UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Enhanced Recommender Systems with Deep Neural Networks

by Ruiping Yin

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of A./Professor Guangquan Zhang and Distinguished Professor Jie Lu

University of Technology Sydney
Faculty of Engineering and Information Technology

September, 2022

Certificate of Original Authorship

I, Ruiping Yin, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of the requirements for a degree at any other academic institution except as fully acknowledged within the text. This thesis is the result of a Collaborative Doctoral Research Degree program with Beijing Institute of Technology.

This research is supported by the Australian Government Research Training Program.

Production Note:
Signature removed prior to publication. Ruiping Yin

20/9/2022 September 2022

Acknowledgements

First and foremost I would like to express my sincere thanks to both my principal supervisor A./Professor Guangquan Zhang and co-supervisor Distinguished Professor Jie Lu. Without their patience and encouragement, I would not have been able to finish my PhD journey. Their strict academic attitude and respectful personality has benefited my PhD study and will be a great treasure throughout my life. I have learnt so much from them. During my doctoral research, they gave me countless guidance and help which inspired me in many aspects including research methodology, experiments, writting skills. and even the sentence structure and mathematical formulas and greatly improved my thesis quality.

Then I would like to express my gratitude to every member of the Decision Systems & e-Service Intelligence Lab (DeSI). Thank you for your care and help. I have benefited a lot from the time I spent with you. I would like to especially thank Qian Zhang for the discussion on my research.

Next I am grateful to the the School of Software, Faculty of Engineering and Information Technology, University of Technology, Sydney for their support. I am also grateful for the financial support I received from the Vice-Chancellor's Postgraduate Conference Fund.

Finally, I would like to express my gratitude to my family who have been supporting and encouring me. I would like to thank my father, my mother and my

wife for their care and support, and to all of my friends who have helped me and cared about me. Love you guys.

Abstract

Recommender system is an intelligent decision making system that adopts machine learning technology to recommend relevant contents to users based on the analysis of users' interests and preferences. It can help users find appropriate contents within a reasonable time and has been proved to be an effective means to deal with the problem of information overload. At present, although the recommender system has been widely used in various social fields, there are still many problems in the existing recommender system. For example, data sparsity, cold start, long tail items are difficult to be recommended, and graph structure data cannot be effectively processed, resulting in low recommendation performance and poor user experience, which restricts the development of personalized recommender system. In recent years, with the rapid development of the theory and technology of deep learning, the study on personalized recommendation on deep neural networks has been paid more and more attention by the industry and academia. How to use the principles and techniques of deep learning to alleviate and overcome the problems in the existing personalized recommender system, so as to improve the performance of the recommender system, is a topic worthy of research.

The main work of this thesis for deep learning-based personalized recommendation methods is as follows:

To model the conception of fashion and visual factors on the fashion recommendation task, we propose a cross-domain recommendation method based on visual collocation knowledge transfer. First, we extract visual collocation knowledge of fashion items from images on a popular fashion website and even street photography, and incorporate the learnt knowledge to the recommender system through transfer learning. By collecting cross-domain information and updating visual collocation knowledge, the accuracy of clothing recommendation is improved.

To overcome the difficulty of accurately extracting latent features of new users and non-popular products, we propose a recommendation method based on deep graph convolutional neural network. Different with the conventional methods which consider the low-order similarity only, we learn the representation of users and items from the high-order similarity between users and items. We treat the recommendation task as an edge prediction problem on a bipartite graph. It inherits the advantages of graph convolutional neural network to quickly combine local information on the graph, so that we can obtain the node embedding which consists of the node's information, neighbors' information and local structure information. At the same time, for the over-smoothing problem caused by the multi-layer graph convolutional neural network, we propose an information propagation method based on the attention mechanism, which can effectively alleviate the over-smoothing problem when the graph convolutional neural network is too deep.

To solve the problem that the user's preference is affected by the environment and changes with time, we propose a recommendation method based on the user's long-term and short-term preference. In one session, the products browsed by the user have a certain continuity. This method models the user's current

shopping intention through the items that the user has browsed in the current session. At the same time, the method also combines the user's long-term stable preferences contained in the user's historical records to provide users with in-time recommendations. The method can quickly adapt to the changes of the user's current interests caused by changes of the context and improve users' stickiness to shopping websites.

To solve the problem of data sparsity, we propose a recommendation method based on generative adversarial strategy. The algorithm generates a user's latent feature vector by training a generator network with a denoising autoencoder, which generates recommendations for the user accordingly, while training a discriminant network to distinguish the recommendation prediction generated by the generating network from the user's real transaction records. The adversarial training between the discriminating network and the generating network helps to push recommendation predictions closer to the real transaction records. Through continuous iterative adversarial training between generation network and discriminant network, the two networks are mutually promoted. Therefore, the final recommendation is improved.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Recommendation technology is a system that gives consumers recommendations for consumption. Given the explosive growth of information available on the web, consumers may face countless popular products, movies or meals in their daily lives. Therefore, personalized recommendation is the basic strategy to give consumers a better user experience. Today's recommender systems play a vital role in various information access systems to facilitate the user's decision-making process. As such, recommender systems are widespread in many areas such as e-commerce or media websites.

