top \mathbf{v}_i$ and $\mathbf{u}_u^\top \mathbf{v}_j$, respectively. When $\mathbf{u}_u^\top \mathbf{v}_i > \mathbf{u}_u^\top \mathbf{v}_j$, the item i is said to be more preferable

to the user u than item j, i.e., $i \succ j$. The strength of this preference relation π_{uij} can be estimated by $\mathbf{u}_u^\top(\mathbf{v}_i - \mathbf{v}_j)$, and the *inverse-logit* function is applied to ensure $\hat{\pi}_{uij} \in [0, 1]$:

$$\hat{\pi}_{uij} = \frac{e^{\mathbf{u}_u^\top(\mathbf{v}_i - \mathbf{v}_j)}}{1 + e^{\mathbf{u}_u^\top(\mathbf{v}_i - \mathbf{v}_j)}} \tag{5.3.3}$$

The latent feature vectors $\mathbf{u}_u$ and $\mathbf{v}_i$ are learned by minimizing regularized squared error with respect to the set of all known preference relations Π:

$$\min_{\mathbf{u}_u, \mathbf{v}_i \in \mathbb{R}^k} \sum_{\pi_{uij} \in \Pi \wedge (i<j)} (\pi_{uij} - \hat{\pi}_{uij})^2 + \lambda(\|\mathbf{u}_u\|^2 + \|\mathbf{v}_i\|^2) \tag{5.3.4}$$

where λ is the regularization coefficient. The optimization can be done with *Stochastic Gradient Descent* for the favor of speed on sparse data, or with *Alternating Least Squares* in favor of parallelization on dense data.

5.3.3 Conditional Random Fields

Conditional Random Fields (CRF) [78] model a set of random variables having Markov property with respect to an undirected graph $\mathcal{G}$, and each random variable can be conditioned on a set of global observations $\mathbf{o}$. The undirected graph $\mathcal{G}$ consists of a set of vertexes $\mathcal{V}$ connected by a set of edges $\mathcal{E}$ without orientation, where two vertexes are neighboring to each other when connected. Each vertex in $\mathcal{V}$ encodes a random variable, and the Markov property implies that a variable is conditionally independent of others given its neighbors.

In this work, we use *CRF* to model interactions among user-wise preferences conditioned on side information with respect to a set of undirected graphs. Specifically for each user u, there is a graph $\mathcal{G}_u$ with a set of vertexes $\mathcal{V}_u$ and a set of edges $\mathcal{E}_u$. Each vertex in $\mathcal{V}_u$ represents a user-wise preference p_{ui} of user u on the item i. Each

edge in $\mathcal{E}_u$ captures a relation between two preferences by the same user.

Specifically, two preferences are connected by an edge if they are given by the same user. Fig. 5.2 shows an example of two graphs for users u and v. Note that vertexes of different graphs are not directly connected, however, the edges between the same pair of items are associated to the same item-item correlation. For example, the edge between p_{ui} and p_{uj} and the edge between p_{vi} and p_{vj} are associated with the same item-item correlation ψ_{ij} between items i and j.

Each vertex is conditioned on a set of global observations $\mathbf{o}$, which is the *side information* in our context. Specifically, each user u is associated with a set of L attributes $\{\mathbf{o}_u\}^L$ such as *age*, *gender* and *occupation*. Similarly, each item i is associated with a set of M attributes $\{\mathbf{o}_i\}^M$ such as *genres* for movie. Those side information is encoded as the set of global observations $\mathbf{o} = \{\{\mathbf{o}_u\}^L, \{\mathbf{o}_i\}^M\}$.

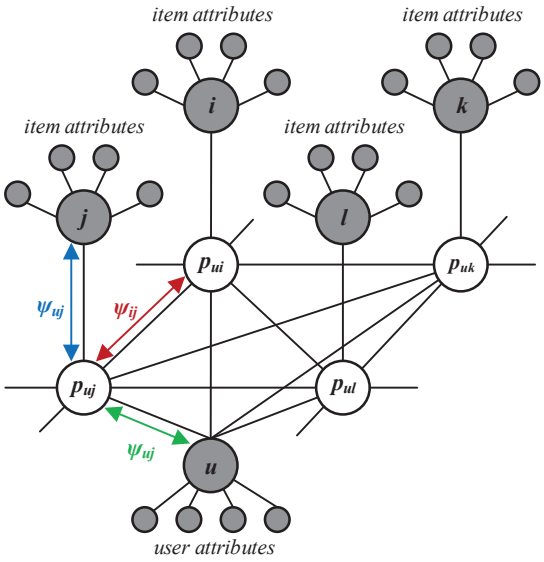
Formally, let $\mathbf{p}_u = \{p_{ui} \mid i \in \mathcal{I}_u\}$ be the joint set of preferences expressed by user u, then we are interested in modeling the conditional distribution $P(\mathbf{p}_u \mid \mathbf{o})$ over the graph $\mathcal{G}_u$.

$$P(\mathbf{p}_u \mid \mathbf{o}) = \frac{1}{Z_u}\Psi_u(\mathbf{p}_u, \mathbf{o}) \tag{5.3.5}$$

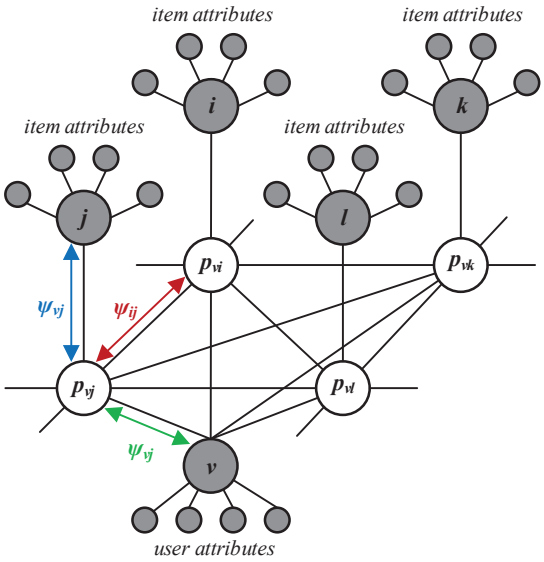
$$\Psi_u(\mathbf{p}_u, \mathbf{o}) = \prod_{(ui)\in\mathcal{V}_u} \psi_{ui}(p_{ui}, \mathbf{o}) \prod_{(ui,uj)\in\mathcal{E}_u} \psi_{ij}(p_{ui}, p_{uj}) \tag{5.3.6}$$

where $Z_u(\mathbf{o})$ is the normalization term that ensures $\sum_{\mathbf{p}_u} P(\mathbf{p}_u \mid \mathbf{o}) = 1$, and $\psi(\cdot)$ is a positive function known as *potential*. The potential $\psi_{ui}(\cdot)$ captures the global observations associated to the user u and the item i, and the potential $\psi_{ij}(\cdot)$ captures the correlations between two preferences p_{ui} and p_{uj}

$$\psi_{ui}(p_{ui}, \mathbf{o}) = \exp\{\mathbf{w}_u^\top \mathbf{f}_u(p_{ui}, \mathbf{o}_i) + \mathbf{w}_i^\top \mathbf{f}_i(p_{ui}, \mathbf{o}_u))\} \tag{5.3.7}$$



