Hierarchical Information and Data Modeling for Neural-based Recommender Systems

Author:
Yi, Peng

Publication Date:
2023

DOI:
https://doi.org/10.26190/unswworks/24852

License:
https://creativecommons.org/licenses/by/4.0/
Link to license to see what you are allowed to do with this resource.

Downloaded from http://hdl.handle.net/1959.4/101145 in https://unswworks.unsw.edu.au on 2023-06-22
Hierarchical Information and Data Modeling for Neural-based Recommender Systems

Peng Yi

A thesis in fulfilment of the requirements for the degree of

Doctor of Philosophy

School of Computer Science and Engineering
Faculty of Engineering
The University of New South Wales

December 2022
Hierarchical Information and Data Modeling for Neural-based Recommender Systems

Thesis Abstract

In the era of information flooding, efficient information retrieval has become a non-negligible problem. Recommender system, an information filtering system that provides customized suggestions of items that are most likely to be of interest to users has been applied to many customer-oriented services. Recently, more and more neural-based and graph-based models have been studied and adapted in recommender systems, owing to their superiority in dealing with fundamental machine learning problems. However, most existing approaches merely focus on accuracy improvements and ignore that higher recommendation accuracy does not directly imply better recommendation. This thesis aims to propose novel methods for recommender systems that enable higher recommendation accuracy and higher recommendation satisfaction simultaneously. In this thesis, three approaches are proposed from two perspectives, which are (1) generating more personalized individual embeddings and (2) reducing inference latency.

To improve the embedding learning process, the valuable information stored in pairwise preference differences and the hierarchical structures exhibited in user (item) latent relationships are explored and investigated. Firstly, a novel and straightforward pointwise training strategy, named Difference Embedding (DIE), is proposed to capture the valuable information retained in pairwise preference differences. More specifically, by using a novel projection on the designed pairwise differences function, the final derived pointwise loss function allows recommendation models to encode valuable personalized information and achieve more customized predictions. Moreover, a U-shaped graph convolutional network-based recommender system, named UGCN, is proposed to explore the implicit and inherent hierarchies of the user or item. Concretely, with the hierarchical encoding-decoding process, the UGCN model is able to capture user-item relationships at various resolution scales and would finally result in better preference customization.

To reduce inference latency, two knowledge distillation methods are also proposed in the model construction and training process. By training with the valuable information distilled from the sophisticated teacher recommends, the compact student model can achieve extraordinary recommendation performances and light model architecture simultaneously.

ORIGINALITY STATEMENT

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

COPYRIGHT STATEMENT

I hereby grant the University of New South Wales or its agents a non-exclusive licence to archive and to make available (including to members of the public) my thesis or dissertation in whole or part in the University libraries in all forms of media, now or hereafter known. I acknowledge that I retain all intellectual property rights which subsist in my thesis or dissertation, such as copyright and patent rights, subject to applicable law. I also retain the right to use all or part of my thesis or dissertation in future works (such as articles or books).

For any substantial portions of copyright material used in this thesis, written permission for use has been obtained, or the copyright material is removed from the final public version of the thesis.

AUTHENTICITY STATEMENT

I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis.
UNSW is supportive of candidates publishing their research results during their candidature as detailed in the UNSW Thesis Examination Procedure.

Publications can be used in the candidate's thesis in lieu of a Chapter provided:

- The candidate contributed greater than 50% of the content in the publication and are the "primary author", i.e. they were responsible primarily for the planning, execution and preparation of the work for publication.
- The candidate has obtained approval to include the publication in their thesis in lieu of a Chapter from their Supervisor and Postgraduate Coordinator.
- The publication is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in the thesis.

The candidate has declared that their thesis has publications - either published or submitted for publication - incorporated into it in lieu of a Chapter/s. Details of these publications are provided below.

Publication Details #1

<table>
<thead>
<tr>
<th>Full Title</th>
<th>A U-shaped Hierarchical Recommender by Multi-resolution Collaborative Signal Modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authors</td>
<td>Peng Yi, Xiongcai Cai, and Zileng Li</td>
</tr>
<tr>
<td>Journal or Book Name</td>
<td>In proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2022)</td>
</tr>
<tr>
<td>Volume/Page Numbers</td>
<td></td>
</tr>
<tr>
<td>Date Accepted/Published</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>published</td>
</tr>
<tr>
<td>The Candidate's Contribution to the Work</td>
<td>I contribute to the design of the architecture, the implementation, and the evaluation of all models. I also wrote up and finalized the accepted paper.</td>
</tr>
<tr>
<td>Location of the work in the thesis and/or how the work is incorporated in the thesis</td>
<td>A large portion of Chapter 4 has appeared in this paper.</td>
</tr>
</tbody>
</table>

Publication Details #2

<table>
<thead>
<tr>
<th>Full Title</th>
<th>Difference Embedding for Recommender Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authors</td>
<td>Peng Yi, Xiongcai Cai, and Zileng Li</td>
</tr>
<tr>
<td>Journal or Book Name</td>
<td>Data Mining and Knowledge Discovery</td>
</tr>
<tr>
<td>Volume/Page Numbers</td>
<td></td>
</tr>
<tr>
<td>Date Accepted/Published</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>accepted</td>
</tr>
<tr>
<td>The Candidate's Contribution to the Work</td>
<td>I contribute to the design of the architecture, the implementation, and the evaluation of all models. I also wrote up and finalized the accepted paper.</td>
</tr>
<tr>
<td>Location of the work in the thesis and/or how the work is incorporated in the thesis</td>
<td>A large portion of Chapter 3 has appeared in this paper.</td>
</tr>
</tbody>
</table>

Candidate's Declaration

I confirm that where I have used a publication in lieu of a chapter, the listed publication(s) above meet(s) the requirements to be included in the thesis. I also declare that I have complied with the Thesis Examination Procedure.
Abstract

In the era of information flooding, efficient information retrieval has become a non-neglectful problem. Recommender system, an information filtering system that provides customized suggestions of items that are most likely to be of interest to users has been applied to many customer-oriented services. Recently, more and more neural-based and graph-based models have been studied and adapted in recommender systems, owing to their superiority in dealing with fundamental machine learning problems. However, most existing approaches merely focus on accuracy improvements and ignore that higher recommendation accuracy does not directly imply better recommendations. This thesis aims to propose novel methods for recommender systems that enable higher recommendation accuracy and higher recommendation satisfaction simultaneously. In this thesis, three approaches are proposed from two perspectives, which are (1) generating more personalized individual embeddings and (2) reducing inference latency.

To improve the embedding learning process, the valuable information stored in pairwise preference differences and the hierarchical structures exhibited in user (item) latent relationships are explored and investigated. Firstly, a novel and straightforward pointwise training strategy, named Difference Embedding (DifE), is proposed to capture the valuable information retained in pairwise preference differences. More specifically, by using a novel projection on the designed pairwise differences function, the final derived pointwise loss function allows recommendation models to encode valuable personalized information and achieve more customized predictions. Moreover, a U-shaped graph convolutional network-based recommender system, named UGCN, is proposed to explore the implicit and inherent hierarchies of the user or item. Concretely, with the hierarchical encoding-decoding process, the UGCN model is able to capture user-item relationships at various resolution scales and would finally result in better preference customization.

To reduce inference latency, two knowledge distillation methods are also proposed in the model construction and training process. By training with the valuable information distilled from the sophisticated teacher recommenders, the compact student model can achieve extraordinary recommendation performances and light model architecture simultaneously.
Acknowledgement

First of all, I would like to extend my sincere gratitude to my supervisor Dr. Xiongcai Cai for his immeasurable support, superb guidance, and instructive advice during my PhD study. I am deeply grateful for his help, encouragement, and support in the completion of this thesis.

I gratefully acknowledge the financial support for my doctoral study provided by the China Scholarship Council (CSC).

I also thank Mr. Ziteng Li for his assistance and valuable discussions.

Special thanks would go to my beloved husband Gang Yang and my parents Mr. Leiming Yi and Mrs. Yange Peng for their material comfort and emotional support all through these years.
List of Publications

Large portion of Chapter 3 have appeared in the following paper:


Large portion of Chapter 4 have appeared in the following paper:


Large portion of Chapter 5 have appeared in the following paper:

## 2 Literature Review

### 2.1 Overview of Recommender Systems

- **2.1.1 Recommendation algorithms**
- **2.1.2 Type of input data**
- **2.1.3 Training of recommendation models**

### 2.2 Review on Recent Recommendation Approaches

- **2.2.1 Neural-based collaborative filtering**
- **2.2.2 Neural-based content-based recommenders**
- **2.2.3 Discussions on recent neural-based recommenders**

### 2.3 Review on Knowledge Distillation for Recommender System

- **2.3.1 Distilled knowledge in recommender systems**
- **2.3.2 Teacher-student architecture**
- **2.3.3 Distillation schemes**

### 2.4 Summary

## 3 Preference Differences Capturing in Recommender Systems

### 3.1 Background

- **3.1.1 Motivation of preference differences capturing**
- **3.1.2 Contribution of this chapter**

### 3.2 Proposed Method

- **3.2.1 Co-preference ratio**
- **3.2.2 Derivation of the new loss function**
- **3.2.3 Application of loss function**
- **3.2.4 Theoretical analysis of difference embedding**
3.3 Experiments Settings ............................................. 62
  3.3.1 Dataset description ........................................ 62
  3.3.2 Evaluation metrics ........................................ 63
  3.3.3 DiffE models ............................................. 64
  3.3.4 Selected baseline models ................................ 70
  3.3.5 Negative sampling strategy ............................. 72
  3.3.6 Other implementation details ........................... 72

3.4 Experiments Results ............................................ 72
  3.4.1 Loss function analysis (RQ1) ........................... 73
  3.4.2 Overall performance comparison (RQ2) ................ 79
  3.4.3 Hyper parameters analysis (RQ3) ........................ 81

3.5 Summary ......................................................... 83

4 Hierarchical Information Modeling in Recommender Systems 85
  4.1 Background .................................................... 85
    4.1.1 Motivation of hierarchical information modeling .... 85
    4.1.2 Contribution of this chapter .......................... 88
  4.2 Preliminary .................................................... 88
    4.2.1 The basic collaborative filtering model ............... 88
    4.2.2 Modeling implicit hierarchies ......................... 89
  4.3 UGCN Recommender ........................................... 90
    4.3.1 Item (user) hierarchies extraction .................... 91
    4.3.2 Graph node embedding learning ....................... 91
    4.3.3 Diverse embedding aggregation ....................... 92
    4.3.4 UGCN architecture ..................................... 93
    4.3.5 Theoretical analysis for UGCN recommender .......... 94
4.4 Experiments Settings

4.4.1 Dataset description

4.4.2 Evaluation metrics

4.4.3 Implementation for UGCN recommender

4.4.4 Compared models

4.4.5 Other implementation details

4.5 Experiment Results

4.5.1 Prediction accuracy comparison (QR1)

4.5.2 Recommendation satisfaction comparison (QR2)

4.5.3 Hyper parameter analysis (QR3)

4.6 Conclusion

5 Prediction Distillation in Hierarchical Recommenders

5.1 Background

5.1.1 Motivation of applying KD on hierarchical recommenders

5.1.2 Contribution of this chapter

5.2 Discussions on existing KD approaches

5.3 Proposed Methods

5.3.1 Prediction-based distillation for data enhancement

5.3.2 Prediction-based distillation for better hierarchies modeling

5.4 Experiments Settings

5.4.1 Dataset description

5.4.2 Evaluation metrics

5.4.3 Implementation of the teacher model

5.4.4 Implementation of the student model

5.4.5 Implemented recommenders
5.4.6 Other implementation details ........................................ 118

5.5 Experiment Results ................................................... 118

5.5.1 Overall accuracy comparison (QR1) ............................. 118

5.5.2 Recommendation satisfaction comparison (RQ2) ............... 120

5.5.3 Hyper parameters analysis (RQ3) ............................... 121

5.6 Summary .................................................................. 123

6 Conclusion and Future Directions .................................... 124

6.1 Thesis Summary ....................................................... 125

6.2 Future Extensions and Directions .................................. 127

References ................................................................. 129
List of Figures

1.1 The architecture diagram of this thesis. ........................................ 9

2.1 Examples of Historical Interactions ........................................ 19

2.2 An illustration of NeuMF Model ............................................. 27

2.3 An illustration of DMF Model ............................................... 28

2.4 An illustration of LightGCN Model ......................................... 31

2.5 An intuitive example of different types of knowledge within a teacher model .......................................................... 39

3.1 An illustration of the proposed DifE-framework structure. The output of the framework is computed by combining the learned user (item) biases and the point-wise preference generated from the point-wise recommendation model. The whole framework is independent of recommenders and can be applied to any point-wise recommendation models, such as MF [1], NeuMF [2] and LightGCN [3]. The detailed architecture of these mentioned models can be found in the referenced paper. ........................................ 59

3.2 Comparison for potential values of $CPR_{ui}$ and $SUBD_{ui}$ with a given preference $P_{ui} = 5$ and an unknown $P_{uj}$ ranging from 1 to 5 ........................................ 61

3.3 Overview of MF-DifE structure .............................................. 65

3.4 Overview of the NeuMF-DifE structure ..................................... 66

3.5 Overview of LightGCN-DifE (SGL-DifE) structure ....................... 68

3.6 MRR results on different $\text{Top}@K$ recommendation scenarios. .................................................. 76

3.7 MRR results on different $\text{Top}@K$ recommendation scenarios. .................................................. 78
3.8 The statistics of hit frequency for different rating value groups of Neural-based and GCNs-based models. 81

3.9 Hyper-parameters comparison on number of neighbors. 82

4.1 An illustration of a 3-levels UGCN Recommender. In this example, the original user-item interaction graph are hierarchically stacked with two compact graphs generated by Pooling operations. Then, simplified GCNs layers are implemented to extract the multi-resolution collaborative signals at three graphs of different scales. The Un-pooling operations would gradually restore the finer graph architecture and integrate the informative signals into final user (item) embeddings. 93

4.2 Coverage Comparison Results 105

4.3 Novelty Comparison Results 106

4.4 Hyper-parameters Comparison Results 107

5.1 R-Coverage@20, H-Coverage@20 and Novelty@20 comparison results on baseline and compact student models. 120

5.2 Hyper-parameters Comparison Results on Epinions 122
List of Tables

3.1 Statistics of experimental datasets. .......................... 62

3.2 Recall@20 and NDCG@20 results for base, SUBD-optimized and DifE-optimized models. .......................... 74

3.3 Novelty@K results of SGL and SGL-DifE models on different datasets. 75

3.4 Recall@20 and NDCG@20 results for the DifE-optimized models and the most recent state-of-the-art recommenders. 79

4.1 Statistics of experimental datasets. .......................... 96

4.2 The comparison of overall performance among UGCN and baseline methods. 103

4.3 The comparison of overall performance among UGCN and baseline methods. 104

5.1 Recall@20 and NDCG@20 results for the implemented recommenders. 119
Abbreviations

AEs Autoencoders
AFM Attentional factorization machine
AmazonMIs Amazon musical instruments dataset
ANR Aspect-based neural recommender
BCE Binary cross entropy
BPR Bayesian personalized ranking
CDAE Collaborative denoising autoencoder
CNNs Convolutional neural networks
ConvNCF Convolutional neural collaborative filtering
CPR Co-preference ratio
CTR Click-through rate
D-Attn Dual attention-based convolutional neural networks
DCN Deep and cross network
DeepCoNN Deep cooperative neural networks
DELF Dual-embedding-based deep latent factor
DiE Difference embedding
DLDA Deep latent Dirichlet allocation
FMs Factorization machines
GAN Generative adversarial networks
GCMC GCNs-based matrix completion framework
GCNs Graph convolutional networks
GMF General matrix factorization
GraphFM Graph factorization machine
HIN Heterogeneous information networks
IBCF Item-based collaborative filtering
KD Knowledge distillation
MF Matrix factorization
MLAM Multilevel attraction model
MLPs Multi-layer perceptrons
MRR Mean reciprocal rank
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>Mean squared error</td>
</tr>
<tr>
<td>NCF</td>
<td>Neural collaborative filtering</td>
</tr>
<tr>
<td>NCR</td>
<td>Neural network-based collaborative ranking</td>
</tr>
<tr>
<td>NeuMF</td>
<td>Neural matrix factorization</td>
</tr>
<tr>
<td>NFM</td>
<td>Neural factorization machine</td>
</tr>
<tr>
<td>NGCF</td>
<td>Neural graph collaborative filtering</td>
</tr>
<tr>
<td>NGMM</td>
<td>Neural gaussian mixture model</td>
</tr>
<tr>
<td>RBMs</td>
<td>Restricted boltzmann machines</td>
</tr>
<tr>
<td>PMF</td>
<td>Probabilistic matrix factorization</td>
</tr>
<tr>
<td>PNN</td>
<td>Product-based neural networks</td>
</tr>
<tr>
<td>RNS</td>
<td>Random negative</td>
</tr>
<tr>
<td>RNNs</td>
<td>Recurrent neural networks</td>
</tr>
<tr>
<td>RRD</td>
<td>Relaxed ranking distillation</td>
</tr>
<tr>
<td>SDAE</td>
<td>Stacked denoising autoencoders</td>
</tr>
<tr>
<td>SUBD</td>
<td>Subtraction-based discrepancy</td>
</tr>
<tr>
<td>TransNets</td>
<td>Transformational neural networks</td>
</tr>
<tr>
<td>UBCF</td>
<td>User-based collaborative filtering</td>
</tr>
<tr>
<td>xDeepFM</td>
<td>Extreme deep factorization machine</td>
</tr>
</tbody>
</table>
Notations

\( a_{ij} \) \hspace{1em} The weight of an edge between entity \( i \) and \( j \) in teacher space

\( A^s \) \hspace{1em} The adjacency matrix of the constructed topological graph in the student space

\( A^t \) \hspace{1em} The adjacency matrix of the constructed topological graph in the teacher space

\( bpM \) \hspace{1em} The binary matrix indicating whether a pair of entity belongs to the same group

\( e^s \) \hspace{1em} The representation or embeddings learned by the student model

\( e_i^s \) \hspace{1em} The representation or embeddings of entity \( i \) learned by the student model

\( e^t \) \hspace{1em} The representation or embeddings learned by the teacher model

\( e_i^t \) \hspace{1em} The representation or embeddings of entity \( i \) learned by the teacher model

\( e_i \) \hspace{1em} The learned embedding of item \( i \)

\( e_i^{(k)} \) \hspace{1em} The item \( i \)’s embedding generated by \( k-th \) layer in GCNs

\( e_j \) \hspace{1em} The learned embedding of item \( j \)

\( e_u \) \hspace{1em} The learned embedding of user \( u \)

\( e_u^{(k)} \) \hspace{1em} The user \( u \)’s embedding generated by \( k-th \) layer in GCNs

\( e_{u(n)} \) \hspace{1em} The user \( u \)’s initial embedding of the \( n-th \) hierarchical level

\( e_{i-cn} \) \hspace{1em} The item \( i \)’s collaborative embedding with the \( n-th \) resolution scale

\( f(\cdot) \) \hspace{1em} The projection function or network

\( f_k(\cdot) \) \hspace{1em} The \( k-th \) projection function or network

\( G(U, I, E) \) \hspace{1em} The bipartite graph with User Nodes \( U \), Item Nodes \( I \), and Edges \( E \)

\( h_u \) \hspace{1em} The ranking position of the first hit item in user \( u \)’s \( L_{rec@K} \)

\( H^s \) \hspace{1em} The student adjacency matrix of topological group-level graph

\( H^t \) \hspace{1em} The teacher adjacency matrix of topological group-level graph

\( L_{rec@K} \) \hspace{1em} The list of top@\( k \) recommendation

\( L_{tar} \) \hspace{1em} The list of target item

\( ohe_i \) \hspace{1em} The one-hot encoding of item \( i \)

\( p^s(z_i, T) \) \hspace{1em} The soft target of item \( i \) generated by student model

\( p^t(z_i, T) \) \hspace{1em} The soft target of item \( i \) generated by teacher model

\( p_i \) \hspace{1em} The average rating of item \( i \)

\( p_u \) \hspace{1em} The modified preference of item \( i \) given by user \( u \)

\( p_u \) \hspace{1em} The average rating given by user \( u \)

\( pop_i \) \hspace{1em} The popularity of item \( i \)
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{ui}$</td>
<td>The adjusted rating of item $i$ given by user $u$</td>
</tr>
<tr>
<td>$P_{uj}$</td>
<td>The adjusted rating of item $j$ given by user $u$</td>
</tr>
<tr>
<td>$r_{ui}$</td>
<td>The rating of item $i$ given by user $u$</td>
</tr>
<tr>
<td>$RN$</td>
<td>The number of ratings in the training set</td>
</tr>
<tr>
<td>$RS$</td>
<td>The number of randomly selected uninterested items</td>
</tr>
<tr>
<td>$S$</td>
<td>The student model</td>
</tr>
<tr>
<td>$Sim_u$</td>
<td>The set of user $u$'s similar neighbors</td>
</tr>
<tr>
<td>$Sim_i$</td>
<td>The set of item $i$'s similar neighbors</td>
</tr>
<tr>
<td>$T$</td>
<td>The teacher model</td>
</tr>
<tr>
<td>$T_i$</td>
<td>The $i$-th teacher model</td>
</tr>
<tr>
<td>$\hat{x}_{uij}$</td>
<td>The predicted preference discrepancies among item $i$ and item $j$</td>
</tr>
<tr>
<td>$\hat{y}^s$</td>
<td>The predicted preference generated by student model</td>
</tr>
<tr>
<td>$\hat{y}^s_{\pi_r}$</td>
<td>The predicted preference of $r$-th item in $\pi_r$ computed by the student model</td>
</tr>
<tr>
<td>$y_i$</td>
<td>The ground truth preference of item $i$</td>
</tr>
<tr>
<td>$\hat{y}_i$</td>
<td>The predicted preference of item $i$</td>
</tr>
<tr>
<td>$\hat{y}_{ui}$</td>
<td>The predicted preferences user $u$ given to item $i$</td>
</tr>
<tr>
<td>$y_{uj}$</td>
<td>The predicted preferences user $u$ given to item $j$</td>
</tr>
<tr>
<td>$z_i$</td>
<td>The estimated logits for the $i$-th class</td>
</tr>
<tr>
<td>$\alpha_i$</td>
<td>The learnable weight</td>
</tr>
<tr>
<td>$\Theta$</td>
<td>The parameter vector</td>
</tr>
<tr>
<td>$\sigma(\cdot)$</td>
<td>The activation function</td>
</tr>
<tr>
<td>$\pi_{1..K}$</td>
<td>The top@K items that predicted by the teacher model</td>
</tr>
<tr>
<td>$\pi_r$</td>
<td>The $r$-th ranked item in $\pi_{1..K}$</td>
</tr>
<tr>
<td>$E$</td>
<td>The embedding matrix</td>
</tr>
<tr>
<td>$\mathcal{N}^i$</td>
<td>The set of all items in the recommender system</td>
</tr>
<tr>
<td>$\mathcal{N}^u$</td>
<td>The set of all users in the recommender system</td>
</tr>
<tr>
<td>$\mathcal{N}_i$</td>
<td>The set of users that has rated or interacted item $i$</td>
</tr>
<tr>
<td>$\mathcal{N}_u$</td>
<td>The set of items that user $u$ have interacted</td>
</tr>
<tr>
<td>$T$</td>
<td>The temperature factor</td>
</tr>
<tr>
<td>$T(\cdot)$</td>
<td>The temperature function</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

In the age of data deluge, efficient information retrieval becomes non-negligible. To efficiently locate the target item from millions of potential products, many information filtering techniques have been implemented since the late 1980s \[4\]. Among all of the existing approaches, recommender system, an information filtering system that can assist users in their information-seeking process and would benefit both customers and sellers, has gained considerable research attention. More specifically, recommender system focuses on modeling user’s (item’s) personalized information from their historical data and aims to predict the potential products that the user will purchase in the near future. With its extraordinary ability of efficient information searching and effective product recommendation, recommender system is permeating our daily lives.

The most direct example would be commodities recommendations on Amazon. When searching for a suitable product by its keywords or title on Amazon\[1\], the recommender system behind the platform would automatically filter and select potential items based on historical data. Hence, without getting lost in the hundreds of products that vary in color, texture, and price, the customer could mainly focus on the top-ranked items and quickly

CHAPTER 1. INTRODUCTION

choose the one they prefer. Another example is movie recommendations on Netflix\(^2\). Once a new movie is released, the system could filter the target users who might be interested in accordance with their preferences. Then, without wasting a tremendous amount of money on advertising, the new movie could only be precisely delivered to these potential audiences and still come into prominence. Apart from the above two examples, news recommendations on New York Times\(^3\) music recommendations on Spotify\(^4\) restaurant recommendations on Yelp\(^5\) studying material recommendations on Coursera\(^6\) etc., are also within reached recommendation services. Briefly speaking, recommender systems can increase retailers' sales and simply individuals’ daily routines. Therefore, vigorous studies on recommender systems are always necessary and indispensable.

There are mainly two types of basic recommender systems, which are: (1) collaborative filtering-based recommender systems\(^5\) that utilize user and item historical behaviors to analyze personalized preferences and (2) content-based recommender systems\(^6\) that rely more on user and item attribute information to generate predictions. To be more specific, collaborative filtering-based algorithms aim to extract collaborative preferences for personalized predictions. In other words, these collaborative filtering-based approaches implicitly assume that similar users will present analogous behaviors. Hence, by extracting valuable collaborative signals from historical behavior data, collaborative filtering-based algorithms can easily make inferences for non-interacted items. Moreover, it is worth pointing out that historical interaction is not the only type of data useful for discovering meaningful recommendations, the descriptive attributes for users or items are also quite valuable in modeling user-item relationships. From this perspective, content-based recommenders utilize the user or item attributes to calculate similarity and model relationship, then match users to items similar to what they have purchased. Compared to the collaborative filtering-based method, content-based recommenders count on the user (item) attributes to model

\(^2\)https://www.netflix.com/
\(^3\)https://www.nytimes.com/international/
\(^4\)https://www.spotify.com/
\(^5\)https://www.yelp.com/
\(^6\)https://www.coursera.org/
relationships and are able to recommend new items to potential customers even if the new item does not obtain any interactions. While content-based algorithms tend to filter out potential items that are similar to users’ historically interacted ones and hence often generate over-specialized and less-novel recommendations. Failure to discover novel and surprising products would vastly decrease the recommendation satisfaction of customers. Moreover, considering the fact that detailed descriptions of items or users are inapplicable in most recommendation scenarios, the application of content-based recommenders is often limited to specific scenarios. Whereas collaborative filtering-based models are suitable in most recommendation cases.

Early recommendation-based approaches often adopt linear algorithms, such as matrix factorization [7], sparse linear method [8], etc., to construct recommender systems. Although these methods can outperform naive filtering methods (e.g., popularity-based filtering), these simple strategies are insufficient for precisely capturing user (item) personalized preferences. Meanwhile, as hardware devices develop rapidly, collecting abundant and complicated data becomes achievable in many real-world recommender systems, complicating the current recommendation. Recently, deep learning techniques [9] have become an eye-actching research topic due to their superiority in processing massive heterogeneous data. In other words, by adopting sophisticated neural-based or graph-based architectures to explore complicated relationships in massive input data, deep learning algorithms can perform extraordinarily on different machine learning problems [10]. Briefly speaking, recommender systems are facing the challenges of processing massive and heterogeneous data, while neural-based models are superior at relationship modeling from complicated input. Thus, it is a popular research direction to integrate the advantages of neural-based models into recommender systems to generate better recommendation results [11].

Based on the discussions above, this thesis aims to design novel algorithms for generating personalized recommendation predictions with neural-based architectures from multiple perspectives: improving the data modeling process, constructing better recommendation architecture, and optimizing the training and inference process.