In recent years, deep learning has achieved remarkable results in many aspects such as natural language processing, image recognition, and scholars have continued to try other areas to solve complex problems that are difficult to solve with traditional methods. Deep learning has also become a popular branch in the entire field of machine learning algorithms. And major domestic and foreign companies have

increased their research and investment in deep learning, and many companies have achieved fruitful results in the research and development of artificial intelligence commercial products.

The theory of Deep Learning was proposed by Hinton and other masters in 2006. It is essentially derived from the development of artificial neural networks. Through the study and learning of shallow networks, features are extracted from them, and then integrated, and finally formed deeper features can be used in the field of classification prediction. In the beginning, deep learning mainly used **D**eep **B**elief **N**etworks (DBN) for unsupervised greedy training layer by layer, and later developed to use more complex multi-layer autoencoders for training and feature extraction. The essence of deep learning is still a neural network, but the difference is that it contains many hidden layers, and these hidden layers are constructed, and then in training, a large amount of data is referenced, so that the model can learn more multi-dimensional features of the data , In order to improve the classification accuracy and improve the accuracy of prediction. Deep learning not only has excellent automatic feature extraction capabilities, but also the ability to automatically learn multi-level and multi-dimensional abstract feature representations, as well as the ability to learn heterogeneous or cross-domain content information. On the one hand, deep learning strengthens the learning of features. On the other hand, it also increases the number of layers of the neural network. The features are converted from layer to layer, thereby converting the feature representation of the sample in the original space into a new spatial structure. Finally, classification or prediction becomes easier. At the same time, using a large amount of data for training can better characterize the data-rich internal information.

In view of the fact that deep learning technology has achieved good application results in many fields, in the field of recommender systems, experts are gradually exploring deep learning technology. The introduction of deep learning technology into the field of recommender systems makes the recommender system more intelligent and can provide users with recommendation services. In 2016, ACM RecSys organized the first deep learning and recommender system conference RecSys 2016, and emphasized that the recommender system will be better developed with the help of deep learning. Subsequently, RecSys 2018 and RecSys 2019 were held in Vancouver, Canada and Copenhagen, Denmark, and RecSys 2020 will also be held in Rio de Janeiro, Brazil. In recent years, deep learning technology has been studied and tried as much as possible in the recommender system, and many positive results have been obtained. In the field of recommender systems, deep learning will gradually develop into a pivotal professional field.

In recent years, the research of personalized recommendation based on deep learning has attracted more and more attention from many scholars. Deep learning can realize the automatic extraction of features, which has changed the way that traditional machine learning methods require manual feature extraction. At the same time, this feature extraction method of deep learning can be integrated into the established model process, thereby reducing feature extraction. At the same time of difficulty, it also reduces the incompleteness of feature acquisition when relying on human design features (Singhal et al., 2017). For example, 80% of the online viewing of movies on the Netflix website comes from recommendations(Gomez-Uribe and Hunt, 2015), and 60% of video clicks on the YouTube website come from recommendations on the homepage of the website(Davidson et al., 2010). Many companies use deep learning principles and techniques to improve the quality of

recommendations(Chen et al., 2019). Different from the previous recommendation technology, the existing recommendation technology using the deep neural network structure has achieved very good recommendation performance, and its application range is becoming more and more extensive. Zeynep Batmaz, Ali Yurekli and others(Batmaz et al., 2019) gave the main distribution of the current application fields of deep learning recommender systems, such as online movies, books, news, music and other e-commerce fields and social network industries.

At present, although personalized recommendation based on deep learning has achieved certain results from both academic and industrial perspectives, it does not meet the further and deeper social needs. In existing recommender systems, there is still a widespread recommendation process data sparseness, cold start, long tail items are difficult to be recommended, and graph structure data is difficult to process(Wu et al., 2020), even in the recommender system supported by some current deep learning methods such as **C**onvolutional **N**eural **N**etwork (CNN) and **R**ecurrent **N**eural **N**etwork (RNN). Therefore, how to use the latest theories and techniques of deep learning, such as the neural network **G**raph **N**eural **N**etwork (GNN), to solve or alleviate these problems in the existing recommender systems to improve the performance of the recommender systems, has become an important and urgent problem to be solved.

1.2 Research Questions and Objectives

This research aims to enhance recommender systems by developing a set of recommendation approaches with deep neural networks. To summarize, the following research questions will be answered by this research:

QUESTION 1. How to take advantage of rich multimedia information, for example, the images of products in online stores?

QUESTION 2. How to make use of high-order similarity information in the user-item interaction graph?

QUESTION 3. How to incoporate the long- and short-term user preferences in recommender systems?

QUESTION 4. How to effectively optimize the recommendation model when the data is too sparse?

To answer these research questions, this research aims to achieve the following objectives:

OBJECTIVE 1. To propose a cross-domain recommendation approach based on visual collocation knowledge transfer to model the influence of fashion trends and visual factors on the recommendation of fashion items.

This objective corresponds to research question 1. Recommender systems often suffer from data sparsity and cold start problem. Ultilizing side information shows promising performance increasing in some situations. With the development of deep learning techniques, we can extract rich information from various of multi-media data sources, such as natural language, images and videos. However, with such a huge dataset, we cannot label them to train our recommendation model. Thus, how to ultilize these unlabeled data is curitical today. We proposed a promising domain adaptation method to transfer fashion collocation knowledge to the recommendation senarios. Based on the domain adapation method, a recommender system is built to ultilize clothing collocation knowledge to improve the recommendation accuracy.