(a) Graph of user u



(b) Graph of user v

Figure 5.2: Undirected graphs for users u and v

$$\psi_{ij}(p_{ui}, p_{uj}) = \exp\{w_{ij} f_{ij}(p_{ui}, p_{uj})\} \tag{5.3.8}$$

where $\mathbf{f}_u$, $\mathbf{f}_i$, and f_{ij} are the features to be designed shortly in Section 5.3.5, and $\mathbf{w}_u$, $\mathbf{w}_i$, and w_{ij} are the corresponding weights realizing the importance of each feature. With the weights estimated from data, the unknown preference p_{ui} can be predicted as

$$\hat{p}_{ui} = \arg\max_{p_{ui} \in [-1,1]} P(p_{ui} \mid \mathbf{p}_u, \mathbf{o}) \tag{5.3.9}$$

where $P(p_{ui} \mid \mathbf{p}_u, \mathbf{o})$ measures the prediction confidence.

5.3.4 Ordinal Logistic Regression

The original *PrefNMF* [19] computes the attractiveness of an item to a user as the product of their latent feature vectors which results a scalar value. Instead of point estimation, we wish to have the distributions over ordinal values. Therefore the *Random Utility Models* [55] and the *Ordinal Logistic Regression* [54] are utilized for the conversion.

Random Utility Models [55] assume the existence of a *latent utility* $x_{ui} = \mu_{ui} + \epsilon_{ui}$ that captures how much the user u is interested in the item i, where μ_{ui} captures the interest and ϵ_{ui} is the random noise, and here assumed to follow the logistic distribution [42].

The *Ordinal Logistic Regression* [54] is then used to convert the user-wise preferences p_{ui} into ordinal values, which assumes that the preference p_{ui} is chosen based on the interval to which the latent utility belongs:

$$p_{ui} = l \text{ if } x_{ui} \in (\theta_{l-1}, \theta_l] \text{ and } p_{ui} = L \text{ if } x_{ui} > \theta_{L-1} \tag{5.3.10}$$

where L is the number of ordinal levels and θ_l are the threshold values of interest.

The probability of receiving a preference l is therefore:

$$Q(p_{ui} = l \mid u, i) = \int_{\theta_{l-1}}^{\theta_l} P(x_{ui} \mid \theta)\, \mathrm{d}\theta = F(\theta_l) - F(\theta_{l-1}) \tag{5.3.11}$$

where $F(\theta_l)$ is the cumulative logistic distribution evaluated at θ_l with standard deviation s_{ui}.

$$F(x_{ui} \leq l \mid \theta_l) = \frac{1}{1 + \exp(-\frac{\theta_{uil} - \mu_{ui}}{s_{ui}})} \tag{5.3.12}$$

The thresholds θ_l can be user-specific or item-specific, and this work uses the user-specific parametrization same as in [42]. Then the thresholds θ_{uil} in Eq. 5.3.12 are replaced with a set of user-specific thresholds $\{\theta_{ul}\}_{l=1}^{L}$ for each user u. These thresholds are then estimated from data for each user.

5.3.5 PrefCRF: Unifying PrefNMF and CRF

The *CRF* provides a principled way of capturing both the side information and interactions among preferences. However, it employs the log-linear modeling as shown in Eq. 5.3.6, and therefore does not enable a simple treatment of *PR*. The *PrefNMF*, on the other hand, accepts *PR* but is weak in utilizing side information. The complementary between these two techniques calls for an unified *PrefCRF* model to take all the advantages.

Unification

Essentially, the proposed *PrefCRF* model captures the side information and promotes the agreement between the *PrefNMF* and the *CRF*. Specifically, the *PrefCRF* model combines the item-item correlations (Eq. 5.3.8) and the ordinal distributions $Q(p_{ui} \mid$

u, i) over user-wise preferences obtained from Eq. 5.3.11.

$$P(\mathbf{p}_u \mid \mathbf{o}) \propto \Psi_u(\mathbf{p}_u, \mathbf{o}) \prod_{p_{ui} \in \mathbf{p}_u} Q(p_{ui} \mid u, i) \tag{5.3.13}$$

where Ψ_u is the potential function capturing the side information and interaction among preferences related to user u. Though there is a separated graph for each user, the weights are optimized across all graphs.

Feature Design

A feature is essentially a function f of $n > 1$ arguments that maps the n-dimensional input into the unit interval $f : \mathbb{R}^n \to [0, 1]$. We design the following kinds of features:

Correlation Features The item-item correlation is captured by the feature

$$f_{ij}(p_{ui}, p_{uj}) = g(|(p_{ui} - \bar{p}_i) - (p_{uj} - \bar{p}_j)|) \tag{5.3.14}$$

where $g(\alpha)$ normalizes feature values and α plays the role of deviation, and $\bar{p}_i$ and $\bar{p}_j$ are the average user-wise preference for items i and j, respectively. This correlation feature captures the intuition that correlated items should be ranked similarly by the same user after offsetting the goodness of each item.

Attribute Features Each user u and item i has a set of attributes $\mathbf{o}_u$ and $\mathbf{o}_i$, respectively. These attributes are mapped to preferences by the following features

$$\begin{aligned} \mathbf{f}_i(p_{ui}) &= \mathbf{o}_u g(|(p_{ui} - \bar{p}_i)|) \\ \mathbf{f}_u(p_{ui}) &= \mathbf{o}_i g(|(p_{ui} - \bar{p}_u)|) \end{aligned} \tag{5.3.15}$$

where $\mathbf{f}_i$ models which users like the item i and $\mathbf{f}_u$ models which classes of items the user u likes.

Since one correlation feature exists for each pair of co-rated items, the number of correlation features can be large, and makes the estimation slow to converge and less robust. Therefore, we only keep strong correlation features $\mathbf{f}_{\text{strong}}$ extracted based on the *Pearson* correlation between items using a user-specified *minimum correlation threshold*. The correlations are computed using user-wise preferences generated from *PR*.