The challenges, motivations, research objectives and contributions of this thesis will be
1.1 Challenges and Problems

Although many state-of-the-art recommendation algorithms have demonstrated outstanding performances on some experimental datasets. There are still severe limitations of those current approaches. The major challenges or problems concerned in this thesis are presented below:

- Complicated relationships. Traditional recommender systems utilize straightforward linear algorithms to model user-item relationships, resulting in unsatisfactory recommendations. To this end, many recent approaches employ sophisticated non-linear models to explore user-item relationships. Although these models can perform better than traditional ones, they also have the obvious drawback of neglecting the inherent hierarchy of items (users). In other words, current recommenders focus on modeling individual items or users independently but ignore the valuable items’ (or users’) hierarchy information, leading to insufficient preference knowledge capturing. In brief, it remains quite challenging to construct a novel recommendation architecture that can precisely capture user (item) personalized preferences from massive, complicated, and sparse input data.

- Insufficient exploration of available data. Among all different recommendation approaches, the collaborative filtering-based method is one of the most widely applied paradigms owing to their remarkable performances as well as easily-collected input data. Despite many attempts have utilized new and sophisticated architecture to construct collaborative filtering-based recommenders, there is little research on exploring better data modeling methods that can largely influence the final information extraction procedure. In other words, the insufficient exploration and inadequate utilization of accessible data not only squander valuable preference information but also restrict collaborative filtering-based recommenders from generating customized
1.2. MOTIVATION

predictions.

• Huge computation expenses. As a larger model with more learning parameters has a higher expression capacity and generally performs better on recommendation problems, the size of recent recommender systems has kept rising with respect to the continuously increasing amount of available data. However, endlessly increasing the model size would be problematic, leading to huge memory and computation costs. These costs not only slow down online inference and hinder their adoption in real-time recommendation applications but also limit the models’ capability to capture shifting user (item) interest and result in outdated predictions. Hence, it is an important concern for current recommendation studies to retain the light model architecture and maintain outstanding performances simultaneously.

This thesis solves the above challenges and problems in the recommender system by utilizing graph-based architecture to explore user (item) implicit hierarchies, capture valuable personalized information from pairwise differences, and obtain compact and extraordinary recommendation models by knowledge distillation.

1.2 Motivation

The rest of this section aims at answering the following questions:

• Why exploring user (item) hierarchies?
• Why emphasizing data modeling and utilization?
• Why employing knowledge distillation?

1.2.1 Why exploring user (item) hierarchies?

As discussed above, most existing recommendation models only focus on modeling individual items or users independently, which cannot fully capture the items’ (or users’)
CHAPTER 1. INTRODUCTION

Hierarchies information and hence fail to precisely model personalized preference. In fact, the user (item) hierarchies play a quite predominant role in precisely exploring user (item) preference information. Take the movie Frozen as an example, it belongs to the subgenre "Family Animation" and can be further categorized into the genre "Animation", demonstrating a hierarchical structure of "individual → sub-genre → genre". Similarly, users may present a similar hierarchy of "individual → occupation → age". Since items of the same subgenre (or genre) are likely to have similar attributes, they are likely to acquire similar preferences [12]. Thus, hierarchical information about items or users can very likely improve the preference modeling process of recommender systems.

Moreover, it is worth pointing out that in real-world recommendation scenarios, explicit hierarchies are often not applicable [13]. For this reason, some researchers [13] utilize stacked linear models to learn implicit item (user) hierarchies from historical interactions. The success of these models confirms the validity of exploring implicit hierarchies. However, the limited model expressions of linear model still largely restrict their applications on complex real world recommendation scenarios. Moreover, solely focusing on building two separate multi-layer architectures for users or items is sub-optimal and problematic since the valuable information stored in user (item) hierarchy is the multi-resolution collaborative signals rather parallel multi-layer architectures. To be more specific, given the interaction that a user "Alice" like a movie "Frozen", exploring the structure of "Alice → Second Grade Nursery Student → Kids" and "Frozen → Family Animation → Animation" merely displays the inner association for a user or an item. Take a step further, this interaction not only reveals "Alice’s" unique preference for "Frozen", but also represents a shared interest that "Second Grade Nursery Students" may prefer "Family Animation", or announces a more general signal that "Kids" prefer "Animation". With these multi-resolution collaborative signals, user (item) preferences information can be modeled in a more reasonable and precise way, leading to better recommendation predictions. Therefore, user (item) implicit hierarchies and multi-resolution collaborative signals will be explored and exploited in this thesis.
1.2. MOTIVATION

1.2.2 Why emphasizing data modeling and utilization?

Apart from engaging new non-linear architectures to construct recommendation models, effectively exploiting subtle personalized information from historical interaction data is also crucial for the success of recommender systems. Specifically, as the fundamental resources required by both collaborative filtering-based and content-based recommenders, historical interactions are simply adopted as the indication for potential pointwise connectivity in most recommendation approaches. This pointwise relationship does not indicate much about the actual recommendation rankings seen by users [14] and tends to be sub-optimal in ranking-based recommendation cases. In fact, more valuable information stored in historical interactions is the personalized preferences that indicate the user’s specific fondness degree for different items. To extract personalized information from interaction data, Rendle et al. [15] propose to model the pairwise differences between the user’s interacted and non-interacted items. Although the concept of capturing users’ personalized information from the pairwise differences is appealing, the current approach that simply binarizes historical interactions into two classes is not ideal since it neglects more subtle and valuable fondness differences among user-interacted items, leading to incomplete and inadequate personalized information extraction. In other words, better pairwise preference differences modeling method that can precisely capture personalized preference information will be investigated.

It is worth pointing out that emphasizing data modeling and utilization brings no conflict with incorporating auxiliary information. More concretely, better data modeling and utilization focuses on thoroughly exploring available data while incorporating auxiliary information targets to collect more accessible data. The discussion within this thesis will be concentrated on proposing data modeling and utilization algorithms owing to their better generalization ability.
1.2.3 Why employing knowledge distillation?

As a larger model with more learning parameters has a higher expression capacity and generally performs better, the size of recent neural-based recommender systems has kept rising with respect to the continuously increasing amount of available data. However, Endlessly increasing the complexity and the size of neural-based recommenders not only brings high inference latency but also causes long delays in updating model parameters, leading to out-of-date recommendations. Hence, how to achieve outstanding performance yet retain the light recommendation architecture has become an prevalent concern in recommender systems.

To this end, a model compression and knowledge transferring technique, named knowledge distillation (KD) [16], which enables a compact model to obtain competitive performance and model efficiency simultaneously, has drawn increasing research attention recently. More specifically, by distilling the essential knowledge from the fine-tuned cumbersome teacher, the compact student can be trained efficiently and effectively, demonstrating astounding performances with lower computational costs. A state-of-the-art KD-based system is composed of three key components: knowledge, teacher-student architecture, and distillation scheme. Knowledge refers to the available information that can be distilled from the cumbersome teacher. Teacher-student architecture can be regarded as a carrier that enables the information transition among models. Meanwhile, distillation schemes are essential for the success of the knowledge distillation system since it reveals how can the distilled knowledge from the teacher model guide the student model to obtain better performance. After introduced and formally popularized by Hinton et al. [17], KD has been extensively researched and applied in a variety of domains, including speech recognition, image processing etc.. Therefore, KD-based methods would enable light recommendation models to achieve outstanding recommendation performances.
1.3 Research Objective

With the challenges and motivations presented above, the research objectives of this thesis are presented below.

- Constructing a graph-based recommendation model to explore user (item) implicit hierarchies and extract valuable multi-resolution collaborative signals within each hierarchical layer.
- Proposing a better data modeling algorithm that enables state-of-the-art recommenders to capture personalized preferences information from pairwise preference differences.
- Employing knowledge distillation to obtain a compact hierarchical recommender with extraordinary performances.

The following overall architecture diagram presents the relationships among the above three research objectives:

![Architecture Diagram](image)

Figure 1.1: The architecture diagram of this thesis.

1.4 Overview of this thesis

The contents of this thesis are summarized below.
Chapter 1 outlines the focusing challenges in this thesis, followed by the research motivations and objectives.

Chapter 2 first introduces the foundations for general recommender systems. Then, an in-depth review of recent approaches for recommender systems is provided, together with a review of integrating knowledge distillation techniques for recommender systems.

Chapter 3 presents a novel and straightforward pointwise training strategy, named difference embedding (DifE), for recommender systems that capture the personalized information retained in pairwise preference differences. A model-independent recommender training framework is proposed and integrated with different state-of-the-art models, leading to huge performance improvements for original recommenders.

Chapter 4 demonstrates a hierarchical recommender, named UGCN, that can capture the multi-resolution collaborative signals from user (item) implicit hierarchies. With valuable multi-resolution collaborative information, this hierarchical recommender can precisely capture user (item) subtle preferences and achieve competitive performances.

Chapter 5 focuses on employing knowledge distillation strategies on hierarchical recommenders. Two prediction-based strategies are proposed to distill valuable knowledge from a sophisticated teacher to a compact student, enabling hierarchical recommenders to achieve extraordinary recommendation performances and light model architecture simultaneously.

Chapter 6 summarizes the content of the thesis and outlines future research directions.

1.5 Contributions to recommender system

A effective pointwise data modeling framework

This thesis proposes a pointwise framework that can extract valuable personalized information from pairwise preference differences. This framework can incorporate most existing interaction-based recommendation models, enabling them to generate more accurate and
customized predictions. Moreover, the proposed pairwise preference difference modeling method offers a novel way to emphasize the user’s subtle preference shift among interested and disliked items. And the point loss function derived from the pairwise modeling method can guide the original pointwise recommendation architecture to capture informative personalized information without extra computational.

**Implicit user (item) hierarchies exploration**

A graph-based hierarchical recommender that can explore the user (item) implicit hierarchical relationships is proposed, which not only confirms the importance of considering user (item) inherent hierarchies in constructing a recommendation model but also extends the existing hierarchical recommender to superior graph-based architecture, endowing more abilities of the hierarchical model in analyzing with complex and heterogeneous recommendation data. Meanwhile, the pooling algorithm designed in user (item) hierarchies exploration process provides a similarity-based solution in merging two-hop neighbors in a bipartite user-item graph.

**Novel training strategies for hierarchical recommender**

The proposed training strategies distill and transfer information among hierarchical recommenders for the first time, offering a possible direction for constructing compact but outstanding hierarchical recommenders in real-world recommendation scenarios. Furthermore, the proposed prediction-based distillation, which utilizes the updated similarity to obtain better hierarchical architecture, enriches the scheme and the knowledge type of distillation in the recommendation field.
Chapter 2

Literature Review

This chapter first introduces the foundations of recommender systems, including recommendation algorithms, input data, and training methods. Then, an in-depth introduction to recent approaches for recommender systems is discussed, followed by the current challenging problems in the field. After that, a review on integrating knowledge distillation techniques for recommender systems is provided, which gives insights for reducing inference latency by knowledge transformation through a teacher-student paradigm.
2.1 Overview of Recommender Systems

Recommender system is a sub-field of information filtering systems that seek to predict the "rating" or "preference" a user would give to an item. In other words, by processing diverse user(item) historical data, recommender system aims to find the potential products that a user will purchase in the near future. In recent years, recommender systems have become increasingly popular and been extensively utilized by a multitude of online retailers and businesses to boost their revenues. Common examples of utilization of recommender systems include movie recommendations demonstrated on Netflix\footnote{https://www.netflix.com/}, product recommendations presented on Amazon\footnote{https://www.amazon.com.au/}, news recommendations displayed on New York Times\footnote{https://www.nytimes.com/international/}, people recommendations on RSVP \cite{5}, etc. The widespread use of recommender systems enabled various retailers and businesses to increase their sales and services and helped customers effectively and efficiently access their personalized products.

2.1.1 Recommendation algorithms

The basic recommender systems can be divided into two different types, which are (1) collaborative filtering-based recommender systems \cite{18} that utilize user and item historical behaviors to analyze personalized preferences and (2) content-based recommender systems \cite{6} that more rely on user and item attribute information to generate predictions. The brief discussions of these two basic recommender systems are presented below.

(1) collaborative filtering-based recommender systems

The basic idea behind collaborative filtering-based algorithms is to extract collaborative preferences and generate personalized predictions from the collaborative information. In other words, these collaborative filtering-based approaches implicitly assume that similar users present analogous behaviors. For example, if user A and user B purchased several
identical items in the past, then user B’s recent interacted item would likely be favored by user A. By extracting valuable collaborative signals from historical behavior data, collaborative filtering-based algorithms can effectively make inferences for non-interacted items. There are two types of methods that are commonly used in collaborative filtering, which are referred to as memory-based methods [18] and model-based methods [7]:

Memory-based collaborative filtering algorithms, also known as neighbor-based collaborative filtering methods, are the earliest proposed collaborative filtering algorithms. These models focus on the precise similarity computation for individuals, and the quality of selected similar neighbors would largely impact the performance of the final prediction. Take the user-based collaborative filtering (UBCF) algorithm [19] as an example. The classical UBCF model adopts cosine similarity [20] on users’ ratings to compute the distance between a pair of users. Then, the k-nearest neighbor for each user can be easily identified, and the ratings rated by similar neighbors can be used to generate ratings for potential items. Since some users might prefer to give higher values while others might be more restricted in rating, the rating pattern mismatch among users would bias the similarity calculation. To this end, adjustment cosine similarity [20] and slope one [21] are proposed and applied to optimize the similarity calculation process in UBCF models.

Item-based collaborative filtering (IBCF) method [18] is another type of memory-based algorithm. Similar to UBCF, IBCF computes the similarity between a pair of items and relies on the received ratings of similar items to predict which user might consume the target item. Researchers [18] have proved that IBCF could produce more reliable predictions than UBCF since the item’s attraction for users would be more stable than the user’s shifting interests among items. In brief, memory-based collaborative filtering algorithms extract the collaborative signal from each individual’s similar neighbors. These methods are usually simple to implement, easy to explain, and also demonstrate acceptable performance on some public experimental datasets. However, when it comes to real-world recommendation scenarios (i.e., where the ratings or interactions data are sparse), it might be difficult for memory-based algorithms to find sufficiently similar neighbors, which leads to unsatisfied recommendation prediction [22].
2.1. OVERVIEW OF RECOMMENDER SYSTEMS

Model-based collaborative filtering algorithms rely on a well-trained model to analyze the user-item collaborative signal and generate recommendation predictions. Many machine learning [23] or data mining algorithms [24] are selected and adopted to construct the model-based collaborative filtering algorithms. Some examples of such recommenders include matrix factorization [25], decision trees [26], rule-based models [27], and bayesian methods [28]. Among all those algorithms, matrix factorization-based models have attracted the most research attentions and been widely applied to many recommendation scenarios. The basic assumption for matrix factorization-based recommenders is that there exists a latent space where similar individuals would be located nearby while dissimilar individuals would be far away. Hence, the corresponding individual representation in that space could naturally capture the valuable similarity information and would be ideal for generating personalized predictions. The paradigm of matrix factorization-based recommender consists of two steps, i.e., individual embedding learning and personalized preference modeling. Traditional matrix factorization-based recommenders such as probabilistic matrix factorization (PMF) [25] directly project an individual’s one-hot encoding to the corresponding low-dimensional embedding and employ dot-product to estimate the preference of a user given an item. Recently, with the superiority of neural-based [29] and graph-based models [30], more and more sophisticated neural-based and graph-based models have been employed in the embedding learning or relationship modeling steps. Compared to memory-based models, these model-based collaborative filtering algorithms are more suitable for real-world recommendation scenarios because they have advantages in dealing with scalable data. However, their expensive training and predicting computational costs still limit the flexibility of the recommendation and might result in unacceptable recommendation latency [31].

(2) Content-based recommender systems

As discussed above, collaborative filtering-based recommender systems merely rely on historical data to model the user-item relationship. However, the ignorance of user or item attributions could be wasteful. For example, if user A like the science fiction movie The Matrix, then it is very likely that user A would also be interested in a similar sic-fi movie,
such as Arrival. In other words, historical interaction is not the only type of data that proper for discover meaningful recommendations, the descriptive attributes for users or items are also quite valuable in modeling user-item relationships. From this perspective, content-based recommenders\textsuperscript{6} utilize the user or item attributes to calculate similarity, then match users to items that are similar to what they have purchased in the past. Generally speaking, content-based systems mainly includes the following components, which are feature extraction\textsuperscript{32}, user profile modeling\textsuperscript{33} and prediction.

**Feature extraction:** As discussed above, content-based recommender mainly focuses on extracting valuable information from descriptive attributes. However, descriptive attributes are not always available. In real-world recommendation cases, the content-based recommendation models often work with a wide variety types of input data, such as product review, news, music features and etc.. Considering this, most content-based recommenders employ feature extraction as their first step. Some content-based models adopt useful strategies to convert the rich input data into keyword-based attributes. Then, the vector-based representations\textsuperscript{33} for each individual can be obtained from the informative keywords-based attributes. Recently, with the development of deep learning techniques\textsuperscript{9}, more and more approaches have been proposed to generate vector-based representations directly from raw and unstructured textual data.

**User profile modeling:** After feature extraction, modeling the user profiles are also essential for generating recommendations in content-based recommenders. Different from collaborative filtering-based models which rely on the interactions of similar neighbors, content-based methods more target on constructing user-specific models which can explore items that have been purchased or rated by the target user. To achieve this, some content-based recommenders direct extract valuable attributes from user past interacted items, and then filter out items that have similar attributes\textsuperscript{34}. Moreover, other recommender\textsuperscript{35} implement more complex strategies, such as classification or regression algorithms, to model user’s preferences from the attributes of historical items and generate predictions based on the learned preference as well as the attributes of non-interacted items.

Compared to collaborative filter-based method, the advantage of content-based recom-
mender can be summarized as generating recommendation under new item cold-start scenarios. More specifically, collaborative filter-based methods which fully rely on historical data often fail to construct relationships between non-interacted new items and their target users. While, content-based recommenders count on the item attributes to model relationships and hence are able to recommend new item to potential customers even if new item does not obtain any interactions. Except for advantages, the disadvantages of content-based recommenders are also obvious. Concretely, content-based algorithms tend to filter out potential items that are similar to user’s historical interacted ones. In this case, the recommendation generated by content-based models often over-specialized and lack of novelty and serendipity. Low novelty and serendipity refers to the fact that user often receive predictions of high popularity and already-seen items. Failure in discovering new and surprisingly products would largely decrease the recommendation satisfaction of customers. Moreover, in most recommendation scenarios, detailed descriptions of items or users are often inapplicable but pure interaction data would be always available in every recommender systems. Hence, the application of content-based recommenders often be limited to specific scenarios, while collaborative filtering based models can be generalized on most recommendation cases.

Based on discussion above, both recommendation algorithms have their corresponding advantages and dis-advantages. To achieve better recommendation results, the hybrid methods which aim to combine the advantages among above two methods become the inevitable trend to build a recommender system. In fact, recommender systems in real world recommendation scenarios are rarely relied on isolated algorithms and often constructed by a combination of a variety of state-of-the-art models.

2.1.2 Type of input data

As the old Chinese saying goes "you can not make something out of nothing", except for the essential recommendation algorithms, the available data would also largely determine or influence the performances of given recommender systems. Corresponding to two types
of recommendation models, the input data of recommendation models can also be divided into two types: (1) historical interactions \[20\] and (2) side information \[34\].

(1) Historical interactions

As the only data resources used in collaborative filtering-based recommenders, the density and quality of historical behaviors often strongly impact the whole user-item relationship modeling process. There are mainly two types of historical interactions which are explicit and implicit interactions.

Explicit interaction often refers to rating which indicates the level of preference or dislikeness for the corresponding items. More concretely, there often exists a specified scale for the rating system and each rating given from the user would be strictly within the scope. Taken the most common 5-point rating scale as an example, the minimum and the maximum value might set for 1 and 5 respectively. In this case, a rating of 1 announces an extreme dislike, and a rating of 5 might reveal a strong affinity to the corresponding item. Meanwhile the value of rating can be either continuous or discrete, while in real-world recommendation scenarios, the value of ratings could be either continues or discrete. For example, the rating value can be any value between -10 to 10 in Jester joke recommender systems\[4\], while this is relatively rare in real world recommendation scenarios. In most rating systems in recommendation scenarios, discrete intervals are utilized to quantify user’s preferences. A quantitative example of explicit interaction is presented in Figure 2.1(a). In this explicit interaction matrix, there are total 6 users and 6 movies together with 14 ratings and 22 missing values, where ratings indicate the certain user’s preferences on corresponding items and missing value refers to unspecified user likeness. Generally speaking, a \(m \cdot n\) matrix are often adopted to represent explicit interaction data, where \(m\) and \(n\) refer to the number of users and items, respectively. Moreover, in real world cases, the quantity of \(m\) and \(n\) often reach numerous large values but ratings in the matrix are often sparse since the scoring for the consumed item seems useless and time-wasting for users and most customers rarely present their feedbacks.

\[4\]https://www.kaggle.com/datasets/crawford/jester-online-joke-recommender
### 2.1. OVERVIEW OF RECOMMENDER SYSTEMS

#### (a) Explicit Interaction Matrix

<table>
<thead>
<tr>
<th></th>
<th>I₁</th>
<th>I₂</th>
<th>I₃</th>
<th>I₄</th>
<th>I₅</th>
<th>I₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>U₁</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U₂</td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U₃</td>
<td></td>
<td>5</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U₄</td>
<td></td>
<td></td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U₅</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>U₆</td>
<td></td>
<td>5</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### (b) Implicit Interaction Matrix

<table>
<thead>
<tr>
<th></th>
<th>I₁</th>
<th>I₂</th>
<th>I₃</th>
<th>I₄</th>
<th>I₅</th>
<th>I₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>U₁</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U₂</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U₃</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U₄</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U₅</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>U₆</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.1: Examples of Historical Interactions

Except for explicit interaction, implicit interaction, which includes clicking, purchasing, long browsing and etc., is also adopted as an available historical data resources in recommender systems. Different from explicit interaction that aims to precisely depict user’s preference on an item, implicit interaction binarizes diverse behaviors into unitary positive preferences. An example of implicit interactions is presented in Figure 2.1(b). In this case, the $m \times n$ matrix generated by implicit interactions would be unary ones where 1 indicates the interaction from a user given an item demonstrating his or her potential likeness. While, miss value in implicit interaction-based matrix would also refer to user’s unspecified preference. Obviously, compared to explicit interaction, implicit interaction covers more types of user behaviors and could be automatically collected without any extra user participants. However, the rude pre-processing step that convert all different interaction directly into unitary positive preferences would be problematic. For example, users might first consume or click an product but finally found out that they are not interested in it. In this case, a low rating given from the user would straightforwardly demonstrate user’s dislike, but unitary preference converted by consuming or clicking behavior would mistakenly lead the recommendation model to treat the disliked item as a preferred ones. In summary, historical interactions are important for the preference mod-
eling in recommender systems. The utilization for either explicit or implicit interactions would be determined by certain recommendation scenarios.

(2) Side information

Side information is another group of valuable data resources that have been widely used by content-based recommenders.

The most intuitive type of side information is the flat feature, which refers to the detailed description of each individual. There are many flat features such as item attributes, product descriptions, user reviews, etc. To extract information contained in these flat features, many researchers implement state-of-the-art natural language processing algorithms on raw text-based or word-based content. Then, by incorporating the extracted information, more precise similarity among individuals can be computed, and better recommendation predictions could be generated. Similar to flat features, feature hierarchy is another type of word-based side information. Different from flat features, which mainly describe the characteristics of corresponding individuals, feature hierarchy often indicates the tag or category-based attributes that reveal the hierarchical structures among users or items. For example, the movie Frozen first belongs to the subgenre "Family Animation" and can be further categorized into the genre "Animation", demonstrating a hierarchical structure of "individual → sub-genre → genre". Analogously, users may present a similar hierarchy of "individual → occupation → age". Based on this, some methods build hierarchical recommenders using the hierarchical structure shown in feature hierarchies. Moreover, the network feature and the knowledge graph are other kinds of side information that have been frequently explored and studied in recommendation scenarios. The network feature implies network-based relationships among individuals. Take the microblogging and social networking platform Twitter as an example; the tweets liked by one user would very likely be favored by his or her followers, even if they haven’t interacted with or liked the same tweets before. Compared to the network feature, the knowledge graph would be a more general architecture that uses different types of semantic relationships to describe the connections among individuals and content. More concretely, knowledge graphs could provide complementary information within specific domains and
would be beneficial for most recommender systems.

Generally speaking, side information contains extra knowledge that might not be included in historical interactions. Hence, incorporating the side information would be beneficial for user-item relationship modeling and enable better recommendation predictions.

2.1.3 Training of recommendation models

Recently, more and more neural-based recommenders \cite{10, 2} have been developed to explore user-item relationships in exceedingly sparse and complicated data. As most neural-based models acquire a training process to finalize their hyper-parameters before conducting model inferences (i.e., generating final predictions), proper training would be the last but important step for constructing competitive recommenders. Meanwhile, it is worth pointing out that model training for exacting valuable knowledge from side information is beyond the scope of this thesis and will not be covered in the following discussion.

(1) Loss function

As the main guidance for the learning process, the loss function that evaluates how well the designed algorithm models the given input often largely determines the performance of the final inference recommender.

There are different loss functions that have been employed in the recommendation scenarios. The most commonly used one is mean squared error (MSE) \cite{41}. To be more specific, MSE loss guides the training process by minimizing the squared differences between the predicted values generated by the model and the true ratings in the training set. The corresponding equation for MSE is presented below.

\[
Loss_{MSE} = \frac{1}{RN} \sum_{i=1}^{RN} (y_i - \hat{y}_i)^2, \tag{2.1}
\]

where \( RN \) implies the number of ratings in the training set, \( y_i \) and \( \hat{y}_i \) indicate the ground truth and the predicted value of item \( i \), respectively. As shown above, MSE loss focuses
CHAPTER 2. LITERATURE REVIEW

on the restoration of preferences and is extensively applied to explicit interaction (i.e.,
ratings)-based recommenders. Binary cross entropy (BCE) [41] is another popular loss
function for the model training process in recommendation algorithms. The formally
deﬁned equation for BCE loss is demonstrated as follows:

\[
\text{Loss}_{\text{BCE}} = -\frac{1}{RN} \sum_{i=1}^{RN} (y_i \ln(\hat{y}_i) + (1 - y_i) \ln(1 - \hat{y}_i))
\] (2.2)

As shown in Equation (2.2), BCE loss calculates the predicted probabilities of two
classes and enables the designed recommender to distinguish the potential items from
non-interested ones. Considering that the binarized implicit interaction describes the in-
teraction behavior rather than a speciﬁc preference level, BCE loss would be a better loss
function than MSE loss in implicit interaction-based recommendation scenarios. Based
on the discussion above, it is obvious that both MSE and BCE focus on reconstruc-
ting pointwise feedback or interactions [41], while some researchers argue that the point-
wise feedback does not indicate much about the actual recommendation rankings seen by
users [14]. For this reason, a general optimization criterion, called bayesian personalized
ranking (BPR) [15], is proposed to capture the pairwise differences between users’ inter-
acted and non-interacted items. The following equation is used for calculating BPR loss:

\[
\text{Loss}_{\text{BPR}} = -\sum_{u=1}^{N} \sum_{i \in \mathcal{N}_u} \sum_{j \notin \mathcal{N}_u} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}),
\] (2.3)

where \( N \) implies for the number of users; \( \mathcal{N}_u \) refers to the set of items that user \( u \) have
interacted; \( \sigma(\cdot) \) is the activation function; \( \hat{y}_{ui} \) and \( \hat{y}_{uj} \) indicate the predicted preferences
user \( u \) given to item \( i \) and item \( j \) respectively. As shown in Equation (2.3), BPR loss guides
the model training process by maximizing the preference differences among user-preferred
(interacted) and dislike (non-interacted) items. Compared to pointwise loss, such as MSE
and BCE, pairwise BPR loss is able to capture the proﬁtable ranking knowledge and hence
would be more suitable for the training of ranking-based recommendation models.

Except for those basic training approaches, some newly proposed techniques have also been
incorporated into the model training process of some speciﬁc recommendation models. For
example, adversarial training has been incorporated into a multimedia recommender to
obtain a more robust inference model [42]. Moreover, contrastive learning, which aims to learn high-quality representation via a self-supervised manner, has been adopted to optimize cold-start problem [43]. In general, a proper and well-designed training strategy is essential for the success of final recommendation predictions. An effective, efficient and generalized loss function is often needed in recommendation training procedures.