OBJECTIVE 2. To develop an Graph Convolutional Neural Network-based recommendation approach which is able to leverage high-order neighbour information in user-item interaction history.

This objective corresponds to research question 2. Following the definition in objective 2, we survey and analysis the tranditional factorization based method. We found that in these method, only low-level neighbour information is used. Therefore, we built an GNN-based recommendation method, through information broadcast and aggregation, we are able to get more information in the recommendation stage.

OBJECTIVE 3. To develop a long- and short-term based recommendation approach which considers both long and short user preferences.

This objective corresponds to research question 3. We aim to build a model that integrates long- and short-term user interests. The model can learn the user's long-term interests and current intentions at the same time. When generating the recommendation list, according to the user's current intentions, the user's long-term interests are combined with context information to generate more accurate recommendations.

OBJECTIVE 4. To develop a new optimization method which alleviates the problem of insufficient user samples.

This objective corresponds to research question 4. For cold-start users, recommender systems cannot obtain enough information to generate accurate recommendations for them. In other words, in the optimization process, it is difficult for users with insufficient samples to have sufficient influence on the optimization process. Therefore, we try to propose a new optimization method to increase the influence of cold-start users on the optimization process, and improve the recommendation effect of the entire system ultimately.

1.3 Research Significance

1.3.1 Theoretical Significance

Theoretically, the research develops a set of recommendation methods and solves the two following issues in recommender systems:

- **Feature representation.** This research solves the visual knowledge modeling problem to take visual complementary information into consideration. We first use a triple network to supervised learning domain knowledge of clothing matching from a clothing matching dataset. Then, the learned collocation knowledge is applied to the target domain that needs to be recommended through the domain adaptation strategy to alleviate the distribution difference between the source domain and the target domain.

- **Model construction.** In this research, two deep learning-based recommendation models were developed to solve the problem of insufficient utilization of collaborative information in recommender systems. The first is to use a graph convolutional neural network to increase the receptive field of the model in the recommendation process, so that the recommender system can use more information to make decisions. The second is to integrate the user's long and short-term interests, so that the recommender system can focus on the current context to generate recommendations.

- **Optimization algorithm.** This research provides a framework for generative adversarial optimization method. This complements and improves the whole methodology of deep learning-based recommendation techniques.

1.3.2 Practical Significance

In practical terms, this research provides guidelines on how to improve the performance of recommendation by addressing the following two problems:

- **Recommendation accuracy.** Data uncertainty and scarcity is very common in real-world recommender system applications. This research provides various ways to improve the accuracy of recommendation and provides users with better decision-making support. In particular, the proposed system and the new users who have recently entered the system can benefit from this research.

- **Recommendation diversity.** There are many online systems that sell different types of goods. The historical records of users are across different categories. This research provides a way to satisfy the demand for recommendation across domains. Since recommendations can be made in different categories of goods, the potential diversity of recommendations is improved.

1.4 Thesis Structure

The logical structure of this thesis and the relationship between the chapters are shown in Figure 1.1. The main contents of each chapter are summarized as follows:

Chapter 2 presents a systematic literature review related to this research. In this chapter, first we give a general introduction of recommendation techniques, followed by a classification of recommendation techniques. Then, a systematic review of deep learning technology is presented together with recent research

development of this area. Finally, deep learning-based recommender systems are reviewed. Related methods and techniques are classified into three groups according to the different deep learning techniques. After this comprehensive review, the current research gaps are discussed and summarized.

Chapter 3 proposes a novel method for the fashion recommendation task with learning compatibility knowledge in visual aspect. In this chapter, A triplet network is used to learn compatibility knowledge from an external dataset. Domain adaptation strategy is used to alleviate the distribution gap between source domain and target domain. Empirical experiments are conducted on serveral real-world datasets. The results show that the proposed approach improves the performance of recommender systems.

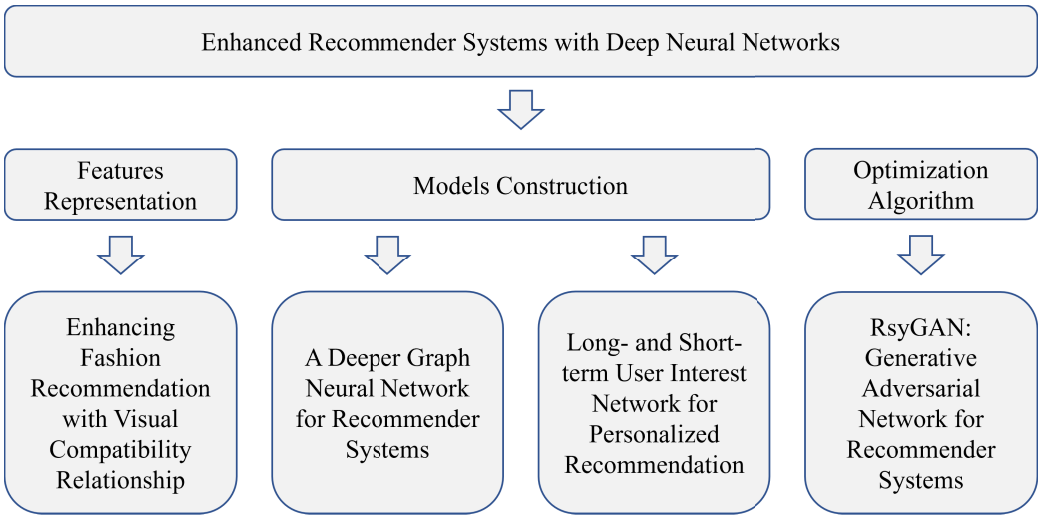


Figure 1.1 Thesis structure

Chapter 4 develops a general framework, named **G**raph neural network-based **C**ollaborative **F**iltering (GCF), and an information propagation-based graph neural network. We conducted experiments on several real-world datasets to reveal the relationship between the number of iterations of information propagation and the

recommendation performance. The experimental results show that the proposed method outperforms all the other models.