Parameter Estimation

In general, *CRF* models cannot be determined by standard maximum likelihood estimations, instead, approximation techniques are used in practice. This study employs the pseudo-likelihood [8] to estimate parameters by maximizing the regularized sum of log local likelihoods:

$$log\mathcal{L}(\mathbf{w}) = \sum_{p_{ui} \in \Pi} \log P(p_{ui} \mid \mathbf{p}_u, \mathbf{o}) - \frac{1}{2\sigma^2}\mathbf{w}^\top \mathbf{w} \tag{5.3.16}$$

where $\mathbf{w}$ are the weights and $1/2\sigma^2$ controls the regularization. To make the notation uncluttered, we write $\mathbf{p}_u$ instead of explicitly as $\mathbf{p}_u \backslash p_{ui}$.

The local likelihood in Eq. 5.3.16 is defined as:

$$P(p_{ui}|\mathbf{p}_u, \mathbf{o}) = \frac{1}{Z_{ui}} Q(p_{ui}|u, i)\psi_{ui}(p_{ui}, \mathbf{o}) \prod_{p_{uj} \in \mathbf{p}_u} \psi_{ij}(p_{ui}, p_{uj}) \tag{5.3.17}$$

where $Z_{ui}(\mathbf{o})$ is the normalization term:

$$Z_{ui} = \sum_{p_{ui}=l_{min}}^{l_{max}} Q(p_{ui} \mid u, i)\psi_{ui}(p_{ui}, \mathbf{o}) \prod_{p_{uj} \in \mathbf{p}_u} \psi_{ij}(p_{ui}, p_{uj}) \tag{5.3.18}$$

where l_{min} is the first and l_{max} is the last interval, i.e., 1 and 3 in our settings.

To optimize the parameters, we use the stochastic gradient ascent procedure that updates the parameters by passing through the set of ratings of each user:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta \nabla log\mathcal{L}(\mathbf{w}) \tag{5.3.19}$$

where η is the learning rate. More specifically, for each p_{ui} we update the attribute weights $\mathbf{w}_o = \{\mathbf{w}_u, \mathbf{w}_i\}$ and correlation weight w_{ij} for each neighbor $p_{uj} \in \mathbf{p}_u$ using the gradient of the log pseudo-likelihood

$$\frac{\partial log\mathcal{L}}{\partial \mathbf{w}_o} = \mathbf{f}_o(p_{ui}, \mathbf{o}) - \sum_{p_{ui}=l_{min}}^{l_{max}} P(p_{ui} \mid \mathbf{p}_u, \mathbf{o}) \mathbf{f}_o(p_{ui}, \mathbf{o}) - \frac{w_i}{\sigma^2} \tag{5.3.20}$$

$$\frac{\partial log\mathcal{L}}{\partial w_{ij}} = f_{ij}(p_{ui}, p_{uj}) - \sum_{p_{ui}=l_{min}}^{l_{max}} P(p_{ui} \mid \mathbf{p}_u, \mathbf{o}) f_{ij}(p_{ui}, p_{uj}) - \frac{w_{ij}}{\sigma^2} \tag{5.3.21}$$

Item Recommendation

The ultimate goal of *RecSys* is often to rank the items and recommend the Top-N items to the user. As the input of *PrefCRF* is the preference relations, the final output will be item rankings instead of ratings.

The *PrefCRF* produces distributions over the user-wise preferences, which can be converted into point estimates by computing the expectation

$$\hat{p}_{ui} = \sum_{p_{ui}=l_{min}}^{l_{max}} p_{ui} P(p_{ui} \mid \mathbf{p}_u, \mathbf{o}) \tag{5.3.22}$$

where l refers to the intervals of user-wise preferences: from the least to the most preferred.

Given the predicted user-wise preferences, the items can be sorted and ranked accordingly. Alg. 3 summarizes the procedures of *PrefCRF*.

Algorithm 3 *PrefCRF* Algorithm

Input: Heterogeneous preferences from sources.

Step 1: Infer and merge *PR*.

Step 2: Predict each user-wise preferences $\hat{p}_{ui}$ using Eq. 5.3.3 and Eq. 5.2.2.

Step 3: Predict distributions of $\hat{p}_{ui}$ with Eq. 5.3.11.

Step 4: Repeat

for each $u \in \mathcal{U}$ **do**

for each $p_{ui} \in \mathbf{p}_u$ **do**

Get local likelihood using Eq. 5.3.17

Get attribute feature $\mathbf{f}_v$ using Eq. 5.3.15

Get attribute feature gradients using Eq. 5.3.20

Update $\mathbf{w}_o$ with gradients using Eq. 5.3.19

for each $p_{uj} \in \mathbf{p}_u$, $i \neq j \wedge f_{ij} \in \mathbf{f}_{\text{strong}}$ **do**

Get correlation feature f_{ij} using Eq. 5.3.14

Get corr. feature gradient using Eq. 5.3.21

Update w_{ij} with gradient using Eq. 5.3.19

end for

end for

end for

Until stopping criteria met

Predictions:

* Predict user-wise preferences using Eq. 5.3.22.

* Sort and select the Top-N items.

Computational Complexity

We perform the computational complexity analysis on the *PrefCRF* and its underlying *PrefNMF* algorithms. Given n users and m items each with d_u and d_i preferences, respectively. Let us temporarily ignore the user-specified latent factors. Then the complexity of both *PrefNMF* and *PrefCRF* is $O(nd_u^2)$. However, in practice few item co-rated by the same user are strong neighbors of each other due to the correlation threshold defined in Section 5.3.5. As a result, the computation time of *PrefCRF* tends to be $O(nd_uc)$ where c is a factor of correlation threshold.

5.4 Experiment and Analysis

To study the performance of the proposed *PrefCRF* model, comparisons were done with the following representative algorithms: *KNN* [65], *NMF* [41], *PrefKNN* [12], and *PrefNMF* [19]. We implemented *PrefCRF* as well as the compared algorithms according to their original papers.

5.4.1 Experimental Settings

Datasets and Experiment Design

Experiments are conducted on four public datasets: *MovieLens*-1M [3], *Amazon Movie Reviews* [4], *EachMovie* [5], and *MovieLens*-20M [3]. These datasets or their subsets are transformed to simulate four scenarios of heterogeneous data:

[3] http://grouplens.org/datasets/movielens
[4] http://snap.stanford.edu/data/web-Movies.html
[5] http://grouplens.org/datasets/eachmovie

Side Information The impact of side information is studied on the *MovieLens*-1M dataset which provides *gender*, *age*, and *occupation* information about users and *genres* of movies. The dataset contains more than 1 million ratings by $6,040$ users on $3,900$ movies. For a reliable comparison, the dataset is split into training and test sets with different sparsities, similar to related work [80, 83]. Specifically, for each user we randomly select $N = 30$, 40, 50 and 60 ratings for training, and put the rest for testing. To ensure that each user has at least 10 movies for testing, users with less than 40, 50, 60 or 70 ratings are removed.