(2) Negative sampling strategy

In the training process of most machine learning cases, balanced positive and negative samples are needed to obtain an unbiased inference model [44]. However, when it comes to real-world scenarios, collecting balanced data is nearly impossible for recommender systems. Take the dense and commonly used Movielens-1M dataset as an example. Nearly 80% of the ratings are positive (higher than the mean value of ratings), and roughly 20% of the ratings are negative. As for more sparse datasets, such as Amazon datasets, more than 90% of the samples are rated higher than the mean value. The imbalance problem becomes extremely severe with regard to implicit interaction-based recommendation scenarios. More specifically, all collected implicit interactions would be regarded as positive samples in those recommender systems, and no negative sample could be identified from the user’s historical data.

To balance the data in most recommendation scenarios, a negative sampling strategy that selects non-interacted items as negative samples is often implemented [45]. The most basic algorithm is the random negative sampling strategy (RNS) [46]. To be more specific, RNS treats each non-interacted in the sampling pool equally and samples with individual probability. RNS can avoid introducing new deviations in the sampling process and is widely applied in different recommendation cases. Moreover, popularity-biased sampling [47] and model-based sampling [45] are also popular sampling strategies widely used in the recommendation training process.

---

5https://grouplens.org/datasets/movielens/
6https://jmcauley.ucsd.edu/data/amazon/
2.2 Review on Recent Recommendation Approaches

Nowadays, with the rapid development in hardware devices, collecting abundant and comprehensive data becomes achievable in many real-world recommender systems. Those informative data undoubtedly provided more possibility for researchers to propose more sophisticated and accurate recommendation algorithms. However, the unneglectable fact is that those heterogeneous data make the recommendation problem much more challenging than before. Therefore, there is a desirable need to create more powerful theoretical methodologies to analyze and extract useful information from these accessible data and then improve the performance of recommender system.

To extract valuable information from massive heterogeneous data, more and more research attention has been drawn to deep learning-related techniques [48]. More specifically, deep learning algorithms adopt complex neural-based architectures to explore sophisticated relationships in massive input data and achieve competitive performances on different machine learning problems. There are many proposed deep learning architectures. A simple and basic example is the multi-layer perceptrons (MLPs) [49], which comprise several perceptron layers with activation functions. With the forward prediction and back propagation training strategies, MLPs are able to approximate any continuous function and can solve problems that are not linearly separable. Another representative example of deep learning models is convolutional neural networks (CNNs) [50]. Concretely, CNNs consist of multiple convolutional and pooling layers and are mainly used for image processing or object detection tasks. With the ability to reduce the quantity of parameters without losing the quality of models, CNNs model has demonstrated astonishing prediction accuracy on image classification tasks. Moreover, recurrent neural networks (RNNs) [51] that are designed for processing sequential data, autoencoders (AEs) [52] that are superior at dimension reduction, and etc., are also outstanding deep learning models that have been widely explored and implemented on structured data. However, when it comes to some real-world cases where unstructured data is collected and provided, relying on the above models would not be an ideal choice. To this end, graph convolutional networks (GCNs) [53] that organize heterogeneous data with graph structure have gained more and
more research attention recently. More specifically, with the ability to iteratively update node representation with the aggregated information from its connected neighbors, GCNs models demonstrate outstanding performances in embedding learning tasks.

As discussed above, recommender systems are facing the challenges of processing complex and heterogeneous data while neural-based models are superior at embedding learning and relationship modeling tasks. Thus, integrating the advantages of neural-based models into recommender system to generate better recommendation results has become an typical and popular research direction. In the following section, a review for recent approaches on neural-based recommender are provided.

2.2.1 Neural-based collaborative filtering

To generate personalized predictions, collaborative filtering-based algorithms focus on extracting collaborative preferences from historical interactions. Although traditional memory-based methods and classical machine learning algorithm-based recommenders could achieve acceptable performances on some experimental datasets when it comes to real-world scenarios where the historical data becomes increasingly sparse and large, utilizing traditional and classical methods are still insufficient for exploring valuable personalized information and modeling complex user-item relationships.

For this reason, Salakhutdinov et al. [54] firstly propose a class of two-layer undirected graphical models, called restricted boltzmann machines (RBMs), to model tabular data and achieve outstanding performances. Recently, with the tremendous success of neural-based models for computer vision and language processing tasks, more and more neural-based collaborative filtering algorithms suitable for recommendation problems have been developed. For example, Sedhain et al. [55] present a novel autoencoder framework, called AutoRec, which can take partly observed ratings as input and then reconstruct them in the output to predict missing ratings for purposes of recommendation. Strub et al. [56] extend the AutoRec and perform collaborative filtering using stacked denoising autoencoders (SDAE). Results show that the proposed methods demonstrate excellent experimental
results compared to benchmark algorithms. Wu et al. [57] also utilize a collaborative
denoising autoencoder (CDAE) to generate the preference of a user given an item from
corrupted ratings. In particular, by adding a user node to the proposed denoising au-
toencoder’s input layer, CDAE could automatically encode a latent vector for each user
and conduct more personalized recommendations. Moreover, Zhuang et al. [58] propose a
collaborative ranking framework that adopts an autoencoder to learn user and item latent
embeddings simultaneously. More concretely, by training with the pair-wise ranked loss,
the proposed REAP model can exhibit competitive performances in dealing with data
sparsity problems.

Many other deep neural architectures are also adopted to construct recommender systems.
Neural matrix factorization (NeuMF) proposed by He et al. [2] is one of the state-of-the-
art neural-based methods. More specifically, He et al. [2] replace the simply dot product
in user-item relationship modeling process with more sophisticated MLPs architecture.
Then, with the high level of non-linearities endowed with MLPs, the proposed neural
collaborative filtering (NCF) model could model non-linear pairwise relationships. Based
on the proposed NCF, a more generalized framework, i.e. NeuMF are presented to extract
both linear and non-linear information among user-item pairs. The detailed architecture
of NeuMF is presented below.

As shown in Figure 2.2, NeuMF generates final predictions by fusing general matrix fac-
torization (GMF) with the MLPs-based NCF. In other words, separate user (item) em-
beddings would first be inserted into two models. After that, by concatenating the last
layers of two models, the fused NeuMF can capture linear and non-linear relationships
simultaneously, leading to outstanding recommendation performances. Subsequently, He
et al. [59] further extend the basic NCF models and propose a convolutional neural collab-
orative filtering (ConvNCF) model. In the proposed ConvNCF, the user-item interaction
map is firstly computed from the projected user (item) embeddings. Then, by insert-
ing more informative interaction map into CNNs architectures, ConvNCF can achieves
better and more stable performance. Moreover, Song et al. [60] propose a pairwise col-
laborative ranking framework, called neural network-based collaborative ranking (NCR),
2.2. REVIEW ON RECENT RECOMMENDATION APPROACHES

Figure 2.2: An illustration of NeuMF Model.

that utilizes the neural-based architecture to model a user’s preference among a pair of items. Compared to pointwise models, pairwise NCR exhibits superior performances in ranking-related recommendation scenarios.

Apart from the above approaches which utilize neural-based architectures to model sophisticated user-item relationships, implementing neural-based model to explore valuable information and encoding the extracted information into personalized embeddings is another direction for constructing collaborative filtering-based recommenders. An representative example is deep matrix factorization (DMF) proposed by Xue et al. [40]. The neural-based architecture of DMF is demonstrated as follows.

As presented in Figure 2.3, DMF takes full matrix (including explicit ratings as well as implicit interactions) as input and uses parallel architectures to encode low dimensional user (item) embeddings. By training with the proposed loss, DMF exhibits competitive performances, confirming the superiority of neural-based architectures in exploring and encoding personalized information. Moreover, considering the ability of neural-based ar-
architecture in both embedding learning and relationship modeling tasks, constructing more sophisticated that incorporating neural-based methods in two steps becomes a naturally research direction. For example, Cheng et al. [61] propose a dual-embedding-based deep latent factor (DELF) model. In the embedding learning process of DELF model, an extra pair of user-item embeddings is generated by an attentive neural method from interaction vectors. Then, four MLPs-based architectures would be further implemented to model the interactions among dual user-item embeddings. Furthermore, Deng et al. [62] propose a more general framework, called deep collaborative filtering (DeepCF), that combines
the state-of-the-art neural-based embedding learning method and neural-based pairwise relationship modeling method. Results show that DeepCF can achieve exceptional performances in prediction accuracy evaluations.

Although above approaches have demonstrated remarkable performances, the mismatch among the structured input required by those neural-based architectures and the heterogeneous data collected from real-world recommendation scenarios still largely limit further improvements of those recommenders. In fact, user-item interactions in a recommender system could be inherently displayed by a bipartite graph architecture. Hence, a graph-based algorithms would be naturally more suitable for exploring user-item relationships in recommendation scenarios. There are several early attempts to construct graph structure-based recommenders. For example, Konstas et al. [63] produce recommendation predictions by adopting random walk algorithms in a click-through graph. Moreover, Li et al. [64] propose a graph kernel-based approach on the user–item pair and use the graph structure to infer whether a user may have a link with an item. The implementation of these models confirm the effective of incorporating graph architecture into recommender systems. While, their simple and direct graph learning algorithms are still insufficient and inappropriate for more sparse and complex recommendation problems.

Recently, GCNs [53] have attracted more and more research attention owing to their remarkable performances in handling heterogeneous data. Based on this, Berg et al. [65] propose a GCNs-based matrix completion framework (GC-MC) that can produce user (item) embeddings through a form of message passing on the bipartite interaction graph. Ying et al. [66] argue that operating convolution on the entire graph architecture would be computationally expensive and time-wasting. Hence, they developed a GCNs-based algorithm called PinSage, which combines efficient random walks and graph convolutional operations to generate user (item) embeddings. Wang et al. [67] propose a more general GCNs-based framework, i.e., neural graph collaborative filtering (NGCF), which formulates the graph convolution on more commonly established collaborative filtering tasks. Following the design of classical GCNs, the core point for implementing graph convolution operations on NGCF is the designed embedding propagation layer. More concretely, the
essential embedding propagation layer consists of two operations: message construction and message aggregation. Message construction is designed for passing valuable messages from the connect items (users) to the target user (item) node. Message aggregation combines the information retained in the original user (item) node with the messages propagated from connected neighbors to generate updated user (item) embeddings. Then, by stacking multiple embedding propagation layers together, the valuable collaborative signal stored in the high-order connections could be encoded into user (item) final embeddings. After that, the personalized recommendation predictions can be produced by the dot product among a pair of embeddings. NGCF represents an novel attempt that can exploit high-order collaborative signals by the message-passing mechanism in model-based collaborative filtering. Although NGCF achieves outstanding performances, lacking thorough ablation analyses on GCNs and simply cloning the basic GCNs algorithms which designed for classification tasks to construct recommendation models would be sub-optimal.

To this end, He et al. [3] further simplify the design of GCNs to make it more concise and appropriate for recommendation. Specifically, He et al. perform extensive ablation studies and point out that the two most common designs in GCNs, i.e., feature transformation and nonlinear activation, contribute little to the performance of collaborative filtering. Motivated by this, a new GCNs-based model, called LightGCN, that only includes neighborhood aggregation step of basic GCNs-based algorithm is proposed. The detailed architecture of LightGCN is presented in Figure 2.4. As shown in Figure 2.4, LightGCN adopt simple weighted sum aggregator and abandon the use of feature transformation and nonlinear activation to perform graph convolutional operation on each graph layer. Meanwhile, the final embedding of user (item) can be calculated by summing over the refined embeddings at each layer. Extensive experiment results demonstrate the simplified LightGCN is easier to train, better in generalization, and more effective for produce recommendations. Mao et al. [68] further simplify the LightGCN by skipping infinite layers of message passing. To be more specific, they propose an ultra-simplified formulation of GCNs, called UltraGCN, which can directly approximate the limit of infinite-layer graph convolutions and enable better connection modeling among complicated user-item relationships.
2.2. REVIEW ON RECENT RECOMMENDATION APPROACHES

Furthermore, Sun et al. [69] pointed out that directly applying GCNs to process the bipartite graph is sub-optimal since this method neglects the intrinsic differences between user nodes and item nodes. Hence, they proposed NIA-GCN, a new framework that explicitly models relationships between neighboring nodes and exploits the heterogeneous nature of the user-item bipartite graph. Liu et al. [70] mentioned that existing GCNs-based recommenders often suffer from the over-smoothing problem. To alleviate that problem, they presented a recommendation model, called IMP-GCN, which performs high-order graph convolution inside subgraphs and hence limits the neighbor exploration process to similar users (items). Wu et al. [71] explored self-supervised learning on GCNs-based recommenders and targeted improving their recommendation accuracy and robustness.
2.2.2 Neural-based content-based recommenders

As shown in Chapter 2 Section 2.1.1, content-based recommendation models take side information as well as historical interactions as their inputs. Then, proper feature extraction and profile modeling algorithms are used to generate predictions. Recently, with the rapid development in computer hardware, more and more detailed side information is collected and provided in content-based recommenders, bringing big challenges for existing content-based algorithms. In other words, being unable to extract information in detailed side information and incapable of capturing complicated relationships among extracted features become the two main problems that highly limit the performance of current content-based recommenders. To this end, many researchers try to enhance the current content-based recommenders by utilizing or incorporating neural-based architectures. Some of recent neural-based approaches are presented and summarized below.

Early attempts neural-based content algorithms often focus on incorporating new types of information which is unusable for traditional content-based recommenders. For example, Liu et al. \cite{72} propose a DeepStyle method to learn style features of item based on its visual information (image). Similarly, Alashkar et al. \cite{73} construct a neural-based model which makes use of the pairwise of Before-After images and the makeup artist knowledge jointly. Vall et al. \cite{74} design a song-to-playlist classifier which could learn the underlying structure from actual playlists by leveraging song features derived from audio, social tags and independent listening logs. Except for visual and audio-related contents, text-based data are also profitable for extracting valuable knowledge. For instance, Khattar et al. \cite{75} a CNNs-based model for news recommendation which utilizes the content of the news articles as well as the sequence of articles that read by the user. Another impressive achievement is proposed by Hu et al. \cite{76}, they build a multilevel attraction model (MLAM) for movie recommendation over the content features including textual data and categorical data of movies. More concretely, they utilize a multilevel neural model with attention mechanism on text content to capture the user attractions (representations) at word-level, sentence-level, and story-level. Extensive experiment results confirm the effectiveness and merits of MLAM.
Although detailed auxiliary information on items is valuable for constructing content-based models, it is still worth pointing out that they are often not sufficient or not even available in most recommendation cases. Based on this, users’ review become another common resource for neural-based architectures to explore informative features. Zhang et al. [77] present a deep cooperative neural networks (DeepCoNN) that can explore and extract the profitable information stored in reviews. More concretely, DeepCoNN utilizes two parallel neural-based architectures to learn user and item embeddings based on rich review contexts. Then, by coupling the learned embeddings with an extra shared layer, the complex user-item relationships can be precisely modeled. As the first model which jointly models both user and item from reviews using neural-based architecture, DeepCoNN achieves outstanding performances compared to other baselines. Similarly, Seo et al. [78] also adopt neural-based architecture to model user preferences and item properties from reviews. Different from DeepCoNN, which directly applies classical CNNs architectures to generate embeddings, Seo et al. [78] believe both local words, as well as global semantic meaning, would be essential for exploiting valuable information. To this end, a dual attention-based CNNs (D-Attn) model is proposed to generate user and item embeddings from aggregated review texts. Moreover, Catherine et al. [79] claim that the DeepCoNN model can only generate the predictive value of items that the user has previously reviewed, but it performs poorly on potential items that the user might be interested. Based on this, they propose transformational neural networks (TransNets), which expand the DeepCoNN model by providing an extra latent layer to represent an approximation of the unavailable review and allowing predictions even when the target user’s review for the target item is not provided. Furthermore, Tay et al. [80] present a review-by-review pointer-based learning scheme that extracts important reviews from the entire review pool and subsequently matches them in a word-by-word fashion. Hence, only a few important reviews would be selected in each calculation, and the experiment results confirm its superiority over the above three. Analogously, Chin et al. [81] also believe that not all parts of each review are equally important. They propose an aspect-based neural recommender (ANR) to perform aspect-based representation learning for reviews via an attention-based component. Deng et al. [82] exhibit another model, called the neural gaussian mixture
CHAPTER 2. LITERATURE REVIEW

model (NGMM), for review-based rating prediction tasks which utilize the gaussian layer to compute the user’s preference of an item based on the embeddings learned from reviews. Wang et al. [83] extend CNNs by introducing a new module called deep latent Dirichlet allocation (DLDA) to capture the contextual features of reviews. Extensive experimental results demonstrate that the proposed NGMM model achieves competitive performance in review-based rating prediction tasks.

Except for the above approaches that focus on extracting informative features from auxiliary contents, employing neural-based architecture to enhance the profiles modeling is another popular direction for constructing neural-based content recommenders. Researchers [84] from Google present a Wide & Deep model, which combines a linear model with fully connected MLPs to generate recommendations from diverse features. Specifically, researchers utilize deep MLPs architecture to capture generalized feature combinations and adopt a linear model to avoid over-generalized prediction. Through joint training, the Wide & Deep model can enable both memorized and generalized feature modeling in content-based recommendation cases. Wang et al. [85] claim that identify effective feature interactions is key to the success of many content-based recommender. However, when it comes large set of sparse and dense features, the effective feature interactions would be difficult. Hence, they propose a deep and cross network (DCN), which can learn explicit cross features of bounded degree jointly with traditional deep representations. Results show that the proposed DCN can outperform the state-of-art algorithms on both sparse and dense datasets, in terms of both model accuracy and memory usage. Zhang et al. [86] propose to leverage three feature transformation methods, i.e., factorization machines (FMs) [87], restricted Boltzmann machines (RBMs) [88] and denoising auto-encoders (DAEs) [89] to effectively and efficiently predict users’ ad click response based on multi-field categorical features. In fact, FMs, the supervised learning approach which can enhances the linear regression model by incorporating the second-order feature interactions, is a useful feature modeling technique that has been widely implemented in content-based recommenders. Despite its effectiveness, simple and direct FMs becomes insufficient for recent real-world recommendation data. To this end, Qu et al. [90] propose a product-based neural networks (PNN) with an embedding layer to learn a distributed
representation of the categorical data, a product layer to capture interactive patterns between inter-field categories, and further fully connected layers to explore high-order feature interactions. Xiao et al. [91] present the attentional factorization machine (AFM) to the importance of each feature interaction via a neural attention network. Moreover, Guo et al. [92] propose a DeepFM model to capture the high order feature interactions. More concretely, DeepFM utilizes the FMs component to model the vector-wise feature interactions and the neural-based architecture to model the complex high-dimensional bit-wise feature interactions. Lian et al. [93] present an extreme deep factorization machine (xDeepFM) which can learn high order feature interactions through a proposed compressed interaction network (CIN). To be more specific, xDeepFM formulates the output of filed embeddings as a matrix and utilizes the matrix to build an interaction feature map. Then, the CIN model will take the learned interaction feature map as input and generate the final predictions. Comprehensive experiment results demonstrate that xDeepFM outperforms state-of-the-art models consistently on three real-world datasets. Another approach of neural-based FMs is proposed by He et al. [94]. They present a novel model, called neural factorization machine (NFM), which combines the linearity of FMs in modeling second-order feature interactions and the non-linearity of neural network in modeling higher-order feature interactions. Apart from the above approaches, graph-based architectures are also implemented to enhance FMs owing to their superiority. Li et al. [95] propose a graph factorization machine (GraphFM) by naturally representing features in the graph structure. In particular, a novel mechanism is designed to select the beneficial feature interactions and formulate them as edges between features. Then, by integrating the interaction function of FMs into the feature aggregation strategy of the graph structure, any arbitrary-order feature interactions can be modeled on the graph-structured features.

There are also many discussions for constructing neural-based hybrid models. For instance, Deldjoo et al. [96] utilize a neural-based architecture to extract valuable features from the visual and audio channels of a video. Then, the extracted features are incorporated with state-of-the-art recommendation models to generate final recommendations. Moreover, Shi et al. [97] firstly design a meta-path-based method to capture the aspect-level latent factors of users and items from heterogeneous information networks (HIN).
Tremendous experimental results confirm the effectiveness of the proposed NeuACF. In general, with the success of neural-based models in constructing collaborative filtering and content-based recommenders, utilizing neural-based architecture to construct hybrid recommenders naturally becomes a common and effective option. At the same time, most neural-based hybrid approaches adopt similar strategies, which are already presented and discussed in the above collaborative and content-based methods. Hence, the extensive discussions for neural-based approaches on constructing hybrid recommendation models will not be included.

2.2.3 Discussions on recent neural-based recommenders

As presented above, neural-based recommenders achieve exceptional and competitive recommendation performances. while it is still worth pointing out that there are still several limitations that restrict the ability of current neural-based recommenders.

(1) Neglecting inherent user (item) hierarchies

Most existing neural-based recommenders only focus on modeling individual items or users independently, which cannot fully capture the items’ (or users’) hierarchies information and hence fail to precisely model personalized preference. Take the movie Frozen as an example, it belongs to the subgenre 'Family Animation’ and can be further categorized into the genre 'Animation", demonstrating a hierarchical structure of 'individual \rightarrow sub-genre \rightarrow genre’. Similarly, users may present a similar hierarchy of 'individual \rightarrow occupation \rightarrow age’. Since items of the same subgenre (or genre) are likely to have similar attributes, they are likely to acquire similar preferences [12]. Thus, hierarchical information about items or users can very likely improve the preference modeling process of neural-based recommender systems. Although some recent content-based approaches [98] utilize HIN to exploit hierarchies from explicit user (item) hierarchical structures, such as categorical tags. It is worth pointing out that in real-world recommendation scenarios, explicit hierarchies are often not applicable [13]. For this reason, some researchers have used MF-based models to learn implicit item/user hierarchies [13]. Specifically, the implicit item (user) hi-
2.2. REVIEW ON RECENT RECOMMENDATION APPROACHES

Hierarchies can be easily obtained by decomposing the original item/user embedding matrix into several smaller and more compressed item (user) matrices, respectively. The success of these models confirms the validity of exploring implicit hierarchies [12]. However, the simple MF-based model is obviously insufficient for increasingly complex and sparse recommendation scenarios. The neural-based hierarchical recommenders which can effectively exploit user (item) implicit hierarchies still requires more research concerns in future studies.

(2) Insufficient data modeling and inadequate model training

Current attempts for constructing neural-based recommenders merely involve implementing new deep architectures or incorporating more auxiliary information but rarely exploring new data modeling and model training methods that largely determine the final inference performance, leading to insufficient utilization of available information and sub-optimal predictions of neural-based models. More specifically, existing neural-based recommenders utilize ordinary pointwise or pairwise training strategies to fine-tune complex neural-based architectures. The pointwise strategy (such as MSE and BCE) does not indicate much about the actual recommendation rankings seen by users and hence is still insufficient for real-world recommendation scenarios [14]. The pairwise strategy (such as BPR) can only be implemented on binary implicit feedback, which describes the users’ historical behaviors rather than the degree of preference, given the possibility of recommending items that the user does not like. To this end, better data data modeling and model training strategies that can comprehensively extract valuable ranking information with the subtle preference degrees are still need.

(3) Large computational cost in training and inference procedure

As a larger model with more learning parameters has a higher expression capacity and generally obtains better recommendation performance, the size of recent neural-based recommender systems has kept rising with respect to the continuously increasing amount of available data. However, endlessly increasing the model size would be problematic, leading to huge memory and computation costs. These memory and computation costs not
only slow down online inference and hinder their adoption in real-time recommendation
applications but also limit the models’ capability to capture shifting user (item) interest
and results in outdated predictions. Hence, how to achieve the outstanding recommenda-
tion performance yet still retain the light model architecture become another inevitable
concern for constructing neural-based recommender systems.

2.3 Review on Knowledge Distillation for Recommender
System

With the rising model complexity of proposed neural-based recommenders, the high in-
fERENCE latency becomes an unavoidable problems. To this end, KD, a model compression
and knowledge transferring technique [17] [16], has been adopted in recommender systems
to reduce model inference latency without a significant performance drop.

To be more specific, KD [17] describes the process of teaching a compact student model
to mimic a large cumbersome teacher model by distilling knowledge from the teacher,
which enables a compact model to obtain competitive performance and model efficiency
simultaneously. According to [16], a KD-based system is composed of three key compo-
nents: knowledge, teacher-student architecture, and distillation scheme. Knowledge refers
to the available information that can be distilled from the cumbersome teacher. There
are different types of distilled knowledge including the logits of the output [99], the repre-
sentation of hidden layer [100], and the relationships among groups of samples [101].
Teacher-student architecture can be regarded as a carrier that enables the information
transition among models. Standard KD-based system adopts distilling from teacher to
train student, while recent studies [102] demonstrates the potential of students, and the
dual training paradigm has been introduced to allow flexible knowledge transfer between
teacher and students. Meanwhile, distillation schemes are also essential for the success of
the knowledge distillation system since it reveals how can the distilled knowledge from the
teacher model guide the student model to obtain better performance. After introduced
and formally popularized by Hinton et al. [17], KD has been extensively researched and
2.3. REVIEW ON KNOWLEDGE DISTILLATION FOR RECOMMENDER SYSTEM

implemented in a variety of application domains, including speech recognition [103], image processing [104], recommender system [105], etc..

In the follow section, a review for applying KD on recommendation models is presented from three perspectives, including distilled knowledge type, teacher-student architecture, and distillation scheme.

2.3.1 Distilled knowledge in recommender systems

There are mainly three different types of knowledge that can be distilled and transferred in recommendation scenarios, which are prediction-based knowledge, representation-based knowledge and relationship-based knowledge. An intuitive example of different types of knowledge within a GCNs-based teacher model is shown in Figure 2.5.

![Figure 2.5: An intuitive example of different types of knowledge within a teacher model](image)

(1) Prediction-based Knowledge

In classical knowledge distillation cases, prediction-based knowledge refers to the numerical
logits of the teacher model’s output layer [99]. The most widely accepted prediction-based knowledge is the soft target in the image classification problems [17], which reveals the probabilities of the input belonging to the classes. With the ability to capture dark knowledge learned by the teacher model, some researchers [106] treat the soft target in the recommender system as the potential preference of non-interacted items generated by the teacher model and teach the student model to mimic these soft target predictions to achieve better performance. Although soft target could contain some valuable pointwise information, it might be still insufficient for real-world recommendation scenarios since the pointwise feedback does not indicate much about the actual recommendation ranking that the user sees. From this perspective, other researchers [107] propose to distill the ranking information in the teacher model’s prediction list and aim to guide the student in generating the same ranking list as the teacher. The detailed discussions for the mentioned two kinds of prediction-based knowledge are presented below.