Chapter 5 develops a recommendation method, named **L**ong- and **S**hort-term user interest network for personalized **rec**ommendation (LSREC), based on long- and short-term user interest. Specifically, the proposed method is composed of three components: **L**ong-term **U**ser **I**nterest **N**etwork (L-UIN) for learning long-term user interest from historical behaviors and user profiles, **S**hort-term **U**ser **I**nterest **N**etwork (S-UIN) for modeling short-term uesr interest from sesssion sequence, and combination for recommendation generating. We particularly devise a graph convolutional neural network to enrich the user's local sturcture information by considering the high order neighbors in L-UIN. We have further developed an attention based RNN to learn short-term user interest in S-UIN. The experimental results on two datasets have demonstrated the superiority of LSREC over methods considering short-term user interest and long-term interest.

Chapter 6 summarizes the contributions of this research and discusses research issues for further study.

1.5 Publications Related to This Thesis

Following is a list of the refereed international journal and conference papers during my PhD research that have been published:

Published:

1. **R. Yin**, K. Li, G. Zhang and J. Lu. "Enhancing Fashion Recommendation with Visual Compatibility Relationship." The World Wide Web Conference on - WWW '19, ACM Press, 2019.

2. **R. Yin**, K. Li, G. Zhang and J. Lu. “A Deeper Graph Neural Network for Recommender Systems.” Knowledge-Based Systems, vol. 185, no. 105020, 2019, p. 105020.

3. **R. Yin**, K. Li, G. Zhang and J. Lu. “RsyGAN: Generative Adversarial Network for Recommender Systems.” 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, 2019.

4. Y. Zhang, **R. Yin***, Z.Yang. "Data Poisoning Attacks to Session-Based Recommender Systems." 2022 International Conference on Communication and Network Security (ICCNS). ACM Press, 2022.

5. **R. Yin**, K. Li, G. Zhang and J. Lu. “Detecting Overlapping Protein Complexes in Dynamic Protein-Protein Interaction Networks by Developing a Fuzzy Clustering Algorithm.” 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE, 2017.

Chapter 2

Literature Review

This chapter presents a literature review of relevant studies in connection with this research. In recent years, recommender systems have become one of the most important research areas in academia and industry. For different recommendation scenarios, the design of recommendation models and algorithms are also different. This research focuses on how to design personalized recommendation methods with deep learning techniques to improve the performance of recommender systems in terms of accuracy. In this chapter, we review the research related to the three aspects of the above issue: recommendation techniques, deep learning techniques and deep learning-based recommender systems. In Section 2.1, we briefly introduces the technical development in the field of recommender systems. This is followed by a review of deep learning techniques related to recommender systems in Section 2.2. Finally, in Section 2.3, we reviews different methods that are related to this research.

2.1 Recommendation Techniques

In recommender systems, the user's preferences are usually implicitly reflected in the user's behavioral records, demographic information, and the context of the current interaction with the system. The recommended items can be commodities, news, movies, short videos, friend relationships, etc.

The input of a typical recommender system includes the user's behavior records (e.g., clicking, purchasing, etc.), the side information of the user and the recommended item in the system, the output of the recommender system is aimed at the target user, producing a personalized recommendation list containing the items that the user is most likely to interact with next. The essence of the goal is to calculate the user's preference for all items in the system for the target user, and sort all items in descending order of preference. Finally, the K head items in this ordered list are used as the user's preference. Personalized recommendation list, hence also called TOP-K recommendation.

User attribute information includes user ID, age, gender, income level, etc.; item attribute information includes item ID, description, picture, category information, price, and the like. In real-world recommendation scenarios, there is little or no overlap between user attributes and product attributes. This creates the first challenge in recommender systems: the semantic gap between users and items, i.e. users and items are two different types of entities with different types of attributes.

A typical recommender system can be represented by Figure 2.1.

Currently popular recommendation algorithms can be basically divided into two categories: recommendation algorithms based on **C**ollaborative **F**iltering (CF) and recommendation algorithms based on general features.