Different Forms *Amazon Movie Reviews* dataset contains two forms of preferences: *textual reviews* and 5-star *ratings*. We extracted a dense subset by randomly selecting 5141 items with at least 60 reviews for each, and 2000 users with at least 60 movies reviews for each, and this results in 271K ratings. For each user, 50 random reviews are selected for training, and the rest are put aside for testing. The training set is further split into half ratings and half textual reviews. Rating-based models are trained on the ratings only, where *PR*-based models utilize textual reviews as well.

Different Scales *EachMovie* dataset contains ratings in 6-star scale that can be easily converted into binary scale, i.e., ratings $1-3$ and $4-6$ are mapped to 0 and 1 respectively. We extract a subset by randomly selecting 3000 users who have rated at least 70 items as a dense dataset is required for splitting. The resultant dataset contains 120K ratings on 1495 items. For each user we randomly select 60 ratings for training and leave the rest for testing, and half of the ratings in the training set are mapped into binary scale. Rating-based models are trained on the 6-star ratings while *PR*-based models will exploit the

binary ratings as well.

Different Biases We study the impact of biases by adding biases into a stable dataset with minimal existing biases. To prepare such dataset we extract a stable subset from the latest *MovieLens*-20M released on April-2015. Specifically, 258K ratings by 2020 users on 4408 movies released between 2010 and 2015 are extracted, where each user has rated at least 60 ratings. Biases are then introduced by adding a different *Laplace noise* sampled from $Laplace(0, b)$ to each user and item.

For *PR*-based methods, the same conversion method as in [19] is used to converted ratings into *PR*. For example, 1, 0 and 0.5 are assigned to the preference relation π_{uij} when $p_{ui} > p_{uj}$, $p_{ui} < p_{uj}$, and $p_{ui} = p_{uj}$, respectively.

Evaluation Metrics

Traditional recommender systems aim to optimize *RMSE* or *MAE* which emphasizes on absolute ratings. However, the ultimate goal of recommender systems is usually to obtain the ranking of items [42], where good performance on *RMSE* or *MAE* may not be translated into good ranking results [42]. Therefore, we employ two evaluation metrics *Normalized Cumulative Discounted Gain*@T (NDCG@T) [34] that is popular in academia, and *Mean Average Precision*@T (MAP@T) [13] that is common in contests. The NDCG@T metric is defined as

$$\text{NDCG@}T = \frac{1}{K(T)} \sum_{t=1}^{T} \frac{2^{r_t} - 1}{\log_2 (t+1)} \tag{5.4.1}$$

where r_t is the relevance judgment of the item at position t, and $K(T)$ is the normalization constant. The MAP@T metric is defined as

$$\text{MAP@}T = \frac{1}{|\mathcal{U}_{test}|} \sum_{u \in \mathcal{U}_{test}} \sum_{t=1}^{T} \frac{P_u(t)}{min(m_u, t)} \tag{5.4.2}$$

where m_u is the number of relevant items to user u, and $P_u(t)$ is user u's precision at position t. Both metrics are normalized to $[0, 1]$ with higher value indicates better performance.

These metrics, together with other ranking-based metrics, require a set of relevant items to be defined in the test set such that the predicted rankings can be evaluated against. In this paper, we follow the same heuristics as in the related work [12, 63] and consider items with the highest ratings as relevant.

Parameter Setting

For a fair comparison, we fix the number of latent factors to 50 for all algorithms, which is the same setting as used in [63].The number of neighbors for *KNN* algorithms is set to 50. We vary the minimum correlation threshold for the *PrefCRF* to examine the performance with different number of features. Different values of regularization coefficient are also tested.

5.4.2 Results and Analysis

Algorithms are compared on four heterogeneous scenarios: *side information*, *different forms*, *different scales* and *different biases*. The impact of sparsity levels and parameters is studied on the *MovieLens*-1M dataset, while these settings for other experiments are fixed. Each experiment is repeated ten times with different random

seeds and we report the mean results with standard deviations. For each experiment, we also performed a paired t-test (two-tailed) with a significance level of 95% on the best and the second best results, and all p-values are less than 1×10^{-5}.

Fusing Side Information

Table 5.1 shows the *NDCG* and *MAP* metrics on Top-N recommendation tasks by compared algorithms. It can be observed that the proposed *PrefCRF*, which captures the side information, consistently outperforms others. To confirm the improvement, we plot the results in Fig. 5.3(b) by varying the position T. The figure shows that *PrefCRF* not only outperforms others but has a strong emphasize on top items, i.e., $T < 5$.

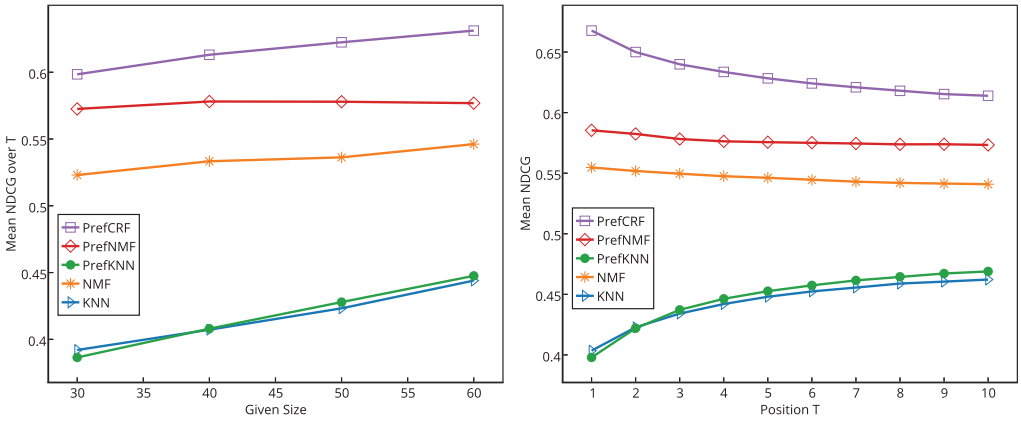
The impact of sparsity is investigated by plotting the results against sparsity levels as in Fig. 5.3(a). We can observe that the performance of *PrefCRF* increases linearly given more training data, while its underlying *PrefNMF* model is less extensible to denser dataset. One possible reason is that incorporating side information has extended the modeling capability of the model, resulting better utilization of more data.

Fusing Preferences in Different Forms

In this experiment, we first converted textual reviews into negative (-1), neutral (0), and positive (1) values using the *NLTK* library [10], and then converted them into *PR*. We study how these additional information can assist *PR*-based methods, and results over ten runs are shown in Table 5.2. Surprisingly, the performance of all *PR*-based methods except *PrefCRF* have decreased by incorporating textual reviews.