**Soft Target** As mentioned above, soft targets are defined as the probabilities that the input belongs to the classes. Specifically, by adopting a softmax function on final output of the model, the soft target can be calculated through:

\[
p(z_i, T) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)},
\]

(2.4)

where \(z_i\) denotes the estimated logits for the \(i\)-th class, and \(T\) is the temperature factor designed to control the importance of each soft target. Compared to binary class labels, the soft target could also capture valuable dark knowledge from the teacher model [16]. Hence, by taking discrepancies among soft targets generated through the teacher and student models into consideration, the student model would easily obtain better performance. In the field of recommender systems, the task of click-through rate (CTR) [108] prediction that aim to estimate the individual’s next clicked item is quite similar to the traditional classification problem. Based on this, Wang et al. [109] and Zhu et al. [110] directly adopt the soft target strategy proposed in Hilton et al. [17] to guide the training of the student model. The objective function for these CTR prediction models can be summarized as follows:

\[
Loss = Loss_{CE}(y_i, \hat{y}_i) + Loss_{KD}(p^t(z_i, T), p^s(z_i, T)),
\]

(2.5)
2.3. REVIEW ON KNOWLEDGE DISTILLATION FOR RECOMMENDER SYSTEM

where \( \text{Loss}_{CE}(\cdot) \) denotes the cross-entropy loss between the predicted value \( \hat{y}_i \) and the ground truth \( y_i \); \( \text{Loss}_{KD}(\cdot) \) indicate the discrepancy of two soft targets including \( p^t(z_i, T) \) learned by the teacher model and \( p^s(z_i, T) \) generated by the student model. Cheng et al. [106] points out that the soft target directly computed from the teacher model might be noisy and could mislead the student model’s learning. They smooth teacher output with the one-hot ground truth vector to obtain the final soft target. When it comes to more generalized recommendation scenarios where each prediction list contains multiple target items, simply adopting the softmax function to calculate the soft target would be problematic. To this end, Pan et al. [111] utilizes a logistic function with the popularity enhanced temperature function to compute the final soft target of each output unit in the teacher model. The soft targets can be calculated through:

\[
p(y^*_{u_i}, T(\cdot)) = \frac{1}{1 + \exp(-y^*_{u_i}/T(\text{pop}_i))},
\]

(2.6)

where \( y^*_{u_i} \) is the user \( u \)'s predicted preference on item \( i \) and \( T(\cdot) \) is the temperature function that can be calculated based on the item \( i \)'s popularity \( \text{pop}_i \). Lee et al. [112] adopts a similar soft target calculation strategy with a ranking-based sampling method to select transferred soft targets. Furthermore, Pan et al. [113] and Kweon et al. [114] simplify the soft target calculation process and directly treat the teacher model’s final predictions as distilled knowledge that student model should mimic. In brief, as the most popular prediction-based knowledge, soft target is frequently adopted as knowledge that can be distilled in recommender system. Different calculation methods are proposed to generate proper soft target according to corresponding recommendation scenarios.

**Ranking Information** Except for the soft target, ranking information is another type of prediction-based knowledge, which can be distilled and transferred in recommender systems. Tang et al. [107] point out that the ultimate goal of the recommendation models is to predict the relative order of items rather than generate a label as in classification. To this end, they propose to train the student model \( M_S \) to mimic the top@K predictions of its teacher \( M_T \). Specifically, the corresponding distillation process can be summarized as:

\[
\text{Loss}_{KD}(\pi_{1..K}, \hat{y}^*) = -\sum_{r=1}^{K} w_r * \log(\sigma(\hat{y}^*_{\pi_r})),
\]

(2.7)
where $\pi_{1..K}$ indicates the top@K items that predicted by the teacher model; $\hat{y}^s$ is the predicted preference generated by student model; $\hat{y}^s_{\pi_r}$ reveals the potential preference of item $\pi_r$ computed by the student model; $\sigma(\cdot)$ represents the sigmoid function; and $w_r$ is the weight for $r-th$ ranked item in $\pi_{1..K}$ that can be calculated by the ranking discrepancy between the student-predicted rank and the teacher-predicted rank. It is worth noticing that Tang et al. [107] merely focus on top@K ranked documents and neglects the negative samples. Wang et al. [115] propose a ranking information distillation strategy that can consider the distribution for both positive and negative samples. More specifically, the author utilizes cross-entropy to ensure the order of positive(or the negative) items in both teacher and student predictions become approximately the same. While, separately ranking the positive and negative is still sub-optimal since the ultimately goal for recommendation is to generate an overall ranking list among all samples. Considering the numerous non-interacted items, Lian et al. [116] present a more efficient distillation strategy which randomly samples $L$ negative items for each positive instance and aims to preserve the relative order of the negative samples in the whole list. Furthermore, Kang et al. [117] [118] specify that the ranking position for positive items are more important than those uninterested negative items. Hence, they present a relaxed ranking distillation (RRD) that is able to match the teacher and student recommendation lists by focusing on the precise rankings of potential items and ignoring the detailed orders among the uninteresting items. In specific, RRD samples $K$ interesting items and $L$ uninteresting items for each user, where the interesting items are chose from a narrow range near the top of the teacher model’s prediction list whereas the uninteresting items are selected from a set of items that have lower rankings than the interesting items [117]. The final ranking distillation loss based on the relaxed permutation probability is formulated as follows:

$$\text{Loss}_{KD}(\pi_{1:K}, \hat{y}^s) = -\sum_{r=1}^{K} \log\left( \frac{\exp(\hat{y}^s_{\pi_r})}{\sum_{i=r}^{K} \exp(\hat{y}^s_{\pi_i}) + \sum_{j=K}^{K+R_S} \exp(\hat{y}^s_{\pi_j})} \right),$$

(2.8)

where $R_S$ indicate the number of randomly selected uninterested items. It is apparent that the Equation (2.8) aims to train the student model to maintain the teacher model’s rankings of interested items and ignore the detailed orders among the uninteresting ones. Furthermore, Kang et al. [118] and Lee et al. [119] point out that merely relying on user-
side ranking information would limit the student’s generalization power and hence lead to insufficient knowledge transition. To this end, they adopt a prediction discrepancy-based sampling strategy on both user-side and item-side ranking lists. Then, the user-side distillation and the item-side distillation can be conducted by maximizing relaxed permutation probability on the sampled user list and the sampled item list, respectively. Similarly, Kang et al. [118] propose an item-side regularization method that enables the student model to extract ranking information from the user-side and item-side simultaneously. Except for the above approaches, Hofstätter et al. [120] provide another direction to distill valuable ranking information. Concretely, they adopts Margin MSE approach to ensure the score difference between the relevant and non-relevant samples are approximately the same in both teacher and student model.

To conclude, the soft-target reveals the point-wise potential preference and the ranking information exhibit list-wise preference order of non-interacted items. Both of these prediction-based knowledge are valuable information that can be incorporated in training the student model leading to better student model performance. However, the prediction-based knowledge solely focuses the final prediction and thus fails to supervise the learning of user (item) embeddings, which plays an important role in capturing user (item) personalized preference information.

(2) Representation-based Knowledge

As mentioned above, predictions-based knowledge is insufficient for transferring the teacher’s knowledge and hidden-representation of intermediate layer would be a good extension for knowledge distillation in recommender system. In fact, the idea of distilling the intermediate representation is introduced and formulated as hint regression by Romero et al. [100]. Specifically, by directly matching the hint (i.e. the representation of hidden layer) among the teacher and student model, the latent information encoded in hidden layer representations can be easily transferred and guided the student to obtain better performance. The success of hint regression strategy in computer vision tasks triggers its application in recommendation scenarios. The detailed discussions are presented below.
To better explore the representation-based knowledge distillation, the analysis of basic hint regression strategies are first presented, following with their corresponding modifications and extensions on recommendation cases. Let $T$ denotes the teacher model which maps the input feature into a low dimensional representation $e^t$. Analogously, $S$ refers to the similar student model that can generate hidden-layer representation $e^s$. The hint regression strategy aim to approximate $e^s$ to $e^t$ and the corresponding distillation scheme is presented below:

$$
Loss_{KD} = ||e^t - f(e^s)||_2^2,
$$

(2.9)

where $f(\cdot)$ denotes an function which project the hidden representation $e^s$ to a comparable dimensional with $e^t$. By jointly training with the distillation loss $Loss_{KD}$ and the original loss $Loss$, the student model could demonstrate competitive performances with less inference latency. Liu et al. [121] directly adopt the hint regression strategy in counterfactual recommendation scenarios to train the representation learned from the non-uniform data based on the embedding generated from the valuable uniform data. Analogously, Xie et al. [122] propose a confidence-guided intermediate distillation which not only calculates direct discrepancy among hidden layer representations but also computes the confidence of each instance to further improve the distillation process. Moreover, Xu et al. [123] minimize the difference among incrementally learned user (item) embeddings since they believe the currently generate user (item) embedding should not move too far from its previous position. Except for the direct discrepancy calculation among $e^t$ and $e^s$, in some recommendation scenarios, the different model complexity of the teacher and the student might lead to the mismatched sizes of hidden-layer representations. To this end, Zhu et al. [110] use a transformation matrix that convert representation $e^s$ to the same size of $e^t$, while Chen et al. [124] apply the a learn-able projection function $f(\cdot)$ on $e^s$ to match the dimension of $e^t$. Meanwhile, Kang et al. [117] argue that merely rely on a single projection to expand the compressed representations $e^s$ would degrade the quality of distillation since it easily mixtures the preference of dissimilar entities. Hence, they adopt a novel expert selection strategy that utilizes teacher knowledge to cluster representations into $K$ different expert groups and then transfers the representations in each group via an independent
2.3. REVIEW ON KNOWLEDGE DISTILLATION FOR RECOMMENDER SYSTEM

The corresponding distillation strategy can be summarized as follows:

$$Loss_{KD} = ||e^t_i - \sum_{k=1}^{K} ohe_{i_k} f_k(e^s_i)||^2_2,$$  \hspace{1cm} (2.10)

where $e^t_i$ and $e^s_i$ denote the representation of entity $i$ learned by the teacher and the student model, respectively. Moreover, $ohe_{i_k} = 1$ stands for the $i$–th entity that belongs to the $k$–th expert group. To generate the differentiable one hot encoding $ohe_i$, Kang et al. [117] first adopt a selection network SNN to calculate the normalized categorical distribution of $i$–th entity for $K$ different expert group based on the representation $e^t_i$. Gumbel-Softmax [125] is then utilized to generate the approximated $ohe_i$. In sum, with the designed expert selection strategy, the representations that contain dissimilar preference information would be transferred through different projection to enable more subtle and precise knowledge distillations. The most recent work PHR [105] further elaborates this approach and employs a personalization network to enable a personalized distillation. To be more specific, PHR utilizes neighborhood information to enrich each entity’s representation, and then the personalized network for the certain entity is generated to transfer the representation-based knowledge. Furthermore, Chen et al. [126] adopt a representation-based distillation framework that can improve the traditional interaction-based recommender by taking advantage of a review-based recommender. They point out the representation learned from different recommenders would contain both shared and private information. From this perspective, they utilize the adversarial adaption to distill shared information by matching partial representations generated by these two recommenders. Conversely, Wang et al. [127] utilize the embedding learned from the interaction-based model to teach the abridged attribute-based model. By directly minimizing the difference between representations produced by the teacher model and the propagated representation generated from the student model, the attribute-based recommender can successfully obtain better performance.

In summary, representation-based distillation transfers the preference information from the teacher to the student by minimizing the point-wise representation difference. Compared to prediction-based ones, these methods are able to supervise the learning of representations and would be suitable for transferring latent information among embedding-based
recommenders. However, the limited model capacity still restricts the student from fully capturing the knowledge encoded in the teacher’s representation. How to properly match the representations and maximally transfer the latent information remains to be further investigated.

(3) Relation-based Knowledge

Both prediction-based and representation-based knowledge distillation rely on predictions or hidden layer representations that are provided by the teacher model. While, relation-based methods focus on inner relationships of entities that can be extracted from the training data.

To explore relationship and compatibility among clothes, Song et al. [128] utilize the auxiliary attributes to construct fashion domain knowledge. Then, the obtained fashion knowledge could further enhance the performance of clothing matching recommender to produce more fashionable pairs. However, the auxiliary information are often inapplicable in most recommendation scenarios. To this end, some researchers [129] adopt the topology information as relation-based knowledge in more generalized recommender systems. More concretely, topology information represents the relationship among the entity and its neighbors in graph convolutional network. Xu et al. [123] utilize a topology-based knowledge distillation, called GraphSAIL, to transfer the information of an entity’s local connections as well as its global position to enable efficient incremental learning in recommender systems. To be more specific, they employ the dot product value between the entity node embedding and its neighborhood aggregated representation as the distillation target to preserve the local connections of the corresponding entity. As for distilling the global positional information, two probabilities are used based on pre-obtained groups, including the probability of an entity belonging to a preference group and another probability of this entity favoring a particular category of objects. Except for the mentioned GraphSAIL, a more generalized topology distillation is proposed by Kang et al. [129]. Kang et al. [129] propose a full topology distillation (FTD) which aims to distill the entire relations in the teacher space. Specifically, a full-connected topological graph can be constructed.
2.3. REVIEW ON KNOWLEDGE DISTILLATION FOR RECOMMENDER SYSTEM

and characterized as follows:

\[ a_{ij}^t = \rho(e_i^t, e_j^t), \ \forall a_{ij}^t \in A^t \]  \hspace{1cm} (2.11)

where \( A^t \) represents the adjacency matrix of the constructed topological graph; \( a_{ij}^t \) describes the weight of an edge between entity \( i \) and \( j \) indicating the corresponding similarity calculated by the function \( \rho(\cdot) \); \( e_i^t \) and \( e_j^t \) are the representations of entity \( i \) and \( j \) learned from the teacher model. After obtaining the topological structures, the student model can be trained to preserve the rational knowledge by designed topology-preserving distillation loss as follows:

\[ \text{Loss}_{KD} = \text{Dist}(A^t, A^s) = ||A^t - A^s||_F^2, \]  \hspace{1cm} (2.12)

where \( \text{Dist}(. , .) \) is Frobenius norm to calculate the distance between the topological structures; \( A^t \) and \( A^s \) are the adjacency matrix calculated in teacher and student space. Theoretically, by minimizing the proposed topology-preserving loss, the student model could distill the valuable rational knowledge and achieve better performance. However, the author point out that directly transfers the entire relational knowledge would hinder the learning of the student owing to the large model capacity gap between the teacher and the student model. Therefore, a hierarchical topology distillation (HTD) is designed to decompose the entire topology information hierarchically and enable transfer the relational knowledge effectively. Intuitively, the author aggregate strongly correlated entities into preference groups based on the learned representations in teacher model. Then, the topology information is hierarchically structured into group-level and entity-level: where group-level topology aims to distill the relational information among preference groups, and entity-level topology focus on the entities similarities among the same preference group. The hierarchical topology preserving loss [129] can be summarized as:

\[ \text{Loss}_{KD} = \gamma(||H^t - H^s||_F^2 + ||bpM \odot (A^t - A^s)||_F^2), \]  \hspace{1cm} (2.13)

where \( H^t \) and \( H^s \) are the adjacency matrix of topological graph constructed by preference group in teacher and student space respectively; \( bpM \) is the binary matrix indicating whether a pair of entity belongs to the same preference group. To better train the student model, the authors combine the proposed loss in Equation (2.12) and the proposed loss in expert distillation and obtain the final topology distillation function.
CHAPTER 2. LITERATURE REVIEW

In brief, relation-based knowledge utilizes the learned or predefined inner relationships among entities to guide the training of the student model. Although some relation-based models are proposed recently, these models still have limited explorations on distilling relation-based knowledge. More directly, how to capture and transfer the relational information from sparse recommendation data still deserves further study.

2.3.2 Teacher-student architecture

As the core carrier to form the knowledge transferring process, the teacher–student architecture \[130\] plays an important role in knowledge distillation models. In this section, the insight from the structures of the teacher model and the student model are provided.

Classical knowledge distillation adopt teacher-student architecture to extract knowledge from a complicated and deep (teacher) model to guide the training of a compact and shallow (student) model. Following this, Chen et al. \[126\] utilize a deep neural network-based processor to distill higher-quality review information into a factorization machine-based prediction model. Wang et al. \[127\] employs a GCNs-based model to transfer the ranking information and guide the training of a simple binary collaborative filtering model. Zhu et al. \[110\] ensemble several state-of-the-art deep models to teach a vanilla DNN network and yields a significant accuracy improvements in CTR predictions. All these above mentioned methods implement different architecture for complex teacher and small student. While, several researchers directly simplify the teacher model by implementing fewer layers and smaller dimensional size to construct the student. To be more specific, these approaches would first increase the number of parameters in teacher model until its performance is no longer increased. Then, student models can be constructed by limiting the hidden layer dimensional. Considering the fact that in some state-of-the-art embedding-based recommenders, the hidden layer corresponds to the personalized embedding for individual entity. Hence, some researchers \[129\] construct the compact student model by simply using smaller dimension for entity embedding, without any changes to the original teacher model structure. Moreover, except for typical complex teacher and
smaller student pattern, employing the same architecture for both teacher and student or implementing a condensed teacher to teach the elaborate student are also explored in several recommendation scenarios. More explicitly, Xu et al. [123] utilize knowledge distillation among identical teacher and student architecture to deal with catastrophic forgetting problems in incremental training of industrial level recommendation systems. Similarly, Wang et al. [109] proposes an incremental method to transfer the historical preferences knowledge learn in the deep teacher model to the student model with the same architecture. Moreover, Song et al. [128] adopt an straightforward algorithm to extract valuable domain information and then use the extracted knowledge to guide the training of a intricate neural-based model.

In summary, most existing works employ the typical pattern which distills knowledge from the sophisticated teacher to teaching the compact student. Although some approaches that adopt identical models or utilize smaller teachers enrich the form of constructing teacher-student architectures. The identical teacher and student are mainly implemented in incremental learning cases. Meanwhile, the smaller teacher is adopted to distill domain knowledge merely because of the limited expressions in auxiliary information. The exploration of miscellaneous teacher-student architectures suitable for general recommender systems still requires future studies.

2.3.3 Distillation schemes

Beside for knowledge types and teacher-student architectures, the scheme for transferring the extracted information is also quite crucial for the success of knowledge distillation in recommender systems. A simple yet effective strategy for knowledge transfer is directly matching the prediction-based knowledge, representation-based knowledge, or relation-based knowledge between the teacher model and the student model. As already presented in Chapter 2 Section 2.1.3, the commonly adopted discrepancy modeling methods include cross-entropy, mean squared error, KL-divergence and etc. Except for the direct matching strategy, many different methods are also designed to enhance knowledge transfer in more
complex settings. A review of these methods is provided in this section.

**Adversarial Distillation** Adversarial mechanism [131] is a widely applied training technology designed for feature adaptations. Chen et al. [126] employ the generative adversarial networks (GAN) to approximate the embedding learned in user-item recommendation model to the embedding generated in review-based recommender. More specifically, the author utilize the min-max training to guide the user-item prediction network (generator $G$) to generate the embedding $e_s$ that cannot be distinguished from the embedding $e_t$ in the review-based recommender by a discriminator $D$. Meanwhile, the author integrate the GAN loss with a traditional loss to enhance the knowledge transfer process.

**Multi-teacher Distillation** As different architectures can extract different information, Zhu et al. [110] propose a multi-teacher distillation strategy which utilizes the knowledge learned in diverse teacher models to guide the student to generate more accurate CTR predictions. Concretely, an adaptive ensemble distillation framework is proposed to dynamically aggregate different teacher’s contributions. The distillation loss can be formulated as:

$$
Loss_{KD} = Loss_{KD}(S, \sum_{i=1}^{K} \alpha_i \ast T_i),
$$

(2.14)

where $\alpha_i$ is the weight to represent the contribution of $i$-th teacher $T_i$, which can be learned through a gating network with softmax function.

**Bidirectional Distillation** Kweon et al. [114] point out that the state-of-the-art methods simply assume the teacher is always superior to the student and hence employ the unidirectional distillation that transfer the knowledge merely from the teacher to the student. In fact, small student recommenders are still able to outperform the complicated teacher with respect to a notable potion of non-interacted items. To this end, a bidirectional distillation scheme is proposed to enable simultaneous knowledge transfer from teacher to student and from student to teacher. Meanwhile, considering the large model capacity gap among two models, the ranking-based sampling strategy are differently tailored according to each distillation direction. When it comes to distill from the teacher to the student, many rank-discrepant items could be considered. Conversely, only a few selected
rank-discrepant items are utilized to guide the teacher to mimic the student’s predictions. With the designed bidirectional distillation scheme, both teacher and student model could achieve better performance.

Generally, the above mention schemes enrich the strategies of knowledge transfer in recommendation scenarios. While, compared to the traditional knowledge distillation cases, other types of distillation schemes, such as graph-based distillation [132], attention-based distillation [133] and etc., can be still customized and applied as potential distillation algorithms for knowledge distillation on recommender systems.
2.4 Summary

In this chapter, the foundation of general recommender systems is first provided, together with reviews on recent neural-based recommendation approaches and KD-based recommendation models. Although the existing methods demonstrate outstanding performances, three unsolved problems or challenges still could largely influence the performances of current recommendation models, requiring more research attention.

The three worth-noticing problems or challenges summarized in this chapter are presented below:

- Most existing neural-based recommenders neglect the inherent user (item) hierarchies stored in user-item implicit interactions and fail to capture user (item) hierarchical structures that are essential for precisely modeling user (item) personalized preferences.

- Current recommendation-related approaches rarely focus on data modeling and model training processes that can largely determine the final inference performance, leading to insufficient utilization of available information and sub-optimal predictions of neural-based models.

- With the rising size of available data, endlessly increasing recommendation model complexity has become a popular trend. However, huge and complicated models not only slow down the online inference but also limit the models’ capability in capturing user (item) shifting interest, leading to outdated predictions.
Chapter 3

Preference Differences Capturing in Recommender Systems

This chapter presents a novel and straightforward point-wise training strategy, named Difference Embedding, for recommender systems that captures the personalized information retained in pair-wise preference differences. First, the background and motivation of the DifE are covered. Then, the detailed derivations for the DifE loss are presented together with a model-independent recommender training framework. The differences and the effectiveness of DifE are also discussed and emphasized. After that, the experiments are conducted from two perspectives, and their corresponding results are exhibited. The last section is the summary for this chapter.
CHAPTER 3. PREFERENCE DIFFERENCES CAPTURING IN RECOMMENDER SYSTEMS

3.1 Background

3.1.1 Motivation of preference differences capturing

To capture personalized preferences and generate customized recommendations, most existing research studies focus on the design of the compact and elegant recommendation architecture but neglect the detailed consideration of the loss function employed in the model training phase. For example, the classical neural-based NeuMF model implements a sophisticated neural-based architecture for user-item relationship modeling but adopts simple point-wise ECE loss for model training. Moreover, the state-of-the-art LightGCN recommender utilizes complex GCN operations to extract personalized information but employs straightforward pair-wise BPR to learn model hyper-parameters. In other words, although these models create distinguished recommendation architectures, their thoughtless implementation for loss function utilized in the model training process still limits the performance of those approaches.

To be more specific, point-wise loss functions, i.e., MSE and BCE, merely target reconstructing point-wise feedback, which is insufficient for real-world recommendation scenarios since the point-wise feedback does not indicate much about the actual recommendation rankings seen by users [14]. On the other hand, point-wise BPR loss proposed by Rendle et al. [15] focuses on the pair-wise differences between user’s interacted and non-interacted items. In other words, by training through users’ pair-wise interaction differences among a pair of interacted and non-interacted items, the BPR-based models can capture more personalized interests and generate acceptable recommendations. Indeed, the concept of capturing users’ personalized information from the user’s pair-wise interaction differences is appealing. However, the pair-wise BPR loss relies only on binary implicit feedback, leading to the possibility of recommending items that the user does not like, since binary interactions only describe the users’ historical behaviors rather than the degree of preference.

Briefly speaking, exiting loss functions are still insufficient for recommendation model
training, and better learning strategies are still needed.

3.1.2 Contribution of this chapter

As discussed above, the information stored in pair-wise differences is crucial for generating personalized recommendations; however, existing pair-wise loss function are inapplicable for rating-based recommendations. Based on this, a novel point-wise learning strategy, named Difference Embedding (DifE), is proposed to captures personalized information stored in pair-wise preference differences. Specifically, a pair-wise formula, Co-Preference Ratio, is firstly designed to capture the pair-wise preference differences in a pair of ratings. Then, a point-wise loss function is derived to capture valuable pair-wise discrepancy information by projecting the pair-wise discrepancy modeling into a new space. Then, with the application of the proposed point-wise loss function to the state-of-the-art base recommender, the base recommender can be trained to generate better predictions. Moreover, as the proposed point-wise loss function is independent of any point-wise preference modeling architecture, it would be easy to employ in different recommendation scenarios.

The contribution of this chapter can be summarized as:

- A novel model-independent training strategy is proposed which can utilize a point-wise loss function to capture personalized information retained in pair-wise preference differences.
- The proposed loss function is integrated with several existing recommenders and establish four optimized recommendation models, named MF-DifE, NeuMF-DifE, LightGCN-DifE and SGL-DifE, which can model personalized preferences and produce better predictions.
- Comparative experiments are conducted on three real-world datasets. Extensive results show that these optimized models achieve significant improvements compared to their corresponding baselines and also outperform a variety of recent recommen-
CHAPTER 3. PREFERENCE DIFFERENCES CAPTURING IN RECOMMENDER SYSTEMS

...dation methods, revealing the excellence and generality of the proposed point-wise loss function.

3.2 Proposed Method

This section presents the proposed point-wise loss function and demonstrates its ability to capture pair-wise information from explicit feedbacks.

3.2.1 Co-preference ratio

As discussed in above, existing point-wise loss functions focus solely on reconstruction accuracy of explicit feedback and ignore the critical pair-wise ranking information, while current pair-wise loss function is only designed for implicit feedback. To capture the crucial pair-wise information stored in explicit feedback, a formula called Co-preference ratio (CPR) is proposed to describe the pair-wise ranking difference between two items. The CPR of item $i$ and item $j$ for user $u$, is defined as follows:

$$CPR_{uij} = \frac{P_{ui}}{P_{uj}},$$

where $P_{ui}$ and $P_{uj}$ represent user $u$’s preferences for item $i$ and item $j$, respectively.

Intuitively, $P_{ui}$ can be defined as $r_{ui}$ since the magnitude of $r_{ui}$ reveals how much user $u$ likes item $i$. However, real-world recommender data contain only a limited number of discrete values in the rating system, which prevents the assigned rating $r_{ui}$ from accurately describing the user’s true interest. Moreover, the pair-wise differences might be easily biased by the point-wise rating pattern. For this reason, a neighbor-based preference adjustment method is proposed to obtain a accurate and robust preference $P_{ui}$. The designed preference adjustment methods are demonstrated below:

$$p_{ui} = r_{ui} + \frac{1}{\|Sim_{u} \cup Sim_{i}\|} \cdot (\sum_{n \in Sim_{u}} r_{ni} + \sum_{m \in Sim_{i}} r_{um})$$

$$P_{ui} = \frac{p_{ui} \cdot p_{ui}}{p_{ui} \cdot p_{i}} = \frac{p_{ui}^2}{p_{ui} \cdot p_{i}},$$
where \( r_{ui} \) denotes the user \( u \)'s rating on item \( i \), and \( Sim_u \) and \( Sim_i \) denote the set of similar neighbors of user \( u \) and item \( i \) respectively. With the rating information obtained from similar neighbors, the modified preference \( p_{ui} \) of user \( u \) given item \( i \) can be calculated by Equation (3.2). Then, by combing the user-side average rating \( p_\bar{u} \) and item-side average rating \( p_\bar{i} \), the final adjusted preference \( P_{ui} \) can be finally obtained through Equation (3.3). Compared with the original rating \( r_{ui} \), the final adjusted \( P_{ui} \) containing the information of similar neighbors would be more suitable for the task of pair-wise preference modeling. Detailed analysis will be provided in Chapter 3 Section 3.2.4.

Briefly, the proposed \( CPR_{uij} \) uses the ratio between the adjusted preferences \( P_{ui} \) and \( P_{uj} \) to describe the difference of user \( u \)'s preferences between item \( i \) and item \( j \), where \( CPR_{uij} \gg 1 \) indicates that user \( u \) prefers item \( i \) over item \( j \), and correspondingly, \( CPR_{uij} \ll 1 \) represents the opposite interests.