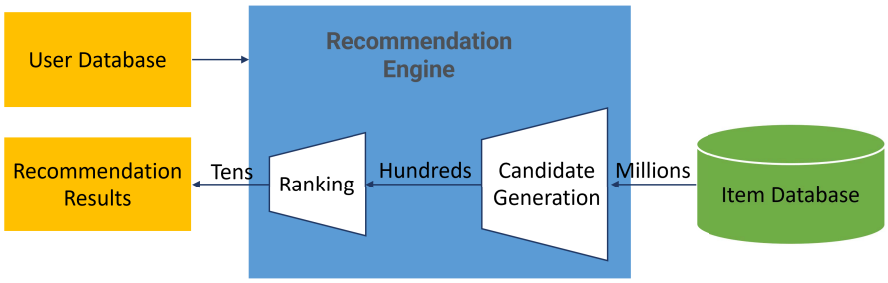


Figure 2.1 Framework of recommender systems(Covington et al., 2016)

2.1.1 Collaborative filtering-based recommendation algorithm

The use of the term Collaborative Filtering can be traced back to 1992. Goldberg first proposed a personalized mail recommender system based on collaborative filtering, Tapestry(Goldberg et al., 1992). The system requires users to mark the mails they have read, and then other users select the mails they need based on these marks. Tapestry requires users to enter some query keywords, such as sender, send date, email subject, etc., before the system can give corresponding recommendations. Therefore, in a sense, Tapestry is a semi-automated recommender system.

1994 was a milestone year for the recommender systems. The GroupLens project team published its famous research results (Resnick et al., 1994), in which the first collaborative filtering algorithm applied to news recommendation was proposed. The algorithm is used to filter out the news that users may be interested in from a large amount of news content. The basic idea is to assume that the user's interest remains stable for a certain period of time, and predict the news content that the user will like in the future based on the user's previous interest. In the GroupLens system, users rate the news that they have browsed. The scoring mechanism is similar to the current most popular 5-star rating system. GroupLens calculates the similarity between users based on the user's rating of news content, and selects

a group with the target Users with the most similar user interests can predict the target user's interest in new content through collective voting. GroupLens laid the foundation for the development of collaborative filtering technology. Since then, the recommender system has received extensive attention from industry and academia, and a large number of well-known domestic and foreign scholars, experts and industrial researchers have participated in related research on the recommender system. In the mid-1990s, recommender systems have become an independent research field.

A basic assumption of the collaborative filtering-based recommendation algorithm is that users with common interests will tend to choose similar items. Therefore, the most important thing for a collaborative filtering-based recommender system is to find people who have common interests and preferences with the target user. According to the development process of recommendation algorithms and the different technologies used, recommendation algorithms based on collaborative filtering can be divided into three categories: memory-based collaborative filtering recommendation algorithms, model-based collaborative filtering recommendation algorithms, and deep learning-based collaborative filtering recommendation algorithms algorithm.

(1) Memory-based collaborative filtering algorithm

The memory-based collaborative filtering algorithm is the earliest recommendation algorithm based on collaborative filtering, which uses heuristic algorithms to calculate the similarity between users or items. Memory-based collaborative filtering algorithms are divided into two categories: user-based collaborative filtering and item-based collaborative filtering(Deshpande and Karypis, 2004). The core algorithm of the memory-based collaborative filtering algorithm is the Nearest

Neighbor Algorithm. The target user's preferences for different items are calculated based on the degree of preference of his neighbors for these items. This algorithm is widely accepted because of its simplicity, effectiveness and interpretability in a certain sense. A typical memory-based collaborative filtering algorithm includes the following steps(Su and Khoshgoftaar, 2009):

a. Similarity calculation

Before predicting the score, it is necessary to calculate the weight of each user/item based on the distance to the target user/item. The significance of calculating this weight is to select the K nearest neighbors of the target user/commodity in the entire data set. This step is very important for the final recommendation accuracy. Therefore, researchers have proposed various measurement methods for how to calculate the distance between user-user or item-item.

Correlation-based similarity was first proposed (Billsus et al., 1998; Lang, 1995). Denote the user's rating matrix for items as $X \in R^{M \times N}$, which means that M users and N items are in this matrix. The items i and j can be replaced by the corresponding columns in the matrix, that is, the vector formed by each user's rating. The similarity between two vectors can be calculated using cosine similarity. The cosine similarity between items i and j can be calculated using the following formula:

$$W_{(i,j)}^{cos} = \frac{\sum_{u \in U_i \cap U_j} r_{u,i} \times r_{u,j}}{\sqrt{\sum_{u \in U_i \cap U_j} r_{u,i}^2} \times \sqrt{\sum_{u \in U_i \cap U_j} r_{u,j}^2}} \tag{2.1}$$

Among them, U_i is the set of all users who have rated the item i, and U_j is the set of all users who have rated the item j. $U_i \cap U_j$ represents the set of users who

have jointly rated the items i and j. $r_{u,i}$ and $r_{u,j}$ are the ratings of user u on items i and j respectively.

Cosine similarity is generally used in item-based collaborative filtering algorithms, and is rarely used in user-based collaborative filtering algorithms, because it does not take into account that the mean and variance of scores are different between users. The Pearson Correlation Coefficient (PCC) solves this problem well. This metric can be used in both item-based collaborative filtering algorithms and user-based collaborative filtering algorithms. In the user-based collaborative filtering algorithm (Sarwar et al., 2001), the Pearson correlation distance between users u and v can be expressed as:

$$W_{u,v}^{PCC} = \frac{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r_u}) \times (r_{v,i} - \bar{r_v})}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r_u})^2} \times \sqrt{\sum_{i \in I_u \cap I_v} (r_{v,i} - \bar{r_v})^2}} \tag{2.2}$$

Among them, I_u is the set of all items rated by user u, and I_v is the set of all items rated by user v. $I_u \cap I_v$ is a collection of items rated by users u and v at the same time. $\bar{r_u}$ and $\bar{r_v}$ are the mean values of all ratings on $I_u \cap I_v$ for user u and user v respectively.