Table 5.1: Results over ten runs on *MovieLens*-1M dataset.

	Given 30				Given 40			
Algorithm	NDCG@1	NDCG@10	MAP@1	MAP@10	NDCG@1	NDCG@10	MAP@1	MAP@10
UserKNN	0.4306 ± 0.0011	0.4081 ± 0.0029	0.3539 ± 0.0071	0.2744 ± 0.0025	0.3695 ± 0.0048	0.4252 ± 0.0036	0.3663 ± 0.0047	0.2877 ± 0.0034
NMF	0.5274 ± 0.0084	0.5195 ± 0.0040	0.5225 ± 0.0081	0.3549 ± 0.0037	0.5424 ± 0.0067	0.5291 ± 0.0034	0.5377 ± 0.0066	0.3631 ± 0.0035
PrefKNN	0.3462 ± 0.0073	0.4048 ± 0.0038	0.3430 ± 0.0072	0.2720 ± 0.0037	0.3651 ± 0.0065	0.4283 ± 0.0024	0.3620 ± 0.0063	0.2904 ± 0.0023
PrefNMF	0.5778 ± 0.0112	0.5680 ± 0.0041	0.5724 ± 0.0109	0.3992 ± 0.0033	0.5883 ± 0.0073	0.5732 ± 0.0028	0.5832 ± 0.0073	0.4019 ± 0.0032
PrefCRF	$\mathbf{0.6206 \pm 0.0076}$	$\mathbf{0.5856 \pm 0.0028}$	$\mathbf{0.6150 \pm 0.0073}$	$\mathbf{0.4195 \pm 0.0028}$	$\mathbf{0.6395 \pm 0.0064}$	$\mathbf{0.5990 \pm 0.0023}$	$\mathbf{0.6340 \pm 0.0062}$	$\mathbf{0.4294 \pm 0.0021}$
	Given 50				Given 60			
Algorithm	NDCG@1	NDCG@10	MAP@1	MAP@10	NDCG@1	NDCG@10	MAP@1	MAP@10
UserKNN	0.3831 ± 0.0063	0.4424 ± 0.0027	0.3803 ± 0.0060	0.3015 ± 0.0026	0.4035 ± 0.0090	0.4622 ± 0.0035	0.4002 ± 0.0085	0.3163 ± 0.0027
NMF	0.5430 ± 0.0083	0.5326 ± 0.0036	0.5390 ± 0.0082	0.3669 ± 0.0025	0.5547 ± 0.0109	0.5409 ± 0.0063	0.5504 ± 0.0113	0.3734 ± 0.0055
PrefKNN	0.3831 ± 0.0092	0.4483 ± 0.0030	0.3803 ± 0.0089	0.3070 ± 0.0022	0.3979 ± 0.0075	0.4689 ± 0.0039	0.3948 ± 0.0069	0.3223 ± 0.0033
PrefNMF	0.5873 ± 0.0096	0.5745 ± 0.0035	0.5830 ± 0.0098	0.4019 ± 0.0033	0.5854 ± 0.0145	0.5733 ± 0.0048	0.5808 ± 0.0142	0.4007 ± 0.0037
PrefCRF	$\mathbf{0.6548 \pm 0.0055}$	$\mathbf{0.6068 \pm 0.0018}$	$\mathbf{0.6499 \pm 0.0059}$	$\mathbf{0.4372 \pm 0.0024}$	$\mathbf{0.6677 \pm 0.0074}$	$\mathbf{0.6139 \pm 0.0018}$	$\mathbf{0.6625 \pm 0.0072}$	$\mathbf{0.4436 \pm 0.0016}$



(a) NDCG VS. Sparsity (b) NDCG@T (Given 60)

Figure 5.3: Varying sparsity on *MovieLens*-1M dataset.

We suspect that this is due to the misclassification errors introduced by sentiment classification on text. Indeed, converting preferences in different forms into *PR* can be accomplished in different ways, and may require domain knowledge fall beyond the scope of this paper. However, in the next subsection we will see that an accurate conversion can actually improve the performance.

Table 5.2: Results over ten runs on *Amazon* dataset.

	Ratings		Ratings + Textual Reviews	
Algorithm	NDCG@10	MAP@10	NDCG@10	MAP@10
UserKNN	0.6244 ± 0.0040	0.4599 ± 0.0035	0.6244 ± 0.0037	0.4599 ± 0.0025
NMF	$\mathbf{0.8073 \pm 0.0040}$	$\mathbf{0.6689 \pm 0.0038}$	$\mathbf{0.8073 \pm 0.0041}$	$\mathbf{0.6689 \pm 0.0000}$
PrefKNN	0.6410 ± 0.0038	0.4690 ± 0.0029	0.5765 ± 0.0039	0.4083 ± 0.0029
PrefNMF	0.7495 ± 0.0040	0.5924 ± 0.0031	0.7377 ± 0.0030	0.5806 ± 0.0031
PrefCRF	$\mathbf{0.8223 \pm 0.0033}$	$\mathbf{0.6813 \pm 0.0027}$	$\mathbf{0.8259 \pm 0.0035}$	$\mathbf{0.6890 \pm 0.0026}$

Fusing Preferences in Different Scales

In this experiment preferences in different scales are fused into *PR* to boost the performance of *PR*-based methods. The binary scale ratings are similar to the positive/negative textual reviews, however without incorrect values introduced by text classification.

Table 5.3: Results over ten runs on *EachMovie* dataset.