### 3.2.2 Derivation of the new loss function

It is clear that the defined pair-wise CPR is able to capture a user’s preference difference among a pair of items. In other words, embeddings that can generate CPR value will contain useful personalized ranking information. From this perspective, the following formula is summarized:

\[
\Phi(f(e_u, e_i) - f(e_u, e_j)) = CPR_{uij} = \frac{P_{ui}}{P_{uj}},
\]

(3.4)

where \( e_u, e_i \) and \( e_j \) are embeddings of user \( u \), item \( i \) and item \( j \) respectively; the function \( f(\cdot) \) is a relationship modeling method that projects corresponding embeddings into a scalar number presenting the degree of preference. Most importantly, a homomorphism \( \Phi() \) is defined to project the preference difference to the proposed CPR value. Since a homomorphism is a structure-preserving mapping between two algebraic structures of the same type, the following homomorphism condition is implemented:

\[
\begin{align*}
\Phi(a + b) &= \Phi(a) * \Phi(b) \\
\Phi(a - b) &= \Phi(a)/\Phi(b)
\end{align*}
\]

(3.5)
According to the homomorphism condition described in Equation (3.5), the left side of Equation (3.4) is written as:

$$\Phi(f(e_u, e_i) - f(e_u, e_j)) = \Phi(f(e_u, e_i)) - \Phi(f(e_u, e_j))$$

(3.6)

With Equation (3.4) and Equation (3.6), the following equation can be obtained.

$$\frac{\Phi(f(e_u, e_i))}{\Phi(f(e_u, e_j))} = \frac{P_{u_i}}{P_{u_j}}$$

(3.7)

The straightforward solution for Equation (3.7) can be specified as:

$$\Phi(f(e_u, e_i)) = P_{u_i} = \frac{p_{u_i}^2}{p_u * p_i}$$

(3.8)

Meanwhile, $\Phi(\cdot)$ is defined as the natural exponential function to fulfill the homomorphism condition. By taking the natural logarithm for both sides in Equation (3.8), the following relations can be obtained:

$$f(e_u, e_j) = 2 * \ln p_{u_i} - \ln p_u - \ln p_i$$

(3.9)

Since the $\ln p_{u\_average}$ and $\ln p_{i\_average}$ in Equation (3.9) are two constant scalar numbers, they can be regarded as the deviation of user $u$ and the deviation of item $i$ respectively. Then, the final equation becomes,

$$f(e_u, e_i) + b_u + b_i = 2 * \ln p_{u_i}$$

(3.10)

Based on the above analysis, it is clear that embeddings that can satisfy the Equation (3.10) can successfully represent CPR value. In other words, by designing a point-wise loss function as follows:

$$\text{Loss} = \sum_{u,i} (f(e_u, e_i) + b_u + b_i - 2 * \ln p_{u_i})^2$$

(3.11)

, the pair-wise ranking information can be successfully encoded into user (item) embeddings.

### 3.2.3 Application of loss function

Based on the above analysis, the proposed loss function can capture the pair-wise differences from the point-wise loss function. The next step is to combine the proposed loss
3.2. PROPOSED METHOD

Figure 3.1: An illustration of the proposed DifE-framework structure. The output of the framework is computed by combining the learned user (item) biases and the point-wise preference generated from the point-wise recommendation model. The whole framework is independent of recommenders and can be applied to any point-wise recommendation models, such as MF [1], NeuMF [2] and LightGCN [3]. The detailed architecture of these mentioned models can be found in the referenced paper.

function with a state-of-the-art recommendation model to achieve better recommendation performance. To this end, a model-independent difference embedding (DifE) framework is proposed based on Equation (3.11), which is described as follows.

As shown in Figure 3.1, \( f(e_u, e_i) \) refers to the point-wise preference generated by a base recommender. The model output can then be computed by summing the generated preference with two trainable scalar deviations. It is worth pointing out that the DifE framework can be applied to any point-wise recommendation model since the whole DifE framework is independent from the choice of the base recommender. More specifically, the DifE framework can be considered as a post-processor that projects the preference modeled from the base recommender into a novel space. Then, by minimizing the point-wise loss in the projection space, the base recommender can be trained to capture the essential pair-wise ranking information and achieve better recommendation performance.

The DifE framework is only utilized during the model training phase. Once the recommender is well trained, the point-wise preference \( f(e_u, e_i) \) will be directly used for gener-
CHAPTER 3. PREFERENCE DIFFERENCES CAPTURING IN RECOMMENDER SYSTEMS

ating the $Top@K$ prediction list. This is because the output of the DifE framework only represents the partial ratio in the projected space, while the crucial pair-wise information for precise preference modeling is already captured in the recommender after training.

3.2.4 Theoretical analysis of difference embedding

In this section, a comprehensive analysis for the proposed loss function is presented in terms of pair-wise difference modeling, preference adjustment strategies and computational complexity.

Pair-wise Difference Modeling Compared to the traditional subtraction-based difference (SUBD) calculation (i.e. $SUBD_{ui} = P_{ui} - P_{uj}$), the proposed CPR modeling can provide more benefits by emphasizing the true negative samples. Taken the pair-wise discrepancy modeling in a real-world recommendation scenario as an example. The potential values of $CPR_{uij}$ and $SUBD_{uij}$ for a given preference $P_{ui} = 5$ and an unknown $P_{uj}$ ranging from 1 to 5 are shown in Figure 3.2 (a) and (b), respectively. As demonstrated in Figure 3.2, the monotonic decreasing relationship in each sub plot reveals that both CPR and SUBD strategies are effective in capturing the pair-wise preference differences between $P_{ui}$ and $P_{uj}$. At the same time, the proposed ratio-based approach enlarges the variation in preference discrepancies for the low-rated component compared to the subtraction-based calculation, thus emphasizing the importance of the low-rated items. Since these low-rated items reveal explicit user dislikes, these disliked items can be interpreted as highlighting the past mistakes and avoiding similar errors in future predictions, which will undoubtedly benefit the recommendation process. The empirically evaluation for the superiority of the proposed ratio-based CPR over the typical subtraction-based SUBD is presented in Chapter 3 Section 3.4.1.

Preference Adjustment Strategy The ratings of similar neighbors are first used to lift the original $r_{ui}$ to a more informative preference $p_{ui}$ (Equation (3.2)). Then, a ratio-based normalization from both user- and item-side is implemented in $P_{ui}$ to reduce the impact of individual deviations in pair-wise preference discrepancy capturing (Equation (3.3)).
3.2. PROPOSED METHOD

Figure 3.2: Comparison for potential values of \( CPR_{uij} \) and \( SUBD_{uij} \) with a given preference \( P_{ui} = 5 \) and an unknown \( P_{uj} \) ranging from 1 to 5.

Indeed, following the derivation procedure in Chapter 3 Section 3.2.2, the ratio-based normalization can be translated into the commonly adopted learn-able biases \( b_u \) and \( b_i \) in derived loss function (Equation (3.11)). Meanwhile, to analyze the neighbor-based preference modification algorithm, the following transformation procedure is employed:

\[
p_{ui} = r_{ui} + \frac{1}{\|Sim_u \cup Sim_i\|} \left( \sum_{n \in Sim_u} r_{ni} + \sum_{m \in Sim_i} r_{um} \right) = r_{ui} + r'_{ui} = r_{ui} \times (1 + \frac{r'_{ui}}{r_{ui}}) = \alpha \times r_{ui}
\]

Clearly, the modified \( p_{ui} \) can be considered as a weighted preference calculated from the original rating \( r_{ui} \), where the weight \( \alpha \) is determined by the average rating \( r'_{ui} \) obtained from similar neighbors. In short, the proposed preference adjustment methods enrich the preference representations and reduce individual bias, which enables more informative and robust preference discrepancy modeling. The experiments for confirming the validity of the proposed adjustment strategy are provided in Chapter 3 Section 3.4.3.

**Model Complexity** Given the base point-wise recommender \( f(e_u, e_i) \) and its complexity \( O(N) \), the model complexity for the corresponding DifE-optimized recommender remains \( O(N) \) since the DifE framework will not change the architecture of the base model. Meanwhile, the computation comes from the bias parameters learned from one-hot vectors would not change the magnitude of computational cost. Moreover, the computational cost from
CHAPTER 3. PREFERENCE DIFFERENCES CAPTURING IN RECOMMENDER SYSTEMS

Table 3.1: Statistics of experimental datasets.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Users</th>
<th>Items</th>
<th>Ratings</th>
<th>Data Density</th>
<th>Rating Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>BXBooks</td>
<td>7025</td>
<td>9432</td>
<td>118668</td>
<td>0.00179</td>
<td>1 ～ 10</td>
</tr>
<tr>
<td>AmazonMIs</td>
<td>27530</td>
<td>10620</td>
<td>219806</td>
<td>0.00075</td>
<td>1 ～ 5</td>
</tr>
<tr>
<td>Epinions</td>
<td>20397</td>
<td>21901</td>
<td>446892</td>
<td>0.00100</td>
<td>1 ～ 5</td>
</tr>
</tbody>
</table>

The preference adjustment process will not add any extra complexity of the algorithm since the adjusted preferences through the neighbors’ ratings can be pre-computed before training.

3.3 Experiments Settings

This section presents the experiment settings from the perspective of data-set, evaluation metrics, and implementation details.

3.3.1 Dataset description

The experiments are conducted on three public datasets, called BXBooks, Amazon musical instruments (AmazonMIs), and Epinions. These datasets are benchmarking datasets that have been extensively chosen in recommendation scenarios. Meanwhile, as the Amazon Datasets follow the 5-core setting\(^1\), i.e., retaining users and items with at least 5 interactions, the Epinions and BXBooks Datasets are prepossessed in the same manner to ensure same data quality. The detailed dataset statistics are shown in Table 1.

In general, 80% of interactions from each dataset are randomly selected to constitute the training set, while the remainder forms the test set. From the each training set, 10% of interactions are randomly selected as a validation set to tune hyperparameters.

\(^1\)https://jmcauley.ucsd.edu/data/amazon/
3.3. EXPERIMENTS SETTINGS

3.3.2 Evaluation metrics

To evaluate the recommendation accuracy, two widely-applied protocols, i.e., \textit{Recall@K} and \textit{NDCG@K} are adopted. More specifically, \textit{Recall@K} focus on the average hit ratio of \textit{Top@K} recommendation list, while \textit{NDCG@K} emphasizes the importance of higher-ranked hit items. In other words, higher \textit{Recall@K} reveals that a larger portion of user-preferred items are successfully recommended in the \textit{Top@K} list. Meanwhile, higher \textit{NDCG@K} indicates the hit items are relatively higher-ranked among the \textit{Top@K} list. The detailed calculation for \textit{Recall@K} is presented in Equation (3.13).

\[
\text{Recall@K} = \frac{|L_{\text{tar}} \cap L_{\text{rec}@K}|}{|L_{\text{tar}}|} \quad (3.13)
\]

where \(L_{\text{tar}}\) refers to the set of target items, \(L_{\text{rec}@K}\) denotes the \textit{Top@K} recommendation list, and \(|·|\) indicates the length of the set. The detailed calculation for \textit{NDCG@K} is demonstrated below.

\[
\text{DCG@K} = \sum_{i=1}^{|L_{\text{tar}} \cap L_{\text{rec}@K}|} \frac{1}{\log_2(i + 1)} \quad (3.14)
\]

\[
\text{IDCG@K} = \sum_{i=1}^{|L_{\text{tar}}|} \frac{1}{\log_2(i + 1)} \quad (3.15)
\]

\[
\text{NDCG@K} = \frac{\text{DCG@K}}{\text{IDCG@K}} \quad (3.16)
\]

In addition to accuracy evaluation, recommendation satisfaction is also considered to obtain more comprehensive comparison results. Specifically, the mean reciprocal rank (\textit{MRR@K}) \cite{135} that reveals the ranking position of the first target items is first implemented. The defined calculation for \textit{MRR@K} is demonstrated in Equation (3.15).

\[
\text{MRR@K} = \frac{1}{h_u} \quad (3.17)
\]

where \(h_u\) refers to the ranking position of the first hit item in user \(u\)’s \textit{Top@K} recommendation list. Customers often prefer higher rankings for their interesting items, so a greater value of \textit{MRR@K} indicates more satisfactory recommendations. Meanwhile, considering the \textit{MRR@K} merely focuses on the first hit item but neglects the rest items in the \textit{Top@K} list.
list, another evaluation metric, called Novelty@$K$ \cite{136,137}, is also employed. More concretely, Novelty@$K$ reveals the average uniqueness of the hit items in the Top@$K$ list. Its computing equation is defined as follows:

$$Novelty@K = \frac{1}{|L_{tar} \cap L_{rec}@K|} \sum_{i \in L_{tar} \cap L_{rec}@K} -log_2 \frac{||N_i||}{||N^u||}$$ (3.18)

where $N_i$ is the set of users that rated item $i$, and $N^u$ is the set of all users. The higher Novelty@$K$ indicates more unpopular items are successfully recommended to user $u$. Since customers often feel bored with popular items, a greater value of Novelty@$K$ would definitely lead to more satisfied recommendations. Last but not least, the above mentioned four metrics are designed for the evaluation on a single user $u$. Their corresponding average values on all users would be treated as the final experimental results with respect to four different evaluation criteria.

### 3.3.3 DifE models

To evaluate the proposed loss function, four state-of-the-art base recommenders are selected and optimized with the model-independent DifE framework. The detailed implementations for four DifE-optimized models are presented below.

**1. MF-DifE Model**

As a classic recommendation method, MF directly projects users and items into a low dimensional space to model complex user-item relationships. With its simple and straightforward strategies, MF has been applied to many real-world recommendation scenarios. To this end, the traditional MF recommendation model is selected as the first base recommender that would be optimized by DifE framework. The implemented architecture of MF-DifE model is depicted in Figure 3.3.

As shown in Figure 3.3, MF-DifE project user’s (item’s) one-hot encoding into a low dimensional representation (i.e., user (item) embedding) to represent certain user (item). However, instead of simply applying dot product on embeddings to reconstruct ratings
(or interactions), a more robust and reasonable modeling learning strategy inspired by
the proposed loss function is adopted in the optimized MF-DifE model. Specifically, the
additional user and item biases are learned to handle the variations caused by different
rating strategies. And the final output of MF-DifE is calculated through summing up the
calculated preferences as well as two learned biases.

To train the MF-DifE model properly, the regularization for embedding is considered to
avoid over-fitting. The final loss function for the MF-DifE model is presented below:

\[
\text{Loss}_{MF-DifE} = \sum_{u,i}(e_u \cdot e_i^T + b_u + b_i - 2 \ast \ln p_{ui})^2 + \lambda \|E\| \tag{3.19}
\]

, where \(\lambda\) controls the L2 regularization strength and \(E\) is the initial embedding matrix.

When it comes to prediction, the learned embedding for user (item) is directly utilized for
generating prediction list rather than the output of the MF-DifE model. The reason can
be describe as the output of the MF-DifE model only represents the partial ratio in the
projected space, while the crucial pair-wise information for precise preference modeling
is already captured in learned embedding after training. In other words, the predicted
preference of user \(u\) on target item \(i\) is defined as follow:

\[
y_{ui} = e_u \cdot e_i^T \tag{3.20}
\]
The final $Top@K$ recommendation list for user $u$ can then be generated with the ranking based on the predicted value.

(2) NeuMF-DifE Model

In the past few years, deep neural networks have yielded immense success on recommendation tasks. As the state-of-the-art neural-based model, NeuMF replaces the inner product with a neural architecture and is able to learn more complex user-item relationships. Based on the success of NeuMF, the state-of-the-art NeuMF recommender are selected as the second base model that would be further optimized. The whole model architecture of the NeuMF-DifE model is presented in Figure 3.4.

Same as the architecture of NeuMF, NeuMF-DifE utilizes a complicated neural-based architecture to model user-item relationships. To be more specific, MLPs and Generalized
Matrix Factorization are utilized to discover latent and linear correlations among user and item, respectively. Then, the fusion of these two relationships enables more complex user-item relationship modeling. Most importantly, inspired by the proposed loss function, a more robust and reasonable calculation strategy is applied in NeuMF-DifE. By summing up the user-item preference calculated through the neural architecture with the learnable user and item biases, the final output of NeuMF-DifE would finally be obtained,

With regard to training process of NeuMF-DifE, the following loss function is employed:

\[
    \text{Loss}_{\text{NeuMF-DifE}} = \sum_{u,i} \left( \hat{y}_{ui} + b_u + b_i - 2 \ast \ln p_{ui} \right)^2 + \lambda \|E\| \tag{3.21}
\]

where \(\lambda\) controls the L2 regularization strength and \(E\) is the embeddings generated by the GMF layer and MLP layer. As for recommendation prediction, the output of the neural architecture, i.e., \(\hat{y}_{ui}\), rather than the output of NeuMF-DifE, would be utilized for the final preference prediction. The reason is the same as the discussion presented in the above MF-DifE model discussions. With the predicted \(\hat{y}_{ui}\), the final \(Top@K\) recommendation list can easily be produced.

(3) LightGCN-DifE and SGL-DifE Models

GCNs have attracted much research attention and been widely applied to recommender systems due to their superiority on node embedding learning tasks. As the state-of-the-art graph-based recommendation model, LightGCN [3] simplifies the design of graph convolution operation and achieves competitive performance. Hence, the proposed DifE are also combined with the state-of-the-art LightGCN recommender. Meanwhile, a more recent GCNs-based achievement, i.e., SGL [71], is also considered as the base model. Since LightGCN and SGL adopt the same graph architecture as well as the same graph convolutional operations, the architecture of the LightGCN-DifE (SGL-DifE) are presented in Figure 3.5.

Identical to LightGCN, LightGCN-DifE integrates user-item connections into user-item embedding learning process to explicitly model crucial collaborative signal and finally generate better embedding. Specifically, the graph convolutional operation only includes
Figure 3.5: Overview of LightGCN-DifE (SGL-DifE) structure

neighborhood aggregation to calculate the embedding of k+1 th layer, which is defined as follows:

\[
e^{(k+1)} = \sum_{i \in N_u} \frac{1}{\sqrt{||N_u||} \sqrt{||N_i||}} e_i^{(k)}
\]

(3.22)

The symmetric normalization term \( \frac{1}{\sqrt{||N_i||} \sqrt{||N_u||}} \) follows the design of standard GCNs [67] to avoid the scale of embeddings increasing with graph convolution operations. Meanwhile, as embeddings at different layers capture different information, the embeddings obtained at each layer is combined to form the final comprehensive user (item) embeddings, which
can be computed as follows:

\[
e_u = \sigma \left( \frac{1}{n} \sum_{k=0}^{n} e_u^{(k)} \right) \quad ; \quad e_i = \sigma \left( \frac{1}{n} \sum_{k=0}^{n} e_i^{(k)} \right)
\]

(3.23)

where \( \sigma(\cdot) \) is the activation function to enable modeling no-linear relationships. More notably, to encode the crucial pair-wise ranking information and generate more robust and informative embeddings, LightGCN-DifE (SGL-DifE) utilizes the same calculation strategy inspired by the proposed loss function instead of reconstructing binary interaction directly from embeddings.

To train the graph model more precisely, the following loss function is adopted for LightGCN-DifE:

\[
Loss_{LightGCN-DifE} = \sum_{u,i} \left( e_u \cdot e_i^T + b_u + b_i - 2 \ln p_{ui} \right)^2 + \lambda \| E \| \quad (3.24)
\]

, where \( \lambda \) controls the L2 regularization strength and \( E \) is the only trainable parameters, i.e., 0-th layer embeddings. As for SGL-DifE, the self-supervised learning paradigm presented in SGL [71] is integrated with the proposed one, which can be finally trained with the following function:

\[
Loss_{SGL-DifE} = \sum_{u,i} \left( e_u \cdot e_i^T + b_u + b_i - 2 \ln p_{ui} \right)^2 + \lambda_1 Loss_{ssl} + \lambda_2 \| E \| \quad (3.25)
\]

, where \( Loss_{ssl} \) is the self-supervised learning (SSL) loss which has been detailed introduced in [71]: \( \lambda_1 \) and \( \lambda_2 \) are hyperparameters to control the strengths of SSL and regularization, respectively. To compute the predicted preference \( \hat{y}_{ui} \), the inner product are employed on final embedding of user \( u \) and item \( i \), which is presented as follow:

\[
\hat{y}_{ui} = e_u \cdot e_i^T \quad (3.26)
\]

The reason of utilizing the calculated \( \hat{y}_{ui} \) rather than the output of LightGCN-DifE (SGL-DifE) model is identical to MF-DifE and NeuMF-DifE models. Based on the calculated \( \hat{y}_{ui} \), the final Top@K recommendation list for user \( u \) can then be generated.
3.3.4 Selected baseline models

To thoroughly evaluate the proposed loss function and demonstrate its superiority, the compared baseline models are selected from two perspectives.

(1) For loss function-based analysis

Considering the fact that DifE-optimized models retain the same architecture with their corresponding base model, the four base recommender would firstly be adopted as the essential baseline models.

- **MF**: The state-of-the-art matrix factorization model that projects one-hot vectors into low-dimensional embeddings to model user-item relationships.

- **NeuMF**[^2] The state-of-the-art neural-based architecture that combines MF and MLPs to learn the user-item interaction.

- **LightGCN[^3]** The state-of-the-art graph-based model that utilizes simplified GCNs to extract the information of multi-hop neighbors.

- **SGL[^4]** The recent graph-based collaborative filtering model that adopts self-supervised graph training strategy.

Moreover, since the DifE-optimized models utilize explicit feedback whereas the base models only require implicit feedback, an additional controlled group of models, named SUBD-optimized models, are also conducted. SUBD-optimized models (i.e., **MF-SUBD**, **NeuMF-SUBD**, **LightGCN-SUBD** and **SGL-SUBD**) employ the same model architecture as their respective DifE-optimized models. The only difference between these models is their loss function, where DifE-optimized models adopt the loss function presented

[^2]: https://github.com/hexiangnan/neural_collaborative_filtering
[^3]: https://github.com/gusye1234/LightGCN-PyTorch
[^4]: https://github.com/wujcan/SGL
in Section 3.3.3 while SUBD-optimized models train the recommender by minimizing the Mean-Squared Error on rating $r_{ui}$.

(2) For model-based analysis

Except for the comparisons among the DifE optimized models and their corresponding baseline, the performances of the optimized-models compared to the most recent state-of-the-art methods are also crucial for confirming the effectiveness. To evaluate the overall performance of DifE optimized-models (i.e., NeuMF-DifE, LightGCN-DifE and SGL-DifE), the following newly proposed methods are selected:

- **DMF** [40]: This is a modern deep matrix factorization method that utilizes explicit ratings as input and adopts the normalized cross-entropy loss as a loss function to generate embeddings.
- **NeuMF** [2]: This is a state-of-the-art neural-based collaborative filter method which combines two parts called GMF and MLP to generate recommendation results.
- **DeepCF** [5] [62]: This is another state-of-the-art neural-based collaborative filtering method which utilizes more complex architecture to generate final recommendations from implicit data.
- **NGCF** [67]: This is a state-of-the-art GCNs-based model designed for collaborative filtering which integrates the user-item interaction into the embedding process through the graph convolution operation to generate user/item embeddings.
- **NIA-GCN** [69]: This is a state-of-the-art GCNs-based model which explicitly exploits the user-user and item-item relationships through pair-wise neighborhood aggregator.
- **LightGCN** [3]: This is a state-of-the-art proposed GCNs-based model that simplifies the design of graph convolution operation and only includes neighbor aggregation

---

5https://github.com/familyld/DeepCF
6https://github.com/huangtinglin/NGCF-PyTorch
in convolution operations.

- **SGL [71]**: This is a newly proposed Graph Training Strategy which implements on the LightGCN model and achieves competitive performance.

- **UltraGCN [68]**: This is a newly proposed ultra-simplified GCNs-based model which skips infinite layers of message passing and yields efficient and effective recommendations.

### 3.3.5 Negative sampling strategy

To handle the unbalanced distribution of low-rated data, the common negative sampling strategy are also adopted during the training process of all these proposed models [40]. In more details, $n$ non-interacted items are randomly sampled as negative samples for each rated positive item during each training iteration. The rating of each selected negative sample is set to 1, and therefore the target value for a sampled negative instance during the training epoch under the DifE-framework should be 0 (calculated by $2 \times \ln 1$).

### 3.3.6 Other implementation details

The proposed models are implemented based on Pytorch 1.1+ with the embedding size fixed to 64 and the embedding parameters initialized with the Xavier method [138]. The default learning rate is 0.001, and the default mini-batch size is 1024. For the implementation of the baseline models, the default training strategies and hyperparameters settings are followed by the corresponding referenced paper.

### 3.4 Experiments Results

In this section, the following research questions are focused:
3.4. EXPERIMENTS RESULTS

- **Q1**: Is the proposed loss function beneficial to the Top@K recommendation tasks? What are the benefits of integrating the proposed loss function with base state-of-the-art recommenders?

- **Q2**: How do the DifE framework optimized models perform compared with newly proposed state-of-the-art models? Does the proposed method achieve higher recommendation accuracy and produce better predictions?

- **Q3**: How do different settings influence the effectiveness of the proposed loss functions?

3.4.1 Loss function analysis (RQ1)

To evaluate the effectiveness of the proposed function, the experimental results from different perspective are compared among base, SUBD-optimized and DifE-optimized models.

(1) Prediction Accuracy Comparison

The Recall@20 and NDCG@20 result for all base recommender, SUBD-optimized model and DifE-optimized models are presented in Table 3.2.

The observations are listed below:

- The four DifE-optimized models consistently achieve higher recall and higher NDCG than their corresponding state-of-the-art baselines, as well as their corresponding SUBD-optimized models. The significant improvements between DifE-optimized models and their corresponding baselines reveal the effectiveness of the proposed loss function.

- In comparison to SUBD-optimized models, the outstanding performance of DifE-optimized models demonstrates that the ratio-based discrepancy modeling (i.e., CPR calculation) can offer more benefits than the subtraction-based discrepancy modeling (SUBD), which in accordance with the analysis provided in Section 3.2.4. Meanwhile, considering that SUBD-optimized models also incorporate explicit feedback,
### Chapter 3. Preference Differences Capturing in Recommender Systems

Table 3.2: Recall@20 and NDCG@20 results for base, SUBD-optimized and DifE-optimized models.

<table>
<thead>
<tr>
<th>Compared Methods</th>
<th>BXBooks</th>
<th></th>
<th></th>
<th>AmazonMIs</th>
<th></th>
<th></th>
<th>Epinions</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Recall</td>
<td>NDCG</td>
<td>Recall</td>
<td>NDCG</td>
<td>Recall</td>
<td>NDCG</td>
<td>Recall</td>
<td>NDCG</td>
<td>Recall</td>
</tr>
<tr>
<td>MF-BPR</td>
<td>0.0372</td>
<td>0.0208</td>
<td>0.1028</td>
<td>0.0593</td>
<td>0.0432</td>
<td>0.0284</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MF-BCE</td>
<td>0.0403</td>
<td>0.0246</td>
<td>0.1334</td>
<td>0.1063</td>
<td>0.0467</td>
<td>0.0293</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MF-SUBD</td>
<td>0.0861</td>
<td>0.0520</td>
<td>0.1670</td>
<td>0.1158</td>
<td>0.0684</td>
<td>0.0431</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MF-DifE</td>
<td>0.0907</td>
<td>0.0546</td>
<td>0.1779</td>
<td>0.1221</td>
<td>0.0803</td>
<td>0.0499</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Impro-O</td>
<td>+125.06%</td>
<td>+121.95%</td>
<td>+33.36%</td>
<td>+14.86%</td>
<td>+71.95%</td>
<td>+70.31%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Impro-SUBD</td>
<td>+5.34%</td>
<td>+5.00%</td>
<td>+6.53%</td>
<td>+5.44%</td>
<td>+17.40%</td>
<td>+15.78%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NeuMF</td>
<td>0.0627</td>
<td>0.0392</td>
<td>0.1494</td>
<td>0.1058</td>
<td>0.0785</td>
<td>0.0505</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NeuMF-SUBD</td>
<td>0.0695</td>
<td>0.0437</td>
<td>0.1498</td>
<td>0.1052</td>
<td>0.0762</td>
<td>0.0491</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NeuMF-DifE</td>
<td>0.0778</td>
<td>0.0512</td>
<td>0.1737</td>
<td>0.1285</td>
<td>0.0844</td>
<td>0.0550</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Impro-O</td>
<td>+24.08%</td>
<td>+30.61%</td>
<td>+16.27%</td>
<td>+21.56%</td>
<td>+7.52%</td>
<td>+8.92%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Impro-SUBD</td>
<td>+11.94%</td>
<td>+17.16%</td>
<td>+15.95%</td>
<td>+22.15%</td>
<td>+10.76%</td>
<td>+12.02%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LightGCN</td>
<td>0.0988</td>
<td>0.0643</td>
<td>0.1900</td>
<td>0.1363</td>
<td>0.1039</td>
<td>0.0686</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LightGCN-SUBD</td>
<td>0.1083</td>
<td>0.0665</td>
<td>0.1909</td>
<td>0.1371</td>
<td>0.1053</td>
<td>0.0685</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LightGCN-DifE</td>
<td>0.1144</td>
<td>0.0722</td>
<td>0.2024</td>
<td>0.1451</td>
<td>0.1111</td>
<td>0.0732</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Impro-O</td>
<td>+15.79%</td>
<td>+12.29%</td>
<td>+6.53%</td>
<td>+6.46%</td>
<td>+6.93%</td>
<td>+6.71%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Impro-SUBD</td>
<td>+5.63%</td>
<td>+8.57%</td>
<td>+6.02%</td>
<td>+5.84%</td>
<td>+4.84%</td>
<td>+6.86%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SGL</td>
<td>0.1002</td>
<td>0.0630</td>
<td>0.2017</td>
<td>0.1473</td>
<td>0.1083</td>
<td>0.0718</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SGL-SUBD</td>
<td>0.1096</td>
<td>0.0704</td>
<td>0.2032</td>
<td>0.1483</td>
<td>0.1092</td>
<td>0.0720</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SGL-DifE</td>
<td>0.1147</td>
<td>0.0742</td>
<td>0.2057</td>
<td>0.1507</td>
<td>0.1116</td>
<td>0.0738</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Impro-O</td>
<td>+14.47%</td>
<td>+17.78%</td>
<td>+2.48%</td>
<td>+2.31%</td>
<td>+3.05%</td>
<td>+2.79%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Impro-SUBD</td>
<td>+4.65%</td>
<td>+5.40%</td>
<td>+1.72%</td>
<td>+1.62%</td>
<td>+2.20%</td>
<td>+2.50%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The results confirm that the improvements of DifE-optimized models are primarily attributed to the proposed loss function instead of more informative input.