For item-based collaborative filtering algorithm (Resnick et al., 1994), the Pearson correlation distance between items i and j can be expressed as:

$$W_{i,j}^{PCC} = \frac{\sum_{u \in U_i \cap U_j} (r_{u,i} - \bar{r_i}) \times (r_{u,j} - \bar{r_j})}{\sqrt{\sum_{u \in U_i \cap U_j} (r_{u,i} - \bar{r_u})^2} \times \sqrt{\sum_{u \in U_i \cap U_j} (r_{u,j} - \bar{r_j})^2}} \tag{2.3}$$

Among them, U_i is the set of all users who have rated the item i, and U_j is the set of all users who have rated the item j. $U_i \cap U_j$ represents the collection of all users who have reviewed the item i and the item j. $\bar{r_i}$ and $\bar{r_j}$ are the average values of the scoring of items i and j by users in $U_i \cap U_j$ respectively.

b. Score calculation

After the calculation of the weight matrix is completed, the target user u's rating $r_{u,i}$ for the item i can be predicted. The main basis for prediction is the neighbors of the target user/item. In the item-based collaborative filtering algorithm, the calculation method can be expressed as the following form (Sarwar et al., 2001):

$$r_{u,i} = \frac{\sum_{j \in I_u} r_{u,j} \times W_{i,j}}{\sum_{j \in I_u} |W_{i,j}|} \tag{2.4}$$

Among them, I_u is the set of all rated items of user u except item i, $W_{i,j}$ is the target item i obtained in the previous step and another known rating The distance between the items j.

In the user-based collaborative filtering algorithm, the target user's rating of the item calculated according to the Pearson correlation coefficient can be expressed as (Resnick et al., 1994):

$$r_{u,i} = \bar{r_u} + \frac{\sum_{v \in U_i} (r_{v,i} - \bar{r_v}) \times W_{u,v}}{\sum_{v \in U_i} |W_{u,v}|} \tag{2.5}$$

Among them, U_i is the set of all users who have rated the item i, and $W_{u,v}$ is between the target user u and another user v who has rated the item i The similarity.

c. Generate recommendations

Whether it is item-based recommendation or user-based recommendation, the k most similar users/items are selected by the weights in the formula 2.1. Then, these k nearest neighbors are used to calculate each user's rating for each unrated item. Then the predicted scores are sorted in descending order, and the K items with the highest scores are recommended to users as a recommendation list. This recommendation is also called TOP-K recommendation.

Although the memory-based collaborative filtering algorithm has been widely used because of its ease of use and relative effectiveness, the technology still has some weaknesses that cannot be ignored(Adomavicius and Tuzhilin, 2005). First of all, the technology cannot handle the cold start problem. When a new user/item is added to the system, there is no relevant score in the system to predict the user/item's score. Second, if an item is not new, but is a non-popular item that is rarely visited, then there will not be enough ratings from users to make predictions. The memory-based collaborative filtering system will always avoid recommending unpopular items to users. Therefore, the range of recommended items will be limited. Third, memory-based collaborative filtering cannot provide real-time recommendations. When the user-item matrix is very large, the heuristic calculation process takes a lot of time to generate recommendations. In item-based collaborative filtering, pre-calculating and storing the weight matrix can alleviate this problem to a certain extent, but the scale of the entire system still cannot meet the needs of the current real scene.(Deshpande and Karypis, 2004).

(2) Model-based collaborative filtering recommendation algorithm

Model-based collaborative filtering is different from the memory-based heuristic methods discussed earlier, but uses machine learning or data mining methods to build a model to predict users' ratings of items. This type of method was originally established to solve some problems in memory-based collaborative filtering, but it has been widely used to solve various problems in some specific fields. In addition to the user-item rating matrix, other additional information is also taken into account, such as location information, tags and comments, etc.(Shi et al., 2014). Model-based collaborative filtering technology is a good choice to combine the scoring matrix with additional information. In the following content, three types of

recommender systems based on collaborative filtering are introduced: recommender systems based on matrix factorization, recommender systems based on graphs, and recommender systems based on context.

a. recommender system based on matrix factorization

The first appearance of matrix factorization is the application of probabilistic Latent Sematic Analysis (pLSA) to the recommender system (Hofmann, 2004). In 2008, in the recommendation algorithm competition (Koren, 2008) held by Netflix, matrix factorization began to gain people's attention with its excellent performance. To this day, it has been one of the most popular models in the recommender system field. This model is also called Singular Value Decomposition (SVD)(Koren et al., 2009). This method maps users and items to the same Latent Factor space so that they can be compared with each other. After that, the probabilistic explanation of the matrix factorization method was also given. (Salakhutdinov and Mnih, 2009). The popularity of matrix factorization technology benefits from its three advantages. First of all, through matrix decomposition, the dimension of the user-item matrix can be significantly reduced, which greatly reduces the scale of the entire system, which is more suitable for the current increasingly large-scale data. Secondly, the process of matrix decomposition produces a dense hidden factor matrix, so the problem of data sparseness is alleviated to a certain extent.(Luo et al., 2016). Compared with the memory-based collaborative filtering algorithm, the recommendation algorithm based on matrix factorization enables users with only a small number of ratings to obtain relatively more accurate recommendation results. Finally, matrix factorization is very suitable for integrating additional information into the model(Liu et al., 2015). This can help the system understand

the user's preferences in more detail, thereby improving the effectiveness of the recommendation.