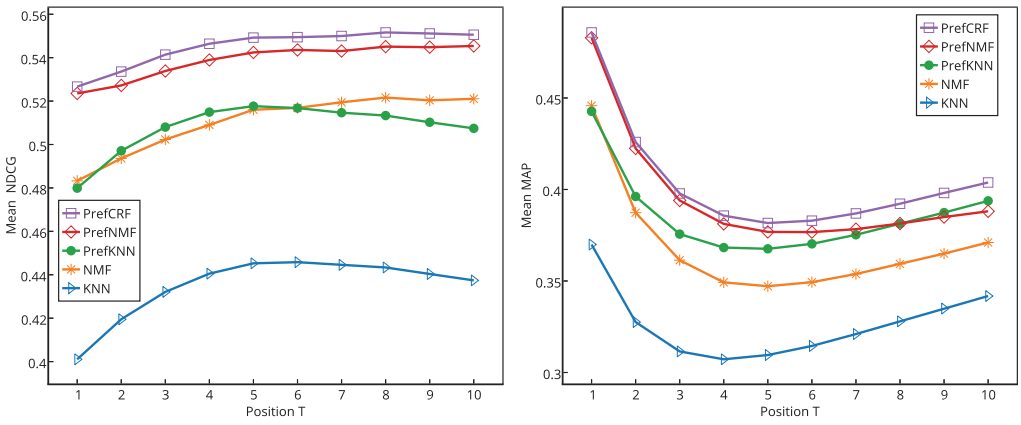
	6-star Ratings		6-star Ratings + Binary Ratings	
Algorithm	NDCG@10	MAP@10	NDCG@10	MAP@10
UserKNN	0.4374 ± 0.0047	0.3418 ± 0.0029	0.4374 ± 0.0047	0.3418 ± 0.0029
NMF	0.5211 ± 0.0078	0.3710 ± 0.0034	0.5211 ± 0.0078	0.3710 ± 0.0034
PrefKNN	0.4908 ± 0.0070	0.3793 ± 0.0031	0.5074 ± 0.0061	0.3938 ± 0.0044
PrefNMF	0.5233 ± 0.0061	0.3820 ± 0.0033	0.5454 ± 0.0060	0.3881 ± 0.0032
PrefCRF	$\mathbf{0.5439 \pm 0.0056}$	$\mathbf{0.4006 \pm 0.0045}$	$\mathbf{0.5506 \pm 0.0053}$	$\mathbf{0.4038 \pm 0.0043}$

Table 5.3 shows the performance of each method and the impact of position T is illustrated in Fig. 5.4. From the table, we can observe that the performance of all *PR*-based methods has increased by incorporating additional binary ratings, while the performance of rating-based methods remains the same.

Fusing Preferences with Different Biases

In this experiment we investigate the impact of different biases, particularly the user-wise and item-wise biases, which are sampled from $Laplace(0, b)$ for each user and each item. Results for unbiased and biased datasets are reported in Table 5.4.

For user-wise biases, we can see that the performance of rating-based methods



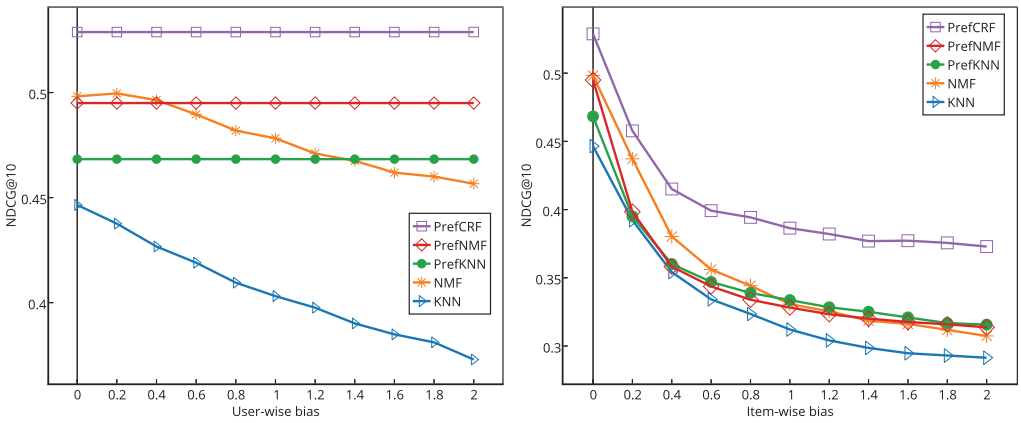
(a) NDCG@T (b) MAP@T

Figure 5.4: Varying position T on *EachMovie* dataset.

Table 5.4: NDCG@10 on *MovieLens*-20M dataset.

Algorithm	Bias = None	User-bias = $Laplace(0, 2)$	Item-bias = $Laplace(0, 2)$
UserKNN	0.4465 ± 0.0033	0.3729 ± 0.0033	0.2914 ± 0.0017
NMF	0.4982 ± 0.0034	0.4566 ± 0.0032	0.3074 ± 0.0019
PrefKNN	$\mathbf{0.4683 \pm 0.0027}$	$\mathbf{0.4683 \pm 0.0027}$	0.3157 ± 0.0021
PrefNMF	$\mathbf{0.4950 \pm 0.0035}$	$\mathbf{0.4950 \pm 0.0035}$	0.3137 ± 0.0017
PrefCRF	$\mathbf{0.5288 \pm 0.0037}$	$\mathbf{0.5288 \pm 0.0037}$	0.3729 ± 0.0023

has decreased while *PR*-based methods are unaffected by such biases. This is further illustrated in Fig. 5.5(a). For item-wise biases, the performance of all methods has decreased, however, to different extent. Fig. 5.5(b) further shows the better resistance of *PrefCRF* to item-wise biases.



(a) NDCG@10 (b) NDCG@10

Figure 5.5: Varying biases on *MovieLens*-20M dataset.

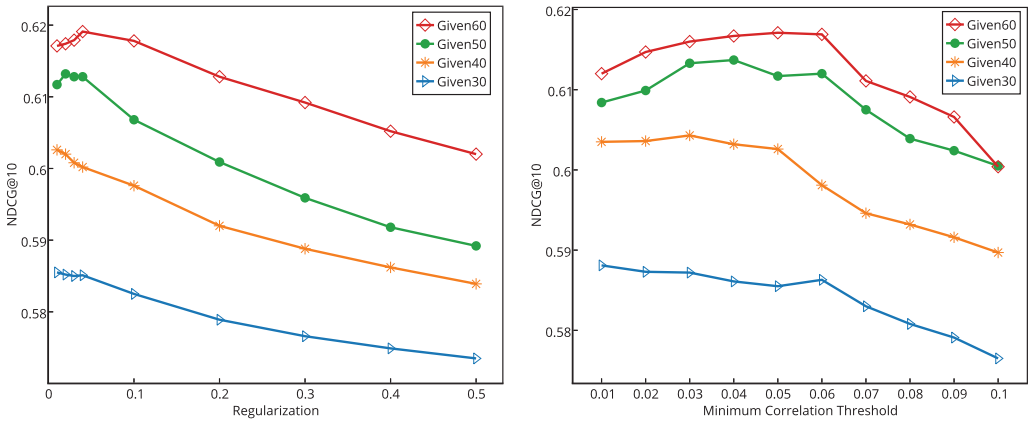
Impact of Regularization and Correlation Threshold

The proposed *PrefCRF* method has two user specified parameters: the *regularization coefficient* and a *minimum correlation threshold* that controls the number of correlation features. We examine the impact of these parameters on the *MovieLens*-1M dataset and the results are plotted in Fig. 5.6.

For the regularization, we can see from Fig. 5.6(a) that the performance gets better when a small regularization penalty applies. In other words, *PrefCRF* can generalize reasonable well without too much regularization. One reason is that the model has already been regularized by its underlying *PrefNMF* model. Another reason is that

the weights of item-item correlations are not user-specific but shared by all users, thus they cannot over-fit every user perfectly.