- GCNs-based models always achieve better performance than classical MF and Neural-based models. Moreover, as the selected baselines can be regarded as the different state-of-the-art prototypes of CF-based recommender systems, the success of all these optimized models exhibit the great potential of the proposed loss function in handling recommendation problems.

- The better performance of MF-DifE than NeuMF-DifE on datasets BXBooks and AmazonMIs can be explained as a more complex model would easier over-fit in small datasets. Correspondingly, in a bigger and more complicated dataset, such as
3.4. EXPERIMENTS RESULTS

Epinions, NeuMF-DifE achieves a higher recall and a higher NDCG.

The comparisons on Recall@20 and NDCG@20 imply that the Top@20 recommendation lists produced by DifE-optimized models achieve higher prediction accuracy. However, in real-world recommendation scenarios, higher accuracy would not necessarily translate into more satisfied recommendations. Further experiments are therefore required to comprehensively evaluate the optimized models.

(2) Recommendation Satisfaction Analysis

Based on the analysis above, two commonly adopted MRR@K and Novelty@K are adopted to evaluate the DifE-optimized models from the perspective of recommendation satisfaction. To begin, the MRR@K comparisons which indicate the average position of the first hit item in the recommended list are discussed. The MRR results of Neural-based and GCNs-based models at four different Top@K scenarios are presented in Figure3.6. As shown in Figure3.6 DifE-optimized models consistently achieve higher MRR values at different Top@K scenarios on all three datasets, which indicates that the recommendation list generated by the DifE-optimized model can obtain a higher satisfaction rate since first target item are often ranked higher.

In addition to MRR evaluation, the Novelty@K of the hit item is also an essential criteria to evaluate recommendation satisfactions. A higher Novelty values would undoubtedly implies more satisfied recommendations since user always prefer innovative predictions. More specifically, the Novelty@K results for SGL and SGL-DifE are calculated and presented in Table 3.3.

Table 3.3: Novelty@K results of SGL and SGL-DifE models on different datasets.

<table>
<thead>
<tr>
<th>Top@K</th>
<th>BXBooks</th>
<th>AmazonMls</th>
<th>Epinions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SGL</td>
<td>SGL-DifE</td>
<td>SGL</td>
</tr>
<tr>
<td>Novelty@5</td>
<td>7.3959</td>
<td>7.4618</td>
<td>7.8890</td>
</tr>
<tr>
<td>Novelty@10</td>
<td>7.5072</td>
<td>7.5074</td>
<td>8.0814</td>
</tr>
<tr>
<td>Novelty@20</td>
<td>7.5795</td>
<td>7.6120</td>
<td>8.2687</td>
</tr>
<tr>
<td>Novelty@50</td>
<td>7.7447</td>
<td>7.7573</td>
<td>8.5081</td>
</tr>
</tbody>
</table>
CHAPTER 3. PREFERENCE DIFFERENCES CAPTURING IN RECOMMENDER SYSTEMS

(a) Comparison among NeuMF, NeuMF-SUBD, NeuMF-DifE.

(b) Comparison among LightGCN, LightGCN-SUBD, LightGCN-DifE.

(c) Comparison among SGL, SGL-SUBD, SGL-DifE.

Figure 3.6: MRR results on different Top@K recommendation scenarios.
Based on the results in Table 3.3, the SGL-DifE model consistently achieves higher average prediction novelties than the basic SGL model on the BXBooks and Epinions datasets under a variety of Top@$K$ recommendation scenarios. The significant novelty improvement achieved by the SGL-DifE model on Top@5 predictions implies that the DifE-optimized models are capable of recommending innovative items at the top-ranking positions, which would certainly increase the user satisfaction. In particular, the results in Figure 3.6 and Table 3.3 demonstrate that the DifE-optimized model can generate more satisfactory recommendations.

(3) Data Sparsity Analysis

Additionally to the above-mentioned accuracy and satisfaction evaluation, the performance of the DifE-optimized model under diverse recommendation scenarios is also quite crucial for the application of the proposed loss function. To this end, four sparsity levels of the training set are used to evaluate the performance of different models in real-world sparse recommendation situations.

To enable performance comparison among different sparsity level experiments, the testing set that is obtained from the original train-test split described in Section 3.3.1 are fixed, and then randomly select $n$ percentage of samples in the original training set to form the $n\%$ sparsity-level training set. Accordingly, the four levels of data sparsity are set to 25%, 50%, 75%, and 100%. Meanwhile, due to the limited space, the experiment results on SGL, SGL-SUBD, and SGL-DifE are shown since these models have the best performance among the base models, SUBD-optimized models, and DifE-optimized models respectively. The Recall@20, NDCG@20, and MRR@20 results for the selected three models with respect to four different sparsity levels are presented in Figure 3.7.

It can be seen from the results that the SGL-DifE model achieve the highest values of Recall@20, NDCG@20, and MRR@20 across all different sparsity levels cases, which illustrates the effectiveness and robustness of DifE-optimized models under diverse recommendation scenarios with varying data sparsity levels.
CHAPTER 3. PREFERENCE DIFFERENCES CAPTURING IN RECOMMENDER SYSTEMS

(a) Comparison on Recall@20.

(b) Comparison on NDCG@20.

(c) Comparison on MRR@20.

Figure 3.7: $MRR$ results on different $Top@K$ recommendation scenarios.
In summary, the comparison results demonstrate that the proposed DifE-optimized models are capable of achieving higher prediction accuracy and better recommendation satisfaction. Moreover, compared to base models, the DifE-optimized models are robust to sparsity level changes in real world recommendation scenarios. In direct terms, it can be confirmed that the proposed loss function benefits point-wise recommendation models to generate more personalized and satisfied predictions, which can also be widely applied in different recommendation scenarios (Answer for RQ1).

### 3.4.2 Overall performance comparison (RQ2)

To compare the performance of different models, the results for Recall@20 and NDCG@20 on three datasets are presented in Table 3.4.

<table>
<thead>
<tr>
<th>Compared Methods</th>
<th>BXBooks</th>
<th>AmazonMls</th>
<th>Epinions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Recall</td>
<td>NDCG</td>
<td>Recall</td>
</tr>
<tr>
<td>DMF</td>
<td>0.0543</td>
<td>0.0321</td>
<td>0.1349</td>
</tr>
<tr>
<td>NeuMF</td>
<td>0.0627</td>
<td>0.0392</td>
<td>0.1494</td>
</tr>
<tr>
<td>DeepCF</td>
<td>0.0665</td>
<td>0.0423</td>
<td>0.1571</td>
</tr>
<tr>
<td>NeuMF-DifE</td>
<td>0.0778</td>
<td>0.0512</td>
<td>0.1737</td>
</tr>
<tr>
<td>Improvement</td>
<td>+16.99%</td>
<td>+21.04%</td>
<td>+10.57%</td>
</tr>
<tr>
<td>NGCF</td>
<td>0.0914</td>
<td>0.0599</td>
<td>0.1862</td>
</tr>
<tr>
<td>NIA-GCN</td>
<td>0.0957</td>
<td>0.0621</td>
<td>0.1867</td>
</tr>
<tr>
<td>LightGCN</td>
<td>0.0988</td>
<td>0.0643</td>
<td>0.1900</td>
</tr>
<tr>
<td>SGL</td>
<td>0.1002</td>
<td>0.0630</td>
<td>0.2017</td>
</tr>
<tr>
<td>UltraGCN</td>
<td>0.0966</td>
<td>0.0641</td>
<td>0.1951</td>
</tr>
<tr>
<td>GCN-DifE</td>
<td>0.1144</td>
<td>0.0722</td>
<td>0.2024</td>
</tr>
<tr>
<td>SGL-DifE</td>
<td>0.1147</td>
<td>0.0742</td>
<td>0.2067</td>
</tr>
<tr>
<td>Improvement</td>
<td>+14.47%</td>
<td>+15.40%</td>
<td>+2.48%</td>
</tr>
</tbody>
</table>

The observations are listed as follows:

- SGL-DifE consistently outperforms the state-of-the-art GCNs-based recommendation models on all datasets in Recall@20 and NDCG@20. Particularly, SGL-DifE achieves 14.47%, 2.48%, and 3.05% improvement of recall over the best existing base-
The improvements by SGL-DifE on NDCG are also promising, especially for BXBooks, which presents a substantial 15.40% increase. The largest improvement in BXBooks can be attributed to the larger rating scale compared to the other two datasets. More specifically, the larger rating scale indicates more diverse ratings. Therefore, the LightGCN-DifE model is able to capture the preference differences between a pair of ratings and hence produce better embeddings for both users and items.

- NeuMF-DifE also demonstrates significant improvements in Recall@20 and NDCG@20 compared to the existing neural-based recommendation models. Meanwhile, as a rating-based neural architecture, DMF presents an unsatisfactory prediction. The reason could be that DMF model emphasizes the interacted items with high ratings while diminishing the effect of interacted low-rated items, which exacerbates the data sparsity problem and leads to lower recommendation accuracy.

Higher Recall@20 and NDCG@20 only reveal more items that have been recommended. Nevertheless, the quality (ratings) of the recommended list is also quite crucial in real-world recommendation scenarios. In other words, it is important to avoid recommending low-rated items to users in order to increase their satisfaction with the recommender system. From this perspective, the statistics on the ratings of successfully recommended items are compiled to compare the more detailed recommendation quality among existing models. The comparison results are demonstrated in Figure 3.8.

As shown in Figure 3.8, the stacked bar chart draw in the polar coordinate system is utilized to exhibit the statistics of successfully recommended items. The four different colors correspond to four different rating value groups. In addition, the lengths of different colors in the same bar indicate the ratio of the specific value groups. Obviously, our DifE-optimized models are more successful in recommending high-rated items and obtain a lower rate in recommending low-rated items in all three datasets. The DifE-optimized model achieves the best performance on the Epinions dataset due to the relatively even rating distribution. Additionally, the moderate improvement on the AmazonMIs dataset can be attributed to the extremely unbalanced rating distribution, making it harder for
3.4. EXPERIMENTS RESULTS

(a) Comparison among Neural-based Models.

(b) Comparison among GCNs-based Models.

Figure 3.8: The statistics of hit frequency for different rating value groups of Neural-based and GCNs-based models.

our proposed models to distinguish the preference differences and, consequently, limit their performances.

All the analyses above confirm that the proposed SGL-DifE model obtains the highest recommendation accuracy and demonstrates better recommendation quality. (Answer for RQ2)

3.4.3 Hyper parameters analysis (RQ3)

As presented and discussed in Chapter 3 Section 3.2.4, similar neighbors’ ratings are utilized to modify the original rating into a more informative preference $p_{ui}$. To investigate
CHAPTER 3. PREFERENCE DIFFERENCES CAPTURING IN RECOMMENDER SYSTEMS

Figure 3.9: Hyper-parameters comparison on number of neighbors.

the necessity and the impact of the neighbor-based preference adjusting strategy, the neighbor number are changed from 0 to 50 with the step size of 10 to compare the performance of different LightGCN-DifE models. The discussion focuses on the experiments on BXBooks and Epinions datasets, but it would be applicable for other datasets as well as other DifE-optimized models.

As shown in Figure 3.9, it is apparent that the optimal neighbor numbers are 10 and 20 on datasets BXBooks, and Epinions, respectively. Compared with the case when there is no neighbor to adjust rating, i.e., the original case, the improvement declares the effectiveness of preference adjusting strategy discussed in Chapter 3 Section 3.2.4. Compared to the optimal neighbor number in Epinions, the best value on BXBooks is much smaller. This can be ascribed to BXBooks naturally has more diverse explicit expressions because of its 1 to 10 rating scale. Consequently, utilizing neighbors in order to produce reliable normalized ratings would be of minor significance. Furthermore, the results also indicate that the number of neighbors should not be too large or too small. This may be caused by the fact that when the number of neighbors is too small, the user’s rating will be significantly affected by neighbors with weak normalization in sparse datasets. However, when the number of neighbors is large, it will contain more non-similar users (items) and thus degrade the model’s performance.
3.5 Summary

In this chapter, a novel point-wise loss function is proposed to capture pair-wise information and enable recommender systems to produce better recommendations. Four point-wise recommendation models are selected and trained with the proposed loss function, named DifE-optimized models. Comprehensive experiments on several datasets demonstrate that these DifE-optimized models are able to capture personalized information preserved in pair-wise differences and can generate better recommendations. The superiority of these DifE-optimized models demonstrates the excellence and generality of the proposed learning strategy.

The main contributions of the work in this chapter includes:

- A novel model-independent training strategy is proposed which can utilize a point-wise loss function to capture personalized information retained in pair-wise preference differences.

- The proposed loss function is integrated with several existing recommenders and establish four optimized recommendation models, named MF-DifE, NeuMF-DifE, LightGCN-DifE and SGL-DifE, which can model personalized preferences and produce better predictions.

- Comparative experiments are conducted on three real-world datasets. Extensive results show that these optimized models achieve significant improvements compared to their corresponding baselines and also outperform a variety of recent recommendation methods, revealing the excellence and generality of the proposed point-wise loss function.
CHAPTER 3. PREFERENCE DIFFERENCES CAPTURING IN RECOMMENDER SYSTEMS
Chapter 4

Hierarchical Information Modeling in Recommender Systems

This chapter presents a hierarchical recommender, named UGCN, that can capture the multi-resolution collaborative signals from user (item) implicit hierarchies. First, the background and motivation of the UGCN recommender are presented. The implicit user (item) hierarchical information modeling is then formulated. Based on this, a U-shaped hierarchical recommender is comprehensively presented together with its theoretical analysis. After that, the experiments are conducted on three million-size datasets, and their corresponding results are exhibited. The last section is the summary of this chapter.

4.1 Background

4.1.1 Motivation of hierarchical information modeling

As discussed in Chapter 2, the core target of the recommender system is to predict users’ interest in non-interacted items and make customized suggestions. Among all the different approaches, collaborative filtering, which focuses on exploiting past user-item interactions
to model user (item) preferences and generate personalized predictions, is the most popular and most widely applied recommendation paradigm. The typical collaborative filtering prototype represents an individual user or item with a low-dimensional embedding and then generates customized predictions based on those learned embeddings. Matrix factorization [134] is an early example of this collaborative filtering technique. Specifically, the classical MF algorithm directly encodes user (item) one hot vector into user (item) embeddings and adopts dot product among embeddings to model a user’s preference for an item. Later, several researchers applied different strategies to optimize the MF-based model and obtain more robust and informative embeddings. In recent years, with the superiority of deep learning techniques, deep neural architecture has been successfully applied to many machine learning scenarios, including recommender systems. DMF [40] is one of the most representative neural-based recommenders that uses multi-layer perceptions to learn user (item) embeddings from binary interaction vectors. Other neural-based models such as NeuMF [2] and DeepCF [62], also prove the power of deep learning techniques in dealing with recommendation problems. Nowadays, GCNs-based recommendation systems have attracted more recent research attention. More specifically, as the user-item interaction data can be naturally modeled as graphs, the GCN-based models are capable of exploiting higher-order connectivity between users and items and, therefore, can obtain better user (item) embeddings.

Based on the analysis above, the algorithm for generating user (item) embeddings plays a crucial role in the success of the existing CF-based recommenders. However, neglecting user (item) inherent hierarchical information still prevents these recommenders from producing optimal predictions. More concretely, most existing CF-based recommendation models only focus on modeling individual items or users independently, which cannot fully capture the items’ (or users’) hierarchical information and hence fail to precisely model personalized preference. Take the movie Frozen as an example, it belongs to the subgenre "Family Animation" and can be further categorized into the genre "Animation", demonstrating a hierarchical structure of "individual → sub-genre → genre". Similarly, users may present a similar hierarchy of "individual → occupation → age". Since items of the same subgenre (or genre) are likely to have similar attributes, they are likely to
acquire similar preferences \[13\]. Thus, hierarchical information about items or users can very likely improve the preference modeling process of recommender systems.

It is worth pointing out that in real-world recommendation scenarios, explicit hierarchies are often not applicable \[12\]. For this reason, some researchers have used MF-based models to learn implicit item (user) hierarchies \[13\]. Specifically, the implicit item (user) hierarchy can be easily obtained by decomposing the original item (user) embedding matrix into several smaller and more compressed item (user) matrices, respectively. The success of these models confirms the validity of exploring implicit hierarchies \[139\]. However, there are still two limitations that limit the performance of these models, which can be summarized as the limited representation based on a simple MF model and the neglect of diverse collaboration signals in different hierarchical levels. More specifically, recent studies have argued that the basic MF-based models simply adopt the interaction between users and items as the ultimate objective function, but ignore the potential similarity signals stored in the interaction \[140\]. From this perspective, models based on GCNs, which can naturally model the user-item relationship in the interaction graph structure, propose a more reasonable and efficient way to build recommender systems. In addition to the limitations imposed by simple models, existing models that focus solely on building two separate multilayer architectures are problematic. More directly, exploring parallel user (item) hierarchies only shows the multi-level relationships within a user or item. However, the core information contained in the hierarchy is the multi-resolution collaborative signals stored in different levels. For example, given the interaction that a user 'Alice' like a movie 'Frozen', exploring the structure of 'Alice → Second Grade Nursery Student → Kids' and 'Frozen → Family Animation → Animation' merely displays the inner association for a user or an item. Indeed, this interaction not only reveals 'Alice’s' unique preference for 'Frozen', but also represents a shared interest that 'Second Grade Nursery Students' may prefer 'Family Animation', or announces a more general signal that 'Kids' prefer 'Animation'. Clearly, these multi-resolution collaborative signals will help generate better recommendation predictions.
CHAPTER 4. HIERARCHICAL INFORMATION MODELING IN RECOMMENDER SYSTEMS

4.1.2 Contribution of this chapter

This chapter proposes an implicit user (item) hierarchical exploration model, i.e., UGCN, which utilizes a u-shaped hierarchical graph convolution model and is able to capture user-item collaboration signals in different resolution scales. Specifically, the u-shaped structure uses a pooling operation to adaptively compress the original user-item interaction graph into smaller compressed graphs. In other words, by merging similar nodes in the fine-grained graph into a new group node in the coarse-grained graph, collaboration signals at different resolution scales can be better captured by graph convolution operations on different compressed graphs. Then, the captured collaborative signals are gradually projected back to the original nodes through symmetric un-pooling operations. With the multi-resolution collaborative signal captured in final item (user) embeddings, better personalized recommendations can be ultimately generated.

In summary, this chapter has the following main contributions.

- A u-shaped hierarchical recommender based on graph convolutional networks is proposed which can captures diverse collaborative signal from a stacked multi-layer graph architecture.

- Comparative experiments are conducted on three million-size datasets. The experimental results show that the proposed model achieves the best recommendation performance and obtains more personalized recommendation results simultaneously.

4.2 Preliminary

4.2.1 The basic collaborative filtering model

The basic idea of traditional model-based collaborative filtering algorithms can be summarized as utilizing a model to project an individual user (item) into a shared latent space and modeling a user’s preference given an item by the inner product of the corresponding user
(item) embeddings. To make it more clear, $e_u$ and $e_i$ are adopted to represent the learned personalized embedding for user $u$ and item $i$ respectively. Moreover, as classic models merely rely on interactions and neglects high-level hierarchical context information, hence $e_u$ and $e_i$ are formulated as $e_{u-c0}$ and $e_{i-c0}$, where $c0$ indicates the context extracted in the $0th$ hierarchy. Then, the preference of user $u$ given item $i$ can be computed as follows:

$$\hat{y}_{ui} = e_u \cdot (e_i)^T = e_{u-c0} \cdot (e_{i-c0})^T \quad (4.1)$$

### 4.2.2 Modeling implicit hierarchies

In the basic model, the interaction is often considered as the unique preference of a single user given a specific item. However, as a member of our modern society, a user’s behavior is inevitably influenced by implicit contextual information and would demonstrate some implicit context-based preferences. When it comes to interaction modeling recommender systems, the implicit contextual information can be interpreted as the group-level collaborative signal. More specifically, according to the example in Section 1, a user’s interaction is not only demonstrating an individual’s personalized interest but also indicates several informative group-level tendencies. As these group-level tendencies (multi-resolution collaborative signal) are useful for understanding the user’s behavior and would be helpful for generating customized recommendations, a hierarchical model becomes a more reasonable way to construct general recommender systems. To be more specific, the user embedding $e_u$ and item embedding $e_i$ are decomposed into the summation of different representations, which corresponds to the preference information learned among different levels of hierarchical architectures. Take a $n$-levels hierarchical model as an example; the final user embedding $e_u$ and item embedding $e_i$ can be calculated through:

$$e_u = e_{u-c0} + e_{u-c1} + e_{u-c2} + \ldots + e_{u-cn} \quad (4.2)$$

$$e_i = e_{i-c0} + e_{i-c1} + e_{i-c2} + \ldots + e_{i-cn} \quad (4.3)$$
where $e_{u-c0}$ and $e_{i-c0}$ represent the personalized embedding for individual user $u$ and item $i$ respectively. While, $e_{u-cx}, x \in (1,n)$ and $e_{i-cy}, y \in (1,n)$ demonstrate the diverse group-level collaborative signals. It is worth noticing that $e_{u-cx}, x \in (1,n)$ and $e_{i-cy}, y \in (1,n)$ would be shared among the group members within the same group at the same hierarchical level. Moreover, the greater $x$ value in $e_{u-cx}, x \in (1,n)$ corresponds to a more compressed group representation, which indicates $e_{u-cx}$ would be shared by a larger number of group members. With the defined $e_u$ and $e_i$ in hierarchical architecture, the user $u$’s preference given item $i$ can be estimated through:

$$\hat{y}_{ui} = e_u \cdot (e_i)^T = (e_{u-c0} + e_{u-c1} + \ldots + e_{u-cn}) \cdot (e_{i-c0} + e_{i-c1} + \ldots + e_{i-cn})^T \quad (4.4)$$

### 4.3 UGCN Recommender

As discussed above, the essential point for constructing a hierarchical architecture is to extract multi-resolution collaborative signals and encode these informative signals into corresponding embeddings. To tackle this issue, the idea of utilizing a hierarchical graph structure are adopted to enable generating diverse group-level embeddings on different graph scales. In other words, the proposed recommender aim to gradually compress the original user-item interaction graph into several group-level interaction graphs. Hence, a node in the compressed graph could represent a group of similar users (items), and an edge in the compressed graph would indicate the shared behavior of group members. Taking a step further, the node embedding learned on diverse graph architectures of different scales can naturally capture the corresponding diverse group-level collaborative signals shared among certain group members. Following the analysis, the whole hierarchical recommender construction are displayed in three parts: item (user) hierarchies extraction (i.e., graph architecture compressing), graph node embedding learning (i.e., multi-resolution collaborative signal modeling), and diverse embedding integrating.
4.3.1 Item (user) hierarchies extraction

To reasonably explore item (user) hierarchies and properly compress the original user-item interaction graph, the core point of collaborative filtering that similar users (items) might exhibit similar preferences are followed. In other words, by merging the similar nodes in the finer scaled graph into a group node in the coarser scaled graph, the generated diverse graph architectures could successfully reveal the user (item) hierarchies relationships. More importantly, the group node in a coarser scaled graph can automatically exhibit the shared interest of those similar nodes in a finer scaled graph. To simplify the discussion, the common pooling operation are adopted to represent the process of compressing a finer graph into a coarser one, which can be defined as:

\[ G'(U', I', E') = \text{Pooling}(G(U, I, E)) \]  

(4.5)

where \( G(U, I, E) \) and \( G'(U', I', E') \) is the finer scaled graph and coarser scaled graph respectively. Expressly, to implement the pooling operation, the following two steps are utilized: 1) group formation: assembling similar neighbors based on the similarity calculation on 1-hop connections; 2) node merging: generating a new group node based on the assembled members, where the edge of a new group node is mainly determined by the common connections of similar nodes. Meanwhile, a predefined ratio \( \alpha \) is utilized to control the percentage of the nodes selected for grouping, enabling the construction of more flexible hierarchical architectures.

4.3.2 Graph node embedding learning

After obtaining the graphs of diverse scales, capturing the diverse collaborative signal in different graphs and learning informative embeddings for each node in each structure become the essential issue. To this end, a graph convolution operation are applied on graph architecture to encode informative collaborative signals into node embeddings. Specifically, the neighborhood aggregation proposed by LightGCN is adopted to simplify the whole
convolutional calculation process, which can be represented as:

\[ e_u^{(k+1)} = \sum_{i \in N_u} \frac{1}{\sqrt{|N_u|} \sqrt{|N_i|}} e_i^{(k)} \]  

(4.6)

\[ e_i^{(k+1)} = \sum_{u \in N_i} \frac{1}{\sqrt{|N_i|} \sqrt{|N_u|}} e_u^{(k)} \]  

(4.7)

where \( e_u^{(k)} \) and \( e_i^{(k)} \) respectively denote the embedding of user \( u \) and item \( i \) generated after \( k \) convolutional operations. Meanwhile, \( N_u \) and \( N_i \) denote the set of items that are interacted by the user \( u \) and the set of users that interact with the item \( i \), respectively. The symmetric normalization term \( 1/(\sqrt{|N_i|} \sqrt{|N_u|}) \) follows the design of standard GCNs to avoid the scale of embeddings increasing with graph convolution operations. The final embeddings after operating \( m \) convolutional operations can be computed as:

\[ e_u = \frac{1}{m} \sum_{k=0}^{m} e_u^{(k)} \quad ; \quad e_i = \frac{1}{m} \sum_{k=0}^{m} e_i^{(k)} \]  

(4.8)

It is worth noticing that the graph convolution operations will be independently implemented on each graph structure. The initialization for graph nodes will only be applied to the original user-item interaction graph. Moreover, the initial node embeddings in the coarser graph can be easily generated by the mean embeddings of similar nodes in the finer graph that are selected and merged into the corresponding group nodes.