McAuley and Leskovec combined the dense features obtained after the scoring matrix decomposition with the topic features in the review data, and learned user preferences through the comments generated by the user. They proposed a method combining matrix decomposition and topic models, and obtained A better effect(McAuley and Leskovec, 2013b). With the help of sentiment analysis technology, Diao et al. proposed an unsupervised method to learn users' interests and opinions on movie ratings(Diao et al., 2014). Qian et al. explained the user's behavior from a probabilistic point of view by mining the attributes and tags of items(Qian et al., 2014). McAuley et al. modeled people's visual perception of pictures into the recommender system, and applied it to the clothing recommendation(He and McAuley, 2016a; McAuley et al., 2015). Yang et al. used the trust relationship between users as the weight of the influence of the user's opinion on other users and applied it to the matrix factorization technology(Yang et al., 2017). All these works show that matrix factorization technology is a basic stepping stone in the field of recommender systems and has great advantages in combining additional information.

b. Graph-based recommender system

A graph G is a non-empty finite set V, which contains all the nodes on the graph, and a possibly empty set E, which contains all the edges on the graph. In the graph-based model, users and items are represented as vertices, and the interaction between users and items is represented as edges on the graph. Therefore, the relationship between users and items in the recommender system can be constructed as a bipartite graph. Among them, the node set V can be divided into two non-

empty sets V_{user} and V_{item} represent the user set and the item set (He et al., 2017a) respectively. The weight on the edge represents the distance relationship between the two vertices. In the recommender system, it is usually directly replaced by score (Gori and Pucci, 2007) or similarity (Han et al., 2017).

Since users and items and their relationships are suitable for describing with graph models, various theoretical methods based on graphs can also be applied to recommender systems. One such attempt is to use a probability graph model. In this model, the ranking of recommended items depends on the probability of reaching a vertex during a random walk on the graph. Many well-known algorithms such as PageRank(Page et al., 1998), or their variants (Gori and Pucci, 2007; Mao et al., 2017; Yildirim and Krishnamoorthy, 2008) are used in the recommendation model. In addition to the similarity of scores, other relationships such as trust degree (Jamali and Ester, 2009), label and classification (Liang et al., 2010), etc. are also included in the graph-based model.

c. Context-based recommender system

The traditional recommender system ignores the user's environment when making the recommendation, which is called Context. Contextual information, such as mood, location, time, equipment, etc., is very important for providing users with accurate recommendations(Adomavicius and Tuzhilin, 2015). recommender systems that incorporate context are called Context-aware Recommender Systems. For example, recommending a city center restaurant to a person who is active in the country is an inappropriate recommendation. In addition, there are a large number of examples showing that user preferences are also affected by context: for example, users' preferences on weekends may be different from those on weekdays; users may be alone with themselves when they are with friends Preferences are different;

users may choose different readings before eating breakfast and preparing for bed. Therefore, it is very important to understand the user's preferences that change with context for designing a good recommender system(Panniello et al., 2014; Wang et al., 2016b).

According to previous research, time is an important factor in contextual information (Hong et al., 2012). By using the time information in the recommendation process, the performance of the recommender system can be improved to a certain extent(Campos et al., 2014). A well-known model is the winning model timeSVD++(Koren, 2010) in the 2010 Netflix recommendation competition. Similar to context-aware recommender systems, this type of recommender system is called time-aware recommender systems, because time information is used as context information here. The time-aware recommender system has attracted a large number of researchers because the timestamp information is very easy to obtain and does not require additional actions by the user. Therefore, from a theoretical and practical perspective, time information should be paid attention to and meaningful when designing a recommender system. According to the different ways of using time information in recommender systems, time-aware recommender systems can be divided into two categories(Campos et al., 2014): categorical time-aware methods and time adaptive methods.

In the category time perception method, time information appears as a discrete variable. Baltrunas et al. designed some contextual variables, such as dividing the time of day into morning and evening categories, dividing the time within a week into working days and rest days, and dividing seasonal variables into hot seasons and Two categories in cold season etc. When the time-related information is expressed as categories, pre-filtering technology and post-filtering technology will apply these

time variables to the recommender system(Panniello et al., 2009; Rendle, 2010b). This kind of method is suitable for modeling periodic or periodic user preferences, but it is difficult to deal with the dynamic changes of user preferences.

Different from the category time perception method, in the time adaptive method, time information is used as a continuous variable. This type of method assumes that the user's current preferences are related to their recent behavior records in the system. The records closer to the current time reflect the user's preferences, and their weights should be greater. Most of the research in this area is related to Concept Drift, such as instance selection and time decay technology (Tsymbal, 2004). Ding and Li proposed exponential time screening weights on time series data (Ding and Li, 2005). Cao et al. proposed a dynamic model of user preferences to detect changes in user preferences on the sequence data of personal ratings(Cao et al., 2009). McAuley and Leskovec defined user records as time-based categorical variables, and proposed a method to simulate changing user preferences through user-contributed comments in a continuous time series(McAuley and Leskovec, 2013a). A temporal dynamic model is proposed to capture changes in context and user preferences to generate recommendations for social media (Yin et al., 2015). Zhang et al. used a transition matrix to model the drift of user preferences and applied it to the Bayesian matrix factorization model(Zhang et al., 2014). In some recommendation scenarios, such as music recommendation, users are more inclined to have periodic behaviors. This kind of scenario is closely related to the time-aware recommender system, because it needs to update the prediction of the next song the user wants to listen to in real time based on the user's feedback. In order to make the recommender system have the ability to model this periodic behavior, Du et al. proposed a time-sensitive recommender system model in 2015, which

generates reasonable recommendations by linking the self-excited point process with the low-rank model. Du2015. Another method predicts when they will visit again by analyzing frequently visited items in user behavior and combining their disappointment and rebuilding interest.(Kapoor et al., 2015).