For the correlation threshold, Fig. 5.6(b) shows that a smaller threshold results better performance by including more correlation features, however, at the cost of more training time and more training data.



(a) NDCG@10 (b) NDCG@10

Figure 5.6: Varying parameters on *MovieLens*-1M.

5.5 Summary

In this chapter we proposed the *PrefCRF* model, which takes advantages of both the representational power of the *CRF* and the ease of modeling *PR* by the *PrefNMF*. To the best of our knowledge, this is the first study on unifying *heterogeneous user preferences* and *heterogeneous side information*. Experiment results on four public datasets demonstrate that various heterogeneous data have been properly handled by

PrefCRF, and significantly improved Top-N recommendation performance has been achieved.

For future work, the computation efficiency of *PR*-based methods can be further improved given that the number of *PR* is usually much larger than ratings. Parallelization is feasible as as each user has a separated set of *PR* that can be processed simultaneously.

Chapter 6

Conclusion and Future Work

The research presented in this thesis consists of two parts: the first part focuses on the *local and global structure* and the *side information* issues and while the second part focuses on the *heterogeneous data source* issue. Several new research problems have been identified together with solutions. This chapter summarizes the research results and the main contributions of this thesis.

6.1 Contributions

Theoretical and experimental results have led to the conclusion and main contribution of this thesis:

- The *Ordinal Random Fields* (ORF) model is proposed to capture both the local and global structures of ordinal user preferences. Through this novel method, both structures are properly captured, resulting improved recommendation quality. *ORF is one of first attempts on modeling both structures for ordinal user preferences.*

- The *Preference Relation-based Markov Random Fields* (PrefMRF) model is proposed to capture both the local and global structures of *Preference Relations*. Due to the flexibility of previous preference relation-based models, only one type of structure can be modeled at a time. Through our novel method, both structures can be modeled, resulting significantly improved recommendation performance. *PrefMRF is the first preference relation-based model that captures both types of structures.*

- The *Preference Relation-based Conditional Random Fields* (PrefCRF) model is proposed to incorporate side information such as item content and user profiles. *PrefCRF is the first preference relation-based model that can incorporate side information in a principled way.*

- The preference relation-based models proposed in this thesis is applied to resolve the heterogeneous data sources problem. This is the first time this problem is spotted and we provide effective solutions to unify heterogeneous data. By exploiting multiple heterogeneous data sources help to alleviate the cold-start problem in which limited data is provided by each data source.

6.2 Future Work

Although the proposed methods have addressed the aforementioned three issues to a certain extend, there remains several problems that need to be addressed in the future. We summarize these follows:

- *Parallelization of Preference Relation-based Models*: The number of preference

relations are usually much larger than the number of traditional ratings. As a result, the modeling process is often slower than rating-based methods. However, we observe that the preference relations of each user are kind of independent from other users'. Therefore, it it possible to parallelize the modeling process for each user to achieve faster modeling process.

- *Identifying Key Preference Relations*: While there are so many possible preference relations, not all of them are important in making recommendations. It remains unknown how to measure the importance of each preference relation to the recommendation quality. If this information can be obtained in future work, then the number of preference relations to be used in modeling can be significantly reduced while the recommendation quality is preserved.

Bibliography

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

[2] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.

[3] G. Adomavicius and J. Zhang. On the stability of recommendation algorithms. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 47–54. ACM, 2010.

[4] G. Adomavicius and J. Zhang. Stability of recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 30(4):23, 2012.

[5] M. Balabanović and Y. Shoham. Fab: content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, 1997.

[6] J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *ICML'04*, pages 65–72. ACM, 2004.

[7] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[8] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B*, 36(2):192–236, 1974.

[9] D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *ICML*, volume 98, pages 46–54, 1998.

[10] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python*. O'Reilly Media, Inc., 2009.

[11] K. Bradley and B. Smyth. Improving recommendation diversity. In *Proceedings of the Twelfth National Conference in Artificial Intelligence and Cognitive Science (AICS-01)*, pages 75–84. Citeseer, 2001.

[12] A. Brun, A. Hamad, O. Buffet, and A. Boyer. Towards preference relations in recommender systems. In *Preference Learning (PL 2010) ECML/PKDD*, 2010.

[13] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *CIKM'09*, pages 621–630. ACM, 2009.

[14] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *J. Artif. Int. Res.*, 10(1):243–270, May 1999.

[15] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys'10*, pages 39–46. ACM, 2010.

[16] A. Defazio and T. Caetano. A graphical model formulation of collaborative filtering neighbourhood methods with fast maximum entropy training. In *ICML'12*, pages 265–272, 2012.

[17] M. S. Desarkar and S. Sarkar. Rating prediction using preference relations based matrix factorization. In *UMAP Workshops*, 2012.

[18] M. S. Desarkar, S. Sarkar, and P. Mitra. Aggregating preference graphs for collaborative rating prediction. In *RecSys'10*, pages 21–28. ACM, 2010.

[19] M. S. Desarkar, R. Saxena, and S. Sarkar. Preference relation based matrix factorization for recommender systems. In *UMAP'12*, pages 63–75. Springer, 2012.

[20] D. Fleder and K. Hosanagar. Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. *Management science*, 55(5):697–712, 2009.

[21] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of machine learning research*, 4:933–969, 2003.

[22] J. Fürnkranz and E. Hüllermeier. Pairwise preference learning and ranking. In *Machine Learning: ECML'03*, pages 145–156. Springer, 2003.

[23] J. Fürnkranz and E. Hüllermeier. *Preference learning*. Springer, 2010.

[24] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *2010 IEEE International Conference on Data Mining*, pages 176–185. IEEE, 2010.

[25] M. Ge, C. Delgado-Battenfeld, and D. Jannach. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 257–260. ACM, 2010.

[26] P. J. Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

[27] I. Guy, I. Ronen, and A. Raviv. Personalized activity streams: sifting through the river of news. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 181–188. ACM, 2011.

[28] I. Guy, N. Zwerdling, D. Carmel, I. Ronen, E. Uziel, S. Yogev, and S. Ofek-Koifman. Personalized recommendation of social software items based on social

relations. In *Proceedings of the third ACM conference on Recommender systems*, pages 53–60. ACM, 2009.

[29] J. Han, M. Kamber, and J. Pei. *Data mining: concepts and techniques*. Morgan kaufmann, 2006.

[30] G. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[31] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *IJCAI*, volume 99, pages 688–693, 1999.

[32] N. Houlsby, J. M. Hernandez-lobato, and Z. Ghahramani. Cold-start active learning with robust ordinal matrix factorization. In *ICML'14*, pages 766–774, 2014.