### 4.3.3 Diverse embedding aggregation

Following the multi-resolution collaborative signal extraction process, properly propagating the extracted information back to final embeddings is also crucial. Symmetric to pooling operation, un-pooling operation are adopted to gradually restore the coarser scaled architecture back to the finer ones:

\[ G(U, I, E) = UnPooling(G'(U', I', E')) \]  

(4.9)

As a group-level node in a coarser scaled graph represents all of its group members’ collaborative interaction patterns, the collaborative information contained in the group node is directly copied to its group members in the finer scaled graph. In this case, the
4.3. UGCN RECOMMENDER

Figure 4.1: An illustration of a 3-levels UGCN Recommender. In this example, the original user-item interaction graph are hierarchically stacked with two compact graphs generated by Pooling operations. Then, simplified GCNs layers are implemented to extract the multi-resolution collaborative signals at three graphs of different scales. The Un-pooling operations would gradually restore the finer graph architecture and integrate the informative signals into final user (item) embeddings.

group-level collaborative signal obtained in a coarser scaled graph can successfully be propagated back to the corresponding group members.

4.3.4 UGCN architecture

To systematically extract and encode diverse group-level information into user (item) preference, a hierarchical recommendation architecture, named UGCN, is proposed. More specifically, the first layer of the UGCN is constructed as a typical GCNs-based model, which is used for capturing user’s (item’s) basic preference from the original bipartite graph. Then, to extract and integrate the diverse group-level contexts into basic preference, an u-shaped encoding-decoding process is adopted. Concretely, the crucial encoding process is constructed by stacking several encoding parts hierarchically. Each encoding part contains a pooling operation and the corresponding graph node embedding learning process to extract valuable group-level collaborative signals for the current graph scale.
After that, the un-pooling operation in the decoding part is designed to restore the coarser-scaled graph to its finer-scaled one. Hence, with the group-level context propagated from the coarser scaled graph and the basic characteristics from the skip-connections, the graph node embedding learning process in the decoding part enables the UGCN model to learn more informative representations at the current graph scale. Once the graph architecture is fully restored, the valuable context information could automatically be encoded into user’s (item’s) preferences. A 3-level UGCN recommender is depicted in Figure 4.1.

4.3.5 Theoretical analysis for UGCN recommender

In this section, the in-depth analyses of the UGCN Recommender are presented, which aims to answer the question: how does this hierarchical recommender benefit real-world recommendation problems? To answer this question, a 3-levels UGCN recommender is depicted in Figure 4.1 as an example.

Given a user $u$ with his (her) initial embedding $e'_u(0)$, the final embedding $e_u$ produced by a 3-levels UGCN model can be represented as:

$$
e_u = \text{GCN}_0^4(\text{GCN}_0^2(e'_u(0)) + \text{GCN}_1^2(\text{GCN}_1^2(e'_u(1)) + \text{GCN}_2^2(e'_u(2)))) \quad (4.10)$$

$$e'_u(1) = \text{Pooling}(\text{GCN}_0^2(e'_u(0))) \quad (4.11)$$

$$e'_u(2) = \text{Pooling}(\text{GCN}_1^2(e'_u(1))) \quad (4.12)$$

where $\text{GCN}_n^m$ represents operating $m$ graph convolutional layers on the graph structure at $(n+1)$-th level of the hierarchical graph model. For example, $\text{GCN}_0^2$ indicates operating two graph convolution layers on the original user-item interaction graph. Meanwhile, $e'_u(1)$ and $e'_u(2)$ represent the computed initial embedding of the merged group node in graph architectures at $2-na$ and $3-na$ level of hierarchical models, respectively. Moreover, the Un-pooling operation is discarded in the whole calculation as the Un-pooling operation simply copies the learned group node embedding back to its corresponding group members.

By properly expanding the Equation (4.10), the following equation can be obtained:

$$e_u = \underbrace{\text{GCN}_0^4(e'_u(0)) + \text{GCN}_0^2(\text{GCN}_1^2(e'_u(1))) + \text{GCN}_2^2(\text{GCN}_2^2(e'_u(2))}}_{e_{u-c0}} + \underbrace{\text{GCN}_0^2(\text{GCN}_1^2(e'_u(1))) + \text{GCN}_2^2(\text{GCN}_2^2(e'_u(2))}}_{e_{u-c1}} + \underbrace{\text{GCN}_2^2(\text{GCN}_2^2(e'_u(2))))}_{e_{u-c2}} \quad (4.13)$$

94
As the final embedding generated by the UGCN recommender can be interpreted as the hierarchical structure discussed in Section 4.2. The simplified expressions is hence be adopted in the following discussion.

Formally, for user $u$, the gradient of the BPR loss w.r.t. the trainable embedding $e_{u(0)}$ is as follows:

$$\frac{\partial BPR}{\partial e_{u(0)}} \propto \sum_{i \in N_u} \sum_{j \notin N_u} \frac{-e^{-\hat{x}_{uij}}}{1 + e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial e_{u(0)}} \hat{x}_{uij} - \lambda \quad (4.14)$$

where $\hat{x}_{uij}$ are the predicted preference discrepancies among user interested and uninterested items. The detailed computation of BPR loss and $\hat{x}_{uij}$ can be found in Equation (4.18) and (4.19). Then, $\hat{x}_{uij}$ can be further expanded as follows:

$$\hat{x}_{uij} = (e_{u-c0} + e_{u-c1} + e_{u-c2}) \cdot [(e_{i-c0} + e_{i-c1} + e_{i-c2}) - (e_{j-c0} + e_{j-c1} + e_{j-c2})] \quad (4.15)$$

As presented above, the initial group embeddings $e_{u(1)}'$ and $e_{u(2)}'$ showed above can be calculated through a linear model contains $e_{u(0)}'$, it can be finally obtained:

$$\frac{\partial}{\partial e_{u(0)'}} \hat{x}_{uij} = \alpha \cdot [(e_{i-c0} - e_{j-c0}) + (e_{i-c1} - e_{j-c1}) + (e_{i-c2} - e_{j-c2})] \quad (4.16)$$

where $\alpha$ is the constant number computed through the partial derivatives of $(e_{u-c0} + e_{u-c1} + e_{u-c2})$ with respect to $e_{u(0)}'$. Based on the Equation (4.14) and (4.16), it is apparent that the U-shape hierarchical model can contain valuable group-level differences information during the model training process. Hence, it can be concluded that the UGCN model can produce more informative embeddings and generate better predictions.

### 4.4 Experiments Settings

This chapter presents the experiment settings from the perspective of data-set, evaluation metrics, and implementation details.
CHAPTER 4. HIERARCHICAL INFORMATION MODELING IN RECOMMENDER SYSTEMS

Table 4.1: Statistics of experimental datasets.

<table>
<thead>
<tr>
<th>Statistics</th>
<th>Users</th>
<th>Items</th>
<th>Interactions</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ml1m</td>
<td>6022</td>
<td>3043</td>
<td>995,154</td>
<td>0.05431</td>
</tr>
<tr>
<td>Gowalla</td>
<td>29,858</td>
<td>40,981</td>
<td>1,027,370</td>
<td>0.00084</td>
</tr>
<tr>
<td>Yelp2018</td>
<td>31,668</td>
<td>38,048</td>
<td>1,561,406</td>
<td>0.00130</td>
</tr>
</tbody>
</table>

4.4.1 Dataset description

The experiments are conducted on three public million-size datasets, called Movielens 1M (Ml1m), Gowalla, and Yelp2018. The detailed dataset statistics are shown in Table 4.1.

For each dataset, 80% of each user’s interactions are randomly split to construct the training set, while the remainder forms the test set. For each training set, 10% of interactions are randomly separated from training set as a validation set, which are utilized to validate the prediction performance during modeling training phase and tune the model’s hyperparameters accordingly.

4.4.2 Evaluation metrics

To evaluate the performances of the distilled hierarchical recommender and the state-of-art baselines, several evaluation metrics are employed. Recall@K and NDCG@K are firstly adopted to calculate the overall recommendation accuracy. The detailed computational equations for Recall@K and NDCG@K are presented in Chapter 3 Section 3.3.2. Except for accuracy evaluation, recommendation satisfaction is also essential for comprehensively evaluate the recommendation results. To this end, three publicly accepted evaluation matrices are adopted, which are MRR@K, Novelty@K and Converage@K. The detailed computational equations for MRR@K and Novelty@K are presented in Chapter 3 Section 3.3.2. Coverage demonstrates the percentage of items that have been recommended to customer. According to common sense, users are always pleased with mixed recommendation results rather than merely recommending the same items \([141]\). A higher Coverage value would often refers to higher recommendation satisfaction. The detailed equation for recommendation coverage \(R – \text{Coverage}\) and hit coverage \(H – \text{Coverage}\) are presented.
below.

\[
R - \text{Coverage}(K) = \frac{N_{\text{recommended}}}{||N^u||}
\]
\[
H - \text{Coverage}(K) = \frac{N_{\text{hit}}}{||N^u||}
\] (4.17)

where \(N_{\text{recommended}}\) and \(N_{\text{hit}}\) imply the number of different items that have been recommended and hit in one or more users’ top@\(K\) list, respectively; \(N^u\) is the set of total items. Last but not least, the above mentioned four metrics are designed for the evaluation on a single user \(u\). Their corresponding average values on all users would be treated as the final experimental results with respect to four different evaluation criteria.

### 4.4.3 Implementation for UGCN recommender

This section demonstrates the detailed implementation for UGCN recommender.

**1) Pooling Algorithm**

As presented in Section 4.3.1, a pooling algorithm is required to explore graph-based user (item) hierarchical architecture. Some existing pooling algorithms, such as DiffPool \[142\] and SAGPool \[143\], are designed for the pooling process in the original graph structure. Nonetheless, it is important to note that they are unsuited for recommendation situations that intrinsically generate bipartite networks. More specifically, different from the existing pooling algorithms, which target merging the one-hop neighbors, the pooling algorithm in UGCN recommender should only merge two-hop neighbors (i.e., user-to-user or item-to-item) since there exists an inherent independence between user sets and item sets. To this end, a more appropriate pooling algorithm that applies to the UGCN recommender is proposed and presented in Algorithm 1.

Notice that the nearest neighbor mentioned in line 4 is calculated through the cosine similarity with the users’ binary interactions. As a bipartite graph is adopted to model user-item interactions, the pooling algorithm would run separately on user nodes and item nodes to obtain the compressed bipartite graph and enable consecutive pooling operations.
Algorithm 1 Pooling Algorithm

Require: Pooling Rate $\alpha$, Graph $G(U, I, E)$

1: Group Set $GP \leftarrow \{\}$
2: Utilize Pooling Rate $\alpha$ to split the user node set $U$ into Pooling Set $U_p$ and Remaining Set $U_r$
3: for each node $u_i$ in $U_p$ do
4: $u_j \leftarrow$ the nearest neighbor of $u_i$ in Remaining Set $U_r$
5: if $u_j$ exist in group $GP_n$, where $GP_n \in GP$ then
6: Add $u_i$ into group $GP_n$
7: else
8: Add the new group $GP_m = \{u_i, u_j\}$ into $GP$
9: end if
10: end for
11: for each group $GP_n$ in $GP$ do
12: $u'_n \leftarrow$ a new group-level node representing the whole $GP_n$
13: $Adj_n \leftarrow$ the adjacency list of $u'_n$ containing all one hop edges from $GP_n$ to $I$, notice that an edge $eg_x$ from $u'_n$ to $i_m$ ($eg_x \in Adj_n$, $i_m \in I$) might appear more than once since similar users in the same group might conduct same interactions.
14: for each $eg_x$ in $Adj_n$ do
15: if $eg_x$ only appear once in $Adj_n$ then
16: Remove $e_x$ from $Adj_n$ with the probability of $1 - \frac{1}{|GP_n|}$
17: end if
18: end for
19: Generate final edge set $EG'_n$ of $u'_n$ from $Adj_n$
20: end for
21: Generate a smaller graph $G'(U', I, E')$ by replacing the assembled user nodes in the same group and their corresponding edges with the group-level $u'_n$ and $EG'_n$. return $G'(U', I, E')$
Moreover, to better select Pooling Set $U_p$, the random shuffling or other sorting strategies are adopted on the nodes list before splitting. In addition, Algorithm 1 simply splits the set into $U_p, U_r$ and assembles each user in $U_p$ with the nearest neighbor in $U_r$ rather than the most similar users in the whole user set. Hence, a similarity-based group formation method is proposed to replace line 2 to line 10 in Algorithm 1. The corresponding pseudocode is presented below.

**Algorithm 2** Similarity-based Group Formation Algorithm

1: Similar List $L_s \leftarrow []$
2: for each node $u_i$ in $U$ do
3: $u_j \leftarrow$ the nearest neighbor of $u_i$ in $U$
4: $s_{ij} \leftarrow$ the similarity value among $u_i$ and $u_j$
5: Append $(v_i, v_j, s_{ij})$ into $L_s$
6: end for
7: Sort $L_s$ on $s_{ij}$ where the highest value comes first
8: for each $u_i, u_j, s_{ij}$ in $L_s$ do
9: if $u_i, u_j \notin GP_n, \forall GP_n \in GP$ then
10: Add a new group $GP_k = \{u_i, u_j\}$ into $GP$
11: else if $u_i \in GP_n, GP_n \in GP \& u_j \notin GP_m, \forall GP_m \in GP$ then
12: Add $u_j$ to $GP_n$
13: else if $u_i \notin GP_n, \forall GP_n \in GP \& u_j \in GP_m, GP_m \in GP$ then
14: Add $u_i$ to $GP_m$
15: end if
16: if the number of processed users $\geq \alpha \times |U|$ then
17: Break loop
18: end if
19: end for

(2) Training Strategy

To properly train the UGCN recommender, the pairwise BPR loss that aims to enlarge the prediction differences among positive and negative samples are utilized. The corresponding
loss function is presented below:

\[
\hat{x}_{uij} = y_{ui} - y_{uj}
\]

\[
Loss_{BPR} = -\sum_{u=1}^{||N^u||} \sum_{i \in N_u \ j \notin N_u} \ln \sigma(\hat{x}_{uij}) + \lambda ||\Theta||
\]

(4.19)

where \(\sigma()\) denotes the activation function, \(\lambda\) controls the L2 regularization strength, and \(\Theta\) represents the parameter vector of the UGCN model. As the only trainable parameter is the initial embedding of the user-item interaction graph, the \(\Theta\) in Equation (4.19) equals the embedding matrix \(E\). Moreover, as the selected datasets are only contain each user’s positive samples (i.e., interactions), the negative samples utilized in model training phase will be randomly sampled from the non-interacted items of the same user.

4.4.4 Compared models

To thoroughly evaluation the proposed UGCN recommender, several state-of-the-art recommendation models are selected as baseline.

- **MF-BPR** [15]: This model presented a generic optimization criterion BPR-Opt for personalized ranking.

- **NeuMF** [2]: This is a state-of-the-art neural Collaborative Filtering model that used implicit user-item interaction.

- **HHMF** [144]: This is a recent proposed hidden hierarchical recommender which learned from the user-item scoring record and does not need prior knowledge.

- **NGCF** [67]: This is a state-of-the-art GCNs-based model which integrated the user-item interaction into the embedding process through the graph convolution operation.

- **NIA-GCN** [69]: This is a newly proposed GCNs-based model which explicitly exploits the user-user and item-item relationships through a pairwise neighborhood aggregator.
• **LR-GCCF** [145]: This is a state-of-art GCNs-based model which utilizes a residual network structure and linear graph convolution operations to alleviate the over-smoothing problem.

• **LightGCN** [3]: This is a newly proposed GCNs-based model that simplifies the design of graph convolution operation and only includes neighbor aggregation in convolution operations.

• **SGL** [71]: This is a newly proposed Graph Training Strategy which implements on the LightGCN model and achieves competitive performance.

Moreover, the following two UGCN recommender were also implemented:

• **UGCN-rdm**: The UGCN model adopts Algorithm 1 in pooling layers. Meanwhile, the random shuffling of the node list is applied before the splitting step mentioned in Algorithm 1 line 2.

• **UGCN-sim**: The UGCN model adopts Algorithm 2 to replace line 2 to line 10 in Algorithm 1, and hence the most similar node in the whole set are considered in the group formation process.

### 4.4.5 Other implementation details

The embedding size is fixed to 64 for all models, and the embedding parameters are initialized with the Xavier method [138]. The default learning rate is 0.002, and the default mini-batch size is 2048. The depth of hierarchical architecture is tested in the range of [2, 3, 4]. The number of graph convolution operations is tested from 1 to 3, and the predefined ratio $\alpha$ is validated in the range of [0.01, 0.99]. The early stopping and validation strategies are the same as LightGCN. The Adam [146] optimizer is also employed and used in a mini-batch manner. Moreover, the dropout mechanisms is adopted on every GCNs layer to mitigate over-fitting and the default dropout rate is 0.6. For the implementation
of the baseline models, the default training strategies and hyperparameters settings from the corresponding referenced papers are followed.

4.5 Experiment Results

In this section, the following research questions are targeted:

- **Q1**: How does the UGCN recommender perform compared to state-of-the-art models? Does UGCN achieve higher accuracy?

- **Q2**: Are multi-resolution collaborative signals helpful for generating better embedding? Can UGCN recommender produce more satisfactory recommendations?

- **Q3**: How do different hyper-parameter settings (e.g., the number of model’s hierarchical levels) affect the recommendation performance?

4.5.1 Prediction accuracy comparison (QR1)

To evaluate the overall prediction accuracy, Recall@20 and NDCG@20 are tested among all different models. The final results are presented in Table 4.2.

The proposed UGCN recommender consistently outperforms the baseline methods on all datasets. In particular, UGCN achieves 2.74%, 2.51%, and 2.23% improvement of recall@20 over the best baseline on ML1m, Gowalla, and Yelp2018, respectively. Meanwhile, the performance of UGCN on NDCG@20 is also outstanding, presenting 3.38%, 2.51%, and 1.45% enhancements. Moreover, compared to MF-based and neural-based models, the significant improvements among GCN-based models reveal the superiority of GCNs in handling embedding learning tasks, which is consistent with the discussion in Section 2. The results also demonstrate that the HHMF model and UGCN recommender perform better than basic-MF and existing GCN-based models, respectively, confirming
Table 4.2: The comparison of overall performance among UGCN and baseline methods.

<table>
<thead>
<tr>
<th>Compared Methods</th>
<th>MI1m Recall</th>
<th>MI1m NDCG</th>
<th>Gowalla Recall</th>
<th>Gowalla NDCG</th>
<th>Yelp2018 Recall</th>
<th>Yelp2018 NDCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>MF-BPR</td>
<td>0.2101</td>
<td>0.1787</td>
<td>0.1291</td>
<td>0.1109</td>
<td>0.0433</td>
<td>0.0354</td>
</tr>
<tr>
<td>NeuMF</td>
<td>0.2297</td>
<td>0.1886</td>
<td>0.1399</td>
<td>0.1212</td>
<td>0.0451</td>
<td>0.0363</td>
</tr>
<tr>
<td>HHMF</td>
<td>0.2311</td>
<td>0.2025</td>
<td>0.1477</td>
<td>0.1283</td>
<td>0.0498</td>
<td>0.0384</td>
</tr>
<tr>
<td>NGCF</td>
<td>0.2513</td>
<td>0.2511</td>
<td>0.1570</td>
<td>0.1327</td>
<td>0.0579</td>
<td>0.0477</td>
</tr>
<tr>
<td>NIA-GCN</td>
<td>0.2359</td>
<td>0.2243</td>
<td>0.1359</td>
<td>0.1358</td>
<td>0.0599</td>
<td>0.0491</td>
</tr>
<tr>
<td>LR-GCCF</td>
<td>0.2231</td>
<td>0.2124</td>
<td>0.1519</td>
<td>0.1358</td>
<td>0.0561</td>
<td>0.0343</td>
</tr>
<tr>
<td>LightGCN</td>
<td>0.2576</td>
<td>0.2427</td>
<td>0.1830</td>
<td>0.1554</td>
<td>0.0649</td>
<td>0.0530</td>
</tr>
<tr>
<td>SGL</td>
<td>0.2700</td>
<td>0.2547</td>
<td>0.1781</td>
<td>0.1501</td>
<td>0.0674</td>
<td>0.0553</td>
</tr>
<tr>
<td>UGCN-rdm</td>
<td>0.2774</td>
<td>0.2633</td>
<td>0.1871</td>
<td>0.1568</td>
<td>0.0679</td>
<td>0.0557</td>
</tr>
<tr>
<td>UGCN-sim</td>
<td><strong>0.2774</strong></td>
<td><strong>0.2633</strong></td>
<td><strong>0.1876</strong></td>
<td><strong>0.1587</strong></td>
<td><strong>0.0689</strong></td>
<td><strong>0.0561</strong></td>
</tr>
</tbody>
</table>

| %Improv.         | +2.74%      | +3.38%    | +2.51%         | +2.12%       | +2.23%         | +1.45%       |

The effectiveness of exploring users (items) hierarchies in recommender systems. In contrast to the HHMF model, the better performance of UGCN demonstrates that modeling the diverse resolution collaborative signal would be a better strategy than merely focusing on exploring user (item) hierarchies separately. Moreover, since a denser dataset would result in more accurate user (item) grouping and lead to more precise group-level signal extraction, this would explain why the most significant improvement is achieved in the ML1m dataset.

All the analyses above exhibit that the proposed UGCN recommender outperforms all the state-of-the-art models and achieves the highest recommendation accuracy.

4.5.2 Recommendation satisfaction comparison (QR2)

As discussed above, MRR@K, Coverage@K, and Novelty@K are employed to comprehensively evaluate the recommendation performances from the perspective of recommendation satisfaction. Moreover, two baseline modules LightGCN [3], and SGL [71] in the following discussion since their competitive results are displayed in Table 4.2. Meanwhile, the UGCN-sim model will be represented as the UGCN for simplification.
(1) MRR

First, MRR are tested on four different Top@K list of all three recommendation models. The final results are presented in Table 4.3.

Table 4.3: The comparison of overall performance among UGCN and baseline methods.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Models</th>
<th>MRR@20</th>
<th>MRR@40</th>
<th>MRR@60</th>
<th>MRR@80</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ml1m</td>
<td>LightGCN</td>
<td>0.4137</td>
<td>0.4158</td>
<td>0.4162</td>
<td>0.4164</td>
</tr>
<tr>
<td></td>
<td>SGL</td>
<td>0.4259</td>
<td>0.4279</td>
<td>0.4282</td>
<td>0.4283</td>
</tr>
<tr>
<td></td>
<td>UGCN</td>
<td>0.4322</td>
<td>0.4342</td>
<td>0.4346</td>
<td>0.4347</td>
</tr>
<tr>
<td>Gowalla</td>
<td>LightGCN</td>
<td>0.2886</td>
<td>0.2902</td>
<td>0.2933</td>
<td>0.2938</td>
</tr>
<tr>
<td></td>
<td>SGL</td>
<td>0.2753</td>
<td>0.2789</td>
<td>0.2801</td>
<td>0.2806</td>
</tr>
<tr>
<td></td>
<td>UGCN</td>
<td>0.2906</td>
<td>0.2942</td>
<td>0.2953</td>
<td>0.2959</td>
</tr>
<tr>
<td>Yelp2018</td>
<td>LightGCN</td>
<td>0.1093</td>
<td>0.1146</td>
<td>0.1163</td>
<td>0.1171</td>
</tr>
<tr>
<td></td>
<td>SGL</td>
<td>0.1144</td>
<td>0.1196</td>
<td>0.1213</td>
<td>0.1222</td>
</tr>
<tr>
<td></td>
<td>UGCN</td>
<td>0.1152</td>
<td>0.1209</td>
<td>0.1227</td>
<td>0.1240</td>
</tr>
</tbody>
</table>

As displayed in Table 4.3, the proposed UGCN consistently obtains higher MRR@K compared to LightGCN and SGL on all different Top@K cases. Moreover, SGL outperforms LightGCN in Ml1m and Yelp2018 but demonstrates the lowest MRR@K value in Gowalla. Considering that MRR@K reveals the average position of the first hit item in the recommendation list, the results in Table 4.3 prove UGCN’s superiority in enabling customers to reach their interested products quickly.

(2) Coverage

Same as MRR evaluation, coverage comparisons for LightGCN, SGL, and the proposed UGCN are also conducted on four different Top@K recommendation scenarios. The detailed experimental results for both $R – Coverage$ and $H – Coverage$ among all three models are presented below.

Based on the results in Figure 4.2, UGCN achieves higher $R – Coverage$ and $H – Coverage$ in all datasets. Taking the worst-case (top@20) as an example, $R – Coverage$ for UGCN are 57.34%, 42.20%, and 34.39%, which indicate the corresponding 1.60%, 4.95% and 2.81% improvement on three datasets compared to the best baseline. Similarly, $H – Coverage$ results also achieve relatively significant increases. Meanwhile, it is worth noticing that
4.5. EXPERIMENT RESULTS

(a) R-Coverage Comparison Results among LightGCN, SGL, and UGCN.

(b) H-Coverage Comparison Results among LightGCN, SGL, and UGCN.

Figure 4.2: Coverage Comparison Results

the Coverage result of SGL on the Yelp2018 dataset is under-performed by those of LightGCN, even though SGL obtains a higher recommendation accuracy according to Table 4.2. These results demonstrate that higher recommendation accuracy might not reveal better recommendations. In comparison, the proposed UGCN recommender obtains higher recommendation accuracy and higher coverage simultaneously, indicating the UGCN’s superiority.

(3) Novelty

Similar to MRR and coverage, the evaluation of novelty is also tested on four different Top@K recommendation lists for all three models. The Novelty@K results are displayed in Figure 4.3.

As shown in Figure 4.3, the average novelty of hit items in UGCN recommendation lists exceeds the corresponding novelty value for two baseline models, which reveals that the proposed UGCN recommender can generate less popular but more personalized recommendation predictions. Moreover, it is worth pointing out that the novelty evaluation targets the average item novelty within each recommendation list. Hence, a small improvement
in novelty can already demonstrate the distinct performance of UGCN in predicting more novel potential items.

In summary, compared to LightGCN and SGL, the UGCN recommender can not only rank the preferred items in higher positions but also increase the recommendation coverage and recommended item novelty simultaneously. Considering the difference between UGCN and the two baseline models, the Q2 can be answered by concluding that integrating multi-resolution collaborative signals allows recommender systems to precisely capture user (item) personalized preferences and leads to more satisfactory recommendations.

### 4.5.3 Hyper parameter analysis (QR3)

The above discussion indicates integrating diverse collaborative signals will benefit the recommender system. Nevertheless, it is still unclear how the number of model’s hierarchical levels impact the recommendation performance? To this end, the comparison experiments are conducted on M11m and Gowalla. The results are presented in Figure 4.4.

As shown in Figure 4.4, the best performance is achieved when the number of the model’s hierarchical levels are set to two and three on M11m and Gowalla, respectively. Compared to the one-level UGCN model (i.e., original LightGCN, which does not contain hierarchical architecture), the huge improvement confirms the necessity of modeling diverse collaborative signals. Meanwhile, the different optimal results on different datasets reveal that a fair number of UGCN’s hierarchical levels are always needed to be optimized on specific
Datasets during training.

Figure 4.4: Hyper-parameters Comparison Results
4.6 Conclusion

This chapter proposes a graph convolutional network-based hierarchical recommender, named UGCN, on a bipartite graph to capture diverse group-level collaborative signals and produce better recommendation predictions. Precisely, the proposed model utilizes a U-shaped architecture to gradually compress the original user-item interaction graph into smaller graphs by merging similar nodes together. Then, simplified graph convolutional operations can easily extract diverse group-level collaborative signals from compressed graphs. After that, those learned informative signals would be projected back to the final user (item) embeddings adaptively. Extensive and comprehensive experiments on three public datasets demonstrate that the UGCN recommender achieves significant overall prediction improvements over state-of-the-art models with better recommendations simultaneously.

The main contributions of the work in this chapter includes:

- A u-shaped hierarchical recommender based on graph convolutional networks is proposed which can captures diverse collaborative signal from a stacked multi-layer graph architecture.