[33] E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16):1897–1916, 2008.

[34] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM TOIS*, 20(4):422–446, 2002.

[35] R. Jin, L. Si, and C. Zhai. Preference-based graphic models for collaborative filtering. In *UAI'02*, pages 329–336. Morgan Kaufmann Publishers Inc., 2002.

[36] N. Jones, A. Brun, and A. Boyer. Comparisons instead of ratings: Towards more stable preferences. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on*, volume 1, pages 451–456. IEEE, 2011.

[37] B. P. Knijnenburg, M. C. Willemsen, Z. Gantner, H. Soncu, and C. Newell. Explaining the user experience of recommender systems. *User Modeling and User-Adapted Interaction*, 22(4-5):441–504, 2012.

[38] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD'08*, pages 426–434. ACM, 2008.

[39] Y. Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456. ACM, 2009.

[40] Y. Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 53(4):89–97, 2010.

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

[42] Y. Koren and J. Sill. Ordrec: an ordinal model for predicting personalized item rating distributions. In *RecSys'11*, pages 117–124. ACM, 2011.

[43] J. D. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *ICML'01*, pages 282–289, 2001.

[44] X. Li, G. Xu, E. Chen, and Y. Zong. Learning recency based comparative choice towards point-of-interest recommendation. *Expert Systems with Applications*, 42(9):4274–4283, 2015.

[45] N. N. Liu, M. Zhao, and Q. Yang. Probabilistic latent preference analysis for collaborative filtering. In *CIKM'09*, pages 759–766. ACM, 2009.

[46] S. Liu, T. Tran, G. Li, and Y. Jiang. Ordinal random fields for recommender systems. In *ACML'14*, pages 283–298. JMLR: workshop and conference proceedings, 2014.

[47] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. Springer, 2011.

[48] L. Lü and W. Liu. Information filtering via preferential diffusion. *Physical Review E*, 83(6):066119, 2011.

[49] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou. Recommender systems. *Physics Reports*, 519(1):1–49, 2012.

[50] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *WSDM'11*, pages 287–296. ACM, 2011.

[51] R. D. Mare. Social background and school continuation decisions. *Journal of the American Statistical Association*, 75(370):295–305, 1980.

[52] B. M. Marlin. Modeling user rating profiles for collaborative filtering. In *NIPS'03*. MIT Press, 2003.

[53] P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. In *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, pages 492–508. Springer, 2004.

[54] P. McCullagh. Regression models for ordinal data. *Journal of the Royal Statistical Society, Series B*, 42(2):109–142, 1980.

[55] D. McFadden. Econometric models for probabilistic choice among products. *Journal of Business*, 53(3):S13–S29, 1980.

[56] S. M. McNee, J. Riedl, and J. A. Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI EA'06*, pages 1097–1101. ACM, 2006.

[57] K. Miyahara and M. J. Pazzani. Collaborative filtering with the simple bayesian classifier. In *PRICAI 2000 Topics in Artificial Intelligence*, pages 679–689. Springer, 2000.

[58] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2012.

[59] A. Nakamura and N. Abe. Collaborative filtering using weighted majority prediction algorithms. In *ICML*, volume 98, pages 395–403, 1998.

[60] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[61] U. Paquet, B. Thomson, and O. Winther. A hierarchical model for ordinal matrix factorization. *Statistics and Computing*, 22(4):945–957, 2012.

[62] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[63] Y. Ren, G. Li, and W. Zhou. Learning rating patterns for top-n recommendations. In *ASONAM'12*, pages 472–479. IEEE Computer Society, 2012.

[64] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

[65] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. An open architecture for collaborative filtering of netnews. In *CSCW'94*, pages 175–186. ACM, 1994.

[66] F. Ricci, L. Rokach, and B. Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

[67] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of*. Addison-Wesley, 1989.

[68] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document, 2000.

[69] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW'10*, pages 285–295. ACM, 2001.

[70] B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the fifth international conference on computer and information technology*, volume 1. Citeseer, 2002.

[71] D. L. Schacter and C. S. Dodson. Misattribution, false recognition and the sins of memory. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 356(1413):1385–1393, 2001.

[72] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR'02*, pages 253–260. ACM, 2002.

[73] A. Sharma and B. Yan. Pairwise learning in recommendation: experiments with community recommendation on linkedin. In *RecSys'13*, pages 193–200. ACM, 2013.

[74] Y. Shi, M. Larson, and A. Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *RecSys'10*, pages 269–272. ACM, 2010.

[75] X. Su, R. Greiner, T. M. Khoshgoftaar, and X. Zhu. Hybrid collaborative filtering algorithms using a mixture of experts. In *WI'07*, pages 645–649. IEEE, 2007.

[76] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Scalable collaborative filtering approaches for large recommender systems. *The Journal of Machine Learning Research*, 10:623–656, 2009.

[77] T. Tran, D. Phung, and S. Venkatesh. Collaborative filtering via sparse markov random fields. *arXiv preprint arXiv:1602.02842*, 2016.

[78] T. Tran, D. Q. Phung, and S. Venkatesh. Preference networks: Probabilistic models for recommendation systems. In *AusDM'07*, pages 195–202. ACS, 2007.

[79] T. Tran, D. Q. Phung, and S. Venkatesh. Ordinal boltzmann machines for collaborative filtering. In *UAI'09*, pages 548–556. AUAI Press, 2009.

[80] T. Tran, D. Q. Phung, and S. Venkatesh. Probabilistic models over ordered partitions with applications in document ranking and collaborative filtering. In *SDM'11*, pages 426–437. SIAM, 2011.

[81] T. Tran, D. Q. Phung, and S. Venkatesh. Learning from ordered sets and applications in collaborative ranking. In *ACML'12*, pages 427–442. JMLR: workshop and conference proceedings, 2012.

[82] T. Tran, D. Q. Phung, and S. Venkatesh. A sequential decision approach to ordinal preferences in recommender systems. In *AAAI'12*, 2012.

[83] M. Weimer, A. Karatzoglou, Q. V. Le, and A. Smola. Maximum margin matrix factorization for collaborative ranking. In *NIPS'07*, pages 1593–1600, 2007.

[84] T. Zhou, L.-L. Jiang, R.-Q. Su, and Y.-C. Zhang. Effect of initial configuration on network-based recommendation. *EPL (Europhysics Letters)*, 81(5):58004, 2008.

[85] T. Zhou, J. Ren, M. Medo, and Y.-C. Zhang. Bipartite network projection and personal recommendation. *Physical Review E*, 76(4):046115, 2007.

[86] A. Zimdars, D. M. Chickering, and C. Meek. Using temporal data for making recommendations. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 580–588. Morgan Kaufmann Publishers Inc., 2001.