- Comparative experiments are conducted on three million-size datasets. The experimental results show that the proposed model achieves the best recommendation performance and obtains more personalized recommendation results simultaneously.
Chapter 5

Prediction Distillation in Hierarchical Recommenders

This chapter presents two novel knowledge transfer and distillation strategies suitable for hierarchical recommenders, which enable compact hierarchical architecture to obtain competitive performance and model efficiency simultaneously. First, the background and motivation for applying knowledge distillation on hierarchical recommenders are presented. Then, the preliminaries of existing state-of-the-art KD methods are provided, followed by two proposed prediction-based distillation methods. After that, the experiments are conducted on three datasets, and their corresponding results are exhibited. The last section is the summary of this chapter.
CHAPTER 5. PREDICTION DISTILLATION IN HIERARCHICAL RECOMMENDERS

5.1 Background

5.1.1 Motivation of applying KD on hierarchical recommenders

As presented in Chapter 4, the graph-based hierarchical UGCN recommender can precisely capture users’ (items’) personalized preference information, and thereby produce customized recommendation predictions. Although stacked multi-graph architecture enables UGCN recommender to capture subtle information, it inevitably brings considerable computational expenses, limiting UGCN’s applications to real-world recommendation scenarios. In fact, increasing the model complexity to construct sophisticated recommendation architectures is a common and natural trend in recommender systems since the larger model with more learning parameters has a higher expression capacity and generally demonstrates better performance. However, endless increase of the complexity and the size of recommenders would be problematic. For example, high inference latency and long delays in updating model parameters could occur, which will bring about out-of-date recommendations. Hence, achieving recommendation performances while retaining the light recommendation architecture has become an increasingly prevalent concern in recommender systems.

To obtain a compact recommender with exceptional performances, KD, a model compression and knowledge transferring technique, has been widely incorporated in the model learning and construction. It teaches a compact student model to mimic a cumbersome teacher model by a teacher-student paradigm. Specifically, by distilling the essential knowledge from the fine-tuned cumbersome teacher, the compact student can be trained more efficiently and effectively, demonstrating outstanding performances with lower computational costs. Motivated by the state-of-the-art KD algorithms that match the class distributions, early KD approaches in recommendation scenarios focus on matching recommendation predictions generated by the teacher and the student model. Moreover, with the belief that the ranking information is more valuable than predictions, some researchers [107] [118] distill the ranking order from the teacher model to guide the training of the student. Several representation-based [122] or knowledge-based [129] distillation
methods are also proposed recently. These approaches achieve outstanding performances, while it is still worth pointing out that current studies merely focus on distilling from and constructing on non-hierarchical recommenders. Better KD methods suitable for hierarchical UGCN still need to be explored.

5.1.2 Contribution of this chapter

With the superiority of KD in generating compact recommendation models, two prediction-based KD strategies suitable for UGCN are studied. Firstly, a prediction-based distillation method that can enhance and enrich the recommendation dataset is proposed. To be specifically, by directly regarding the top-ranked items generated by sophisticated teachers as positive instances, the enhanced training data enable the compact UGCN to capture profitable information and demonstrate outstanding performance. Meanwhile, given the essential role of modeling user (item) hierarchies in building UGCN recommenders, another prediction-based distillation method that allows a compact student model to obtain a better hierarchical architecture is investigated. Concretely, by employing top-ranked teacher predictions to modify the student’s hierarchy extraction process, more precise neighbors can be identified in the compact student model, which will bring more accurate multi-resolution preferences modeling and better recommendation predictions.

The contribution of this chapter can be summarized as follows:

- Two prediction-based strategies are proposed to distill valuable knowledge from a sophisticated teacher to a compact student, enabling hierarchical recommenders to simultaneously achieve extraordinary recommendation performances and light model architecture.

- Comparative experiments are conducted on three datasets. Extensive results show that these proposed KD methods help the student model achieve outstanding recommendation accuracy and obtain more satisfactory predictions with higher coverage and novelty.
5.2 Discussions on existing KD approaches

There are many existing KD approaches employed for or applied to recommendation scenarios. The early attempts focus on distilling soft targets from sophisticated teacher to compact student. Since the soft targets can reveal latent and dark relationships among similar but independent classes [16], these soft target-based algorithms are proven effective in classical classification problems. However, in most real-world scenarios, the recommendation model only obtains binary interaction data, indicating no middle classes among preferred and disliked. The lack of latent and dark information inevitably limits the strength of soft target-based distillation strategies. To this end, some researchers [107] [119] propose ranking-based distillation for recommendation problems. In other words, by transferring the relative order of top-ranked items predicted by the teacher, the compact student can achieve outstanding ranking performances. Theoretically, the ranking-based distillation is quite fascinating. However, the existing ranking-based approaches [115] [107] [119] often over-exaggerate the ranking position of certain items, which easily leads to over-specification and over-fitting.

Except for the above prediction-based distillation strategies, several representation-based [122] and knowledge-based distillation strategies [129] are also proposed recently. Despite their promising performances, these models are inapplicable to graph-based hierarchical UGCN recommenders. Firstly, most representation-based approaches assume the existence of hidden layers within a sophisticated model, so the compact student can obtain a smaller parameter size without changing the dimension of the final embedding. Nevertheless, the only trainable and flexible parameter of a graph-based recommender's is the user (item) embedding. Fixing the dimension of embeddings prohibits knowledge transfer among different hierarchical models of different sizes. Some attempts [117] [105] are made to match embeddings of different dimensions, but their complicated strategies would bring extract computational and training expenses to a sophisticated recommendation architecture. On top of that, existing knowledge-based distillations [129] focus on distilling noisy and vague multi-resolution relationships from the teacher to the student. In contrast, the hierarchical UGCN recommender has already employed well-designed hierarchies extraction algorithms.
5.3 Proposed Methods

5.3.1 Prediction-based distillation for data enhancement

As discussed above, the idea of distilling top-ranked items from sophisticated teachers to teach the compact student is fascinating. However, the current model over-exaggerates the ranking order mismatches of restricted items, leading to over-specification and over-fitting.

In fact, the higher position of items in the teacher’s prediction list only reveals the greater confidence in the user’s potential preferences. These top-ranked items may obtain the almost same fondness of the target user. To this end, a more loosen ranking-based distillation strategy is proposed. Specifically, the top-ranked predictions generated by the sophisticated teacher will be treated as positive instances during the training of the compact student. In other words, the potential items filtered by the teacher model are used to enhance and enrich the ground truth training sets without extracting ranking-based sampling or smoothing strategies. The corresponding loss function for the compact student model is presented as follows:

$$\text{Loss}_{\text{student}} = -\sum_{u=1}^{||N^u||} \sum_{i \in (N_u \cup \pi_{1..K})} \sum_{j \notin N_u} \ln \sigma(y_u^i - y_u^j) + \lambda ||\Theta||$$

(5.1)

It is worth pointing out that the potential items in $\pi_{1..K}$ would not be excluded from the negative sampling pool. The reason is that: these potential positive samples are not as convinced as the ground-truth samples, and hence simply treating them as true positive would be bring more noisy to the training process. 

In summary, better KD methods suitable for hierarchical UGCN recommender still await exploration.
5.3.2 Prediction-based distillation for better hierarchies modeling

Unlike traditional neural-based models that only adopt historical interactions in the model training, graph-based recommenders incorporate user-item interactions in the construction of bipartite graph architecture \[67\] [3]. Moreover, the hierarchical UGCN recommender takes a step further, utilizing historical data to identify neighbors and to explore user (item) hierarchies. As the quality of constructed hierarchical architecture broadly impacts the extracted multi-resolution information, more precise user (item) hierarchies are always needed for hierarchical recommenders.

To this end, another prediction-based distillation designed for better hierarchies modeling is proposed. The corresponding knowledge transfer and distillation process can be summarized as follows:

\[
G_{s}'(U', I', E') = \text{Pooling}(G_s(U, I, E^{t\text{-update}}))
\] (5.2)

where \(G_s(\cdot)\) and \(G_{s}'(\cdot)\) are the finer-scaled graph and coarser-scaled graph of the student model, respectively. Compared to the pooling process presented in Chapter 4, the graph edges \(E^{t\text{-update}}\) (i.e., user-item connections) would be updated by adding the user-item connections based on top-ranked teacher predictions. In other words, the top-ranked predictions generated by the sophisticated teacher are incorporated into the student model’s hierarchical architecture exploration process. With these extra edges, a more reasonable hierarchical structure of the student model can be modeled, together with more informative multi-resolution collaborative signals. And finally, the compact student model is able to demonstrate competitive recommendation performances.

5.4 Experiments Settings

This chapter presents the experiment settings from the perspective of dataset, evaluation metrics, and implementation details.
5.4. EXPERIMENTS SETTINGS

5.4.1 Dataset description

Considering the computational cost during the training for the sophisticated teacher model, three datasets, BXBooks, AMIs, and Epinions, suitable for evaluating the proposed methods are thus selected to conduct the comparison experiments. The detailed dataset statistics can be found in Chapter 3 Table 3.1. The following section’s analysis is also applicable to other large datasets.

For each dataset, 80% of each user’s interactions are randomly split to construct the training set, while the remainder forms the test set. For each training set, 10% of interactions are randomly separated from training set as a validation set, which are utilized to validate the prediction performance during modeling training phase and to tune the model’s hyperparameters accordingly.

5.4.2 Evaluation metrics

To evaluate the performances of the distilled hierarchical recommender and the state-of-art baselines, several evaluation metrics are employed. More specifically, Recall@K and NDCG@K are adopted to calculate the overall commendation accuracy. Coverage@K (including both R-Coverage@K and H-Coverage@K) and Novelty@K are implemented to comprehensively evaluate the prediction results from the perspective of recommendation satisfactions. The detailed computational equations for these evaluation metrics are presented in Chapter 3 Section 3.3.2 and Chapter 4 Section 4.4.2.

5.4.3 Implementation of the teacher model

As discussed above, the teacher model in a KD paradigm aims to capture valuable information through sophisticated architectures. Given the superior performance of UGCN displayed in see in Chapter 4), the hierarchical UGCN recommender with similarity-based pooling algorithm is employed as the teacher model. Meanwhile, the default embedding
size of the teacher model is enlarged to 128, enabling teacher model to capture more complex and profitable information. Other implementation details, such as BPR training strategy, negative sampling, and hyper-parameter settings, are identical to the original UGCN implementations presented in Chapter 4.

5.4.4 Implementation of the student model

The implementation of the student model is analogous to that of teacher model. While several differences are worth-noticing. First, the embedding size of the student model is tested in the range of $[32, 64]$. It is because that the student model is designed for the recommendation inference stage and it often requires for light and compact architectures. Additionally, the number of top-ranked predictions is set in the range of $[1, 3, 5, 7]$. The student model trained by prediction-based distillation for data enhancement utilizes on the original training data to extract user (item) hierarchies and incorporates the teacher’s top-ranked predictions during its model training process. In contrast, the student model trained by prediction-based distillation for better hierarchies modeling aggregates the predictions with original training data to calculate new similarity matrix and relies on the new similarity matrix to obtain new hierarchical architecture. The similarity among user-to-user (item-to-item) pair is calculated by cosine similarity function.

5.4.5 Implemented recommenders

To thoroughly evaluate the proposed two KD strategies in obtaining compact and outstanding hierarchical recommenders, several state-of-the-art recommendation models are selected as baseline.

- **MF-BPR** [15]: This model presents a generic optimization criterion BPR-Opt for personalized ranking.
- **NeuMF** [2]: This is a state-of-the-art neural collaborative filtering model that used implicit user-item interaction.
NGCF \[67\]: This is a state-of-the-art GCNs-based model which integrates the user-item interaction into the embedding process through the graph convolution operation.

NIA-GCN \[69\]: This is a newly proposed GCNs-based model which explicitly exploits the user-user and item-item relationships through a pairwise neighborhood aggregator.

LR-GCCF \[145\]: This is a state-of-art GCNs-based model which utilizes a residual network structure and linear graph convolution operations to alleviate the over-smoothing problem.

LightGCN \[3\]: This is a newly proposed GCNs-based model that simplifies the design of graph convolution operation and only includes neighbor aggregation in convolution operations.

UltraGCN \[68\]: This is a newly proposed ultra-simplified GCNs-based model which skips infinite layers of message passing and yields efficient and effective recommendations.

UGCN: This is novel hierarchical GCNs-based recommender which can capture multi-resolution collaborative signals.

Moreover, the following recommenders under the KD paradigm are also implemented:

- **UGCN-t-128**: The sophisticated teacher recommender with the embedding dimension set to 128.
- **UGCN-sp-32, UGCN-sp-64**: The compact student recommenders trained by prediction-based distillation for data enhancement. The embedding dimension of these two models are set to 32 and 64 respectively.
- **UGCN-sk-32, UGCN-sk-64**: The compact student recommenders trained by prediction-based distillation for better hierarchies modeling. The embedding dimension of these two models are set to 32 and 64 respectively.
5.4.6 Other implementation details

The embedding size is fixed to 64 for baseline models, and the embedding parameters are
initialized with the Xavier method [138]. Other implementation details of the baseline
models, such as the default training strategies and hyperparameters settings are referred
from the corresponding papers.

5.5 Experiment Results

In this section, the following research questions are targeted:

- **Q1**: Are the proposed KD strategies effective for graph-based hierarchical recom-
menders? Does the student model teach by the sophisticated teacher achieve com-
petitive recommendation performances?

- **Q2**: Does the proposed KD strategy enable the compact student to produce more
satisfactory recommendations?

- **Q3**: How do different hyper-parameter settings (e.g., the number of top-ranked
items) affect the performances of student recommenders?

5.5.1 Overall accuracy comparison (QR1)

To evaluate the overall prediction accuracy, Recall@20 and NDCG@20 are tested in all
different models. The final results are presented in Table 5.2. The observations are listed
as follows:

- UGCN consistently achieves the highest recommendation accuracy among all base-
line models on all datasets, indicating its superiority and suitability as the teacher
model. In addition, compared to the baseline UGCN, the larger teacher model
UGCN-t-128 demonstrates better performances, which is in accordance with the
5.5. EXPERIMENT RESULTS

Table 5.1: Recall@20 and NDCG@20 results for the implemented recommenders.

<table>
<thead>
<tr>
<th>Compared Methods</th>
<th>BXBooks</th>
<th>AmazonMIs</th>
<th>Epinions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Recall</td>
<td>NDCG</td>
<td>Recall</td>
</tr>
<tr>
<td>BPR</td>
<td>0.0543</td>
<td>0.0321</td>
<td>0.1349</td>
</tr>
<tr>
<td>NeuMF</td>
<td>0.0627</td>
<td>0.0392</td>
<td>0.1494</td>
</tr>
<tr>
<td>NGCF</td>
<td>0.0914</td>
<td>0.0599</td>
<td>0.1862</td>
</tr>
<tr>
<td>NIA-GCN</td>
<td>0.0957</td>
<td>0.0621</td>
<td>0.1867</td>
</tr>
<tr>
<td>LightGCN</td>
<td>0.0988</td>
<td>0.0643</td>
<td>0.1900</td>
</tr>
<tr>
<td>UltraGCN</td>
<td>0.0987</td>
<td>0.0641</td>
<td>0.1951</td>
</tr>
<tr>
<td>UGCN</td>
<td>0.1007</td>
<td>0.0617</td>
<td>0.1964</td>
</tr>
<tr>
<td>UGCN-t-128</td>
<td>0.1144</td>
<td>0.0722</td>
<td>0.2011</td>
</tr>
<tr>
<td>UGCN-sp-32</td>
<td>0.0988</td>
<td>0.0637</td>
<td>0.1950</td>
</tr>
<tr>
<td>UGCN-sp-64</td>
<td>0.1018</td>
<td>0.0654</td>
<td>0.2038</td>
</tr>
<tr>
<td>UGCN-sk-32</td>
<td>0.0967</td>
<td>0.0629</td>
<td>0.1920</td>
</tr>
<tr>
<td>UGCN-sk-64</td>
<td>0.1019</td>
<td>0.0656</td>
<td>0.1981</td>
</tr>
</tbody>
</table>

discussion in Section 1 that a larger model with more learning parameters has a higher expression capacity and hence demonstrates better performances.

- In contrast to the baseline UGCN, both student models UGCN-sp-64 and UGCN-sk-64 achieve substantial improvements on recall@20 and ndcg@20 in all three datasets, confirming the effectiveness of two proposed prediction-based distillations. More promising results are exhibited in two compact student models UGCN-sp-32 and UGCN-sk-32. As shown in the Epinions dataset, these two student models (whose embedding size is 32) surprisingly outperform all complicated baseline models (whose embedding size is 64). Meanwhile, the performances on the other two datasets are also quite impressive.

The results and analysis above confirm the effectiveness of proposed distillation methods. In other words, by training with the top-ranked predictions from the sophisticated teacher, the compact student can demonstrate outstanding recommendation accuracy with a light model architecture. (Answer for RQ1)
CHAPTER 5. PREDICTION DISTILLATION IN HIERARCHICAL RECOMMENDERS

(a) R-Coverage Comparison Results.

(b) H-Coverage Comparison Results.

(c) Novelty Comparison Results.

Figure 5.1: R-Coverage@20, H-Coverage@20 and Novelty@20 comparison results on baseline and compact student models.

5.5.2 Recommendation satisfaction comparison (RQ2)

Besides the comparison in the general recommendation accuracy, three satisfaction evaluation metrics are employed in this section to comprehensively evaluate the recommendation performances of the student models. Moreover, two baseline modules UltraGCN [68], and UGCN are selected as baseline models in the following discussion owing to their competitive results displayed in Table 5.1.

Two types of student models with two sizes of embedding dimensions are tested on three datasets. The corresponding Coverage@20 and Novelty@20 are presented in Figure 5.1.
5.5. EXPERIMENT RESULTS

As shown in Figure 5.1, baseline UGCN achieves higher R-Coverage and H-Coverage than UltraGCN, which is consistent with the results in Chapter 4 that hierarchical recommender generates more satisfactory predictions. While compared to baseline UGCN, the performance of the two student models is totally different. Among them, UGCN-sp models perform poorly in Coverage@20 and Novelty@20 comparison, indicating that the direct incorporation of high-ranked teacher predictions into the model training might be sub-optimal. This can be explained by the fact that the top-ranked predictions are normally noisy and biased in certain popular items. Directly teaching the compact students with these biased data is easy to mislead the model training process, causing unsatisfactory recommendations. In contrast, the student model UGCN-sk demonstrates encouraging performances, achieving the highest Coverage and Novelty in two datasets while maintaining acceptable predictions in another dataset. Based on these results, it can be concluded that utilizing the high-ranked predictions to modify the user (item) hierarchical structure enables better and more precise group formation, resulting in better recommendation predictions.

In brief, two student models taught by two prediction-based distillation strategies can achieve outstanding prediction accuracy, but their satisfaction evaluation performances vary greatly. The prediction-based distillation for data enhancement biases the student model’s training with little over-popular items, causing unsatisfactory predictions. While the prediction-based distillation for better hierarchies extraction performs much better in recommendation satisfaction evaluation, enabling higher prediction accuracy and more satisfying predictions simultaneously. (Answer for RQ2)

5.5.3 Hyper parameters analysis (RQ3)

As the only type of knowledge distilled and transferred in the proposed strategies are the teacher’s top-ranked predictions, the quality and the quantities of these selected predictions could largely determine the performances of the trained student. In this section, several comparison experiments are conducted on the Epinions dataset to investigate the
corresponding impact of the selected predictions. Specifically, by changing the number of the top-ranked item from 1 to 7 with a step size of 2, the performances of UGCN-sp-64 and UGCN-sk-64 can exhibit valuable information. The comparison results and discussions are presented below. Moreover, although the following discussions only focus on UGCN-sp-64 and UGCN-sk-64 models with their corresponding results on the Epinions dataset, the analysis would also be applicable to other datasets as well as other models with different sizes.

As shown in Figure 5.2, it is apparent that the optimal numbers for selected top-ranked teacher predictions are 5 for both UGCN-sp-64 and UGCN-sk-64. Compared with the case where the number is set to 1 or 3, the best performances of UGCN-sp-64 and UGCN-sk-64 achieved in the number of 5 confirm that top-ranked predictions are often profitable and would benefit the student learning process. Meanwhile, keeping the number of distilled items would not be ideal since the performance of these two student models is largely decreased when the number of top-ranked items is set to 7. This can be interpreted as the student training or construction process will be significantly affected by a large portion of unconvinced and noisy distilled data, leading to performance decreases.

In brief, a fair number of distilled teacher predictions are always needed to be exploited and optimized on specific datasets. (Answer for RQ3)
5.6 Summary

In this chapter, two KD strategies, i.e., prediction-based distillation for data enhancement and prediction-based distillation for better hierarchies modeling, are proposed to transfer and distill knowledge from a sophisticated teacher to a compact student. Specifically, prediction-based distillation for data enhancement utilize top-ranked teacher predictions to directly guide the training of the student, while prediction-based distillation for better hierarchies modeling employ top-ranked teacher predictions to update the implicit hierarchical architecture of the student. Then, by training with the top-ranked predictions from the sophisticated teacher, the compact student can retain a light model architecture and demonstrate outstanding recommendation performances simultaneously.

The main contributions of the work in this chapter includes:

- Two prediction-based strategies are proposed to distill valuable knowledge from a sophisticated teacher to a compact student, enabling hierarchical recommenders to simultaneously achieve extraordinary recommendation performances and light model architecture.

- Comparative experiments are conducted on three datasets. Extensive results show that the proposed KD strategy can help the student model achieve outstanding recommendation accuracy and obtain more satisfactory predictions with higher coverage and novelty.
Chapter 6

Conclusion and Future Directions
6.1 Thesis Summary

This thesis has investigated the topic of constructing personalized recommender systems. The major challenges or problems concerned are (1) complicated relationships among user-item relationships; (2) insufficient exploration of available data; (3) huge computation expenses, and long inference delays of sophisticated recommendation architecture. To tackle these challenges or problems, three corresponding algorithms: (1) utilizing graph-based architecture to explore user (item) implicit hierarchies; (2) capturing valuable personalized information from pairwise differences; (3) employing knowledge distillation to obtain compact and extraordinary recommendation models, are proposed and presented in this thesis. The detailed summations are presented below:

Chapter 3 focuses on proposing a better data modeling strategy. The limitations of current data modeling and model training methods are first presented. Then, the idea of capturing personalized information from pairwise preference differences is discussed and extended to explicit interactions. Based on this, the CPR, a formula that can capture pairwise preference differences, is proposed. After that, with proper derivation and transformation process, a novel point-wise loss function, i.e., DifE, is proposed to capture pairwise information and enable recommender systems to produce better recommendations. Four state-of-the-art point-wise recommendation models are selected and trained with the proposed loss function. Comprehensive experiments on several datasets demonstrate that these DifE-optimized models are able to capture personalized information preserved in pairwise differences and can generate better recommendations. The superiority of these DifE-optimized models demonstrates the excellence and generality of the proposed data modeling and model learning strategy.

Chapter 4 targets the relationship modeling problem in recommender systems. The problem of existing recommenders often neglecting user (item) inherent hierarchical relationships is first introduced, together with the importance and benefits of exploring user (item) hierarchies in constructing superior recommenders. Then, an implicit user (item) hierarchical exploration model, i.e., UGCN, is proposed, which utilizes a u-shaped hierarchical
CHAPTER 6. CONCLUSION AND FUTURE DIRECTIONS

graph convolution model and is able to capture user-item collaboration signals in different resolution scales. Precisely, the proposed model uses a U-shaped architecture to gradually compress the original user-item interaction graph into smaller graphs by merging similar nodes together. Then, simplified graph convolutional operations can easily extract diverse group-level collaborative signals from compressed graphs. After that, those learned informative signals would be projected back to the final user (item) embeddings adaptively. Extensive and comprehensive experiments on three public datasets demonstrate that the UGCN recommender achieves significant overall prediction improvements over state-of-the-art models with better recommendations simultaneously.

Chapter 5 explore the solution and optimization for huge computation cost problem. The common trend of continuously increasing the model complexity to construct recent recommendation architectures is first presented, together with the corresponding problems and restrictions caused by endlessly increasing model size. Two prediction-based distillation strategies suitable for hierarchical UGCN recommender are proposed to retain the light architecture and maintain outstanding recommendation performances simultaneously. Firstly, a prediction-based distillation method that can enhance and enrich the recommendation dataset is proposed. By directly regarding the top-ranked items generated by the sophisticated teacher as positive instances, the enhanced training data enable the compact student to capture more profitable information and demonstrate outstanding performance. Meanwhile, considering the essential role of modeling user (item) hierarchies, another prediction-based distillation method that allows a compact student model to obtain a better hierarchical architecture is also proposed. Concretely, by employing top-ranked teacher predictions to modify the student’s hierarchy extraction process, more precise neighbors can be identified in the compact student model, resulting in more accurate multi-resolution preferences modeling and better recommendation predictions. Extensive and comprehensive experiments on three public datasets demonstrate that the proposed distillation strategies enable hierarchical recommenders to achieve extraordinary recommendation performances and light model architecture simultaneously.
6.2 Future Extensions and Directions

In this section, the possible extensions and future directions of the proposed methods in this thesis are discussed and presented.

(1) Incorporate DifE loss with more recommendation scenarios

As presented in Chapter 3, the proposed DifE loss function only integrates with state-of-the-art collaborative filtering-based recommenders. Content-based recommenders, which utilize both auxiliary side information and historical interactions to generate recommendation predictions, also play an essential role in many recommendation scenarios. Hence, incorporating effective data modeling and model training methods with the state-of-the-art content-based recommender seems promising. The straightforward way for this incorporation can be directly replacing the original loss in content-based recommender with the proposed DifE loss. Moreover, considering that DifE loss only works for historical interactions and content-based recommenders mainly rely on side information to model user-item relationships, a more generalized incorporation could be modifying the original DifE calculation and enabling pairwise differences modeling from side information. With better data modeling and training methods, content-based recommenders will likely achieve better and more personalized predictions. In addition, as a large number of real-world recommender systems merely collect user (item) implicit behaviors (such as clicks), an implicit feedback-based DifE loss is also researching-worthy. More concretely, incorporating similar neighbors’ interactions to adjust the probability of the ambiguous implicit behavior and extracting personalized preference from pairwise adjusted implicit interactions could be a promising way to improve the prediction accuracy of those implicit interaction-based recommenders.

(2) Explore better hierarchical architecture

For GCNs-based hierarchical UGCN recommender, to the best of the author’s knowledge, it is the first GCNs-based model which can exploit and modeling user (item) implicit hierarchies in recommendation scenarios. In other word, the quality of the explored and
CHAPTER 6. CONCLUSION AND FUTURE DIRECTIONS

constructed user (item) hierarchical structures would largely impact the performances of
the proposed UGCN recommender. As presented in Chapter 4, the current hierarchy
exploration method relies on 1-nearest neighbor merging to form user (item) groups. Al-
though the proposed methods are already demonstrating outstanding performances, better
group formation and hierarchy exploration methods may be still needed to further improve
the proposed UGCN model. Meanwhile, the proposed UGCN merely relies on historical
interactions. In fact, different types of data resources, such as user (item) attributes,
descriptive contents, etc., are also quite valuable and suitable for UGCN recommender.

(3) Extend proposed distillation schemes on more recommendation architec-
tures

The proposed distillation strategies are proved to be effective in transferring and distilling
knowledge among the state-of-the-art hierarchical recommenders. While in Chapter 5,
there is little discussion on applying the proposed distillation strategies on other state-
of-the-art recommendation approaches. In fact, the idea of distilling the prediction is
applicable to many recommendation architectures. For example, the distillation strategy
proposed in Section 5.3.1 is independent of the selection of the teacher or student model.
Hence, it can be naturally incorporated with other state-of-the-art recommendation archi-
tectures. Meanwhile, the core point behind the distillation strategy presented in Section
5.3.2 is to distill accurate similarity among users (items). Therefore, it would be reason-
able to implement or modify the proposed distillation strategy to any similarity-based
recommenders. In conclusion, the primary direction for further work could be extending
the proposed distillation strategies to more recommendation approaches.
References


[96] Y. Deldjoo, M. G. Constantin, H. Eghbal-Zadeh, B. Ionescu, M. Schedl, and P. Cremonesi, “Audio-visual encoding of multimedia content for enhancing movie recom-