To sum up, user preferences tend to change over time, but these changes happen for various reasons. However, up to now, time-sensitive recommender systems rarely detect and utilize changes in user interests. They are more inclined to model and adapt to the variable interests of individual users. However, a time-sensitive recommender system should give more consideration to the reasons that cause changes in user interests. These reasons can be user statistics, item attribute information, or context information. With the emergence of more recommendation intentions and application scenarios, in order to be able to model more fine-grained user interest drift, these influencing factors should all be taken into consideration.

2.1.2 Generic Feature-based recommendation algorithm

The recommendation algorithm based on general features regards the recommendation algorithm as a classification problem in traditional machine learning. Its input features include the behavior record of the user's access to the item, the attribute information of the user and the item, and the context information of the current recommendation. It can be simply divided into methods based on traditional machine learning and methods based on deep learning.

Recommendation algorithms based on general features in the pre-deep learning era basically belong to the world of generalized linear models. Logistic Regression (LR) as a simple and effective classification model has achieved good results in the field of recommender systems. In 2010, the factorization machine (Factorization

Machine, FM) (Rendle, 2010a) is proposed to solve the problem of manual feature cross relying heavily on the prior knowledge of the model designer, complex model adjustment and lag. The LR+GBDT model proposed by Facebook (He et al., 2014) uses a gradient boosting tree (Gradient Boosting Decision Tree, GBDT) to combine and filter features, and further achieve more accurate recommendation results in the production environment.

Since 2015, the blossoming of deep learning in various fields has enabled researchers in the field of recommender systems to get a lot of inspiration for improving current models. Deep learning technology has also achieved unparalleled advantages in the field of recommendation, and various model frameworks are emerging in endlessly. In order to enhance the generalization ability of the model, the researchers introduced a deep neural network to map the high-dimensional sparse features to the low-dimensional dense feature space. However, due to the unbalanced data distribution, the dense features of users and items in the long tail cannot be fully learned, leading to the problem of over-generalization.

In 2016, Google proposed the Wide&Deep model(Cheng et al., 2016), which combines the linear model with the deep neural network, while taking into account the generalization and memory of the model, and has become a classic model in the recommendation field. On the basis of Wide&Deep, the literature (Guo et al., 2017) replaced the shallow model with the factorization machine model, and proposed the DeepFM model, which enables the recommender system to automatically learn cross-features, thereby avoiding the original model The operation of manual feature engineering. At the same time, the model uses the original features as the common input of the shallow model part and the deep model part, ensuring the accuracy and consistency of the model features.

2.2 Deep Learning Techniques

Deep learning technology has always been regarded as a subarea of machine learning. The main feature of deep learning is to learn representations from data, which can understand user needs, project characteristics, and historical interactions through the extraction of latent features from the data. Research have found that as long as the data sample size is large enough and the deep learning network layers are deep enough, even without data pre-training, deep learning technology can be used to obtain good experimental results. The most representative deep neural networks are convolutional neural networks, recurrent neural networks and multilayer perceptrons. In addition, in order to improve performance, a variety of neural networks will be combined into a hybrid neural network.

2.2.1 Multi-layer Perceptron

Multi-layer **P**erceptron (MLP) is one of the feedforward neural networks with one or more hidden layers between input layer and output layer. Here, the perceptron can employ arbitrary activation function and does not necessarily represent strictly binary classifier. The structure of the shallow network is a single-layer input layer-hidden layer output layer. Shallow neural network is regarded as the simplest MLP, which is the structural basis of deep network. The difference between deep neural network and shallow neural network is that there is no specific layer limit for the middle hidden layer. The same point is that there is only one input layer. and the output layer.

Multilayer perceptrons can be used to solve linear inseparable problems, and the learning process is as follows. (1) The data is input through the input layer, and the

output is calculated through the forward propagation of the network. (2) Calculate the error according to the actual value. (3) Back propagation calculates the error, and updates the weight of each layer of the network according to the obtained error value.

2.2.2 Autoencoder

Autoencoder (AE) is an unsupervised model attempting to reconstruct its input data in the output layer. In general, the bottleneck layer is used as a salient feature representation of the input data. Autoencoder can be seen as a variant of the traditional multi-layer perceptron, first proposed by Rumelhart et al. (1986).

Autoencoder reconstructs the input data to learn the latent feature of the data through coding and decoding process. Autoencoder consists of a three-layer network in which the number of neurons in the input layer is equal to the number of neurons in the output layer, and the number of neurons in the middle layer is less than that of the input layer and the output layer. During network training, for each training sample, a new signal is generated at the output layer through the network. The purpose of network learning is to make the output signal and input signal as similar as possible. This similarity is represented by the reconstruction error. Autoencoder can form a deep structure by cascading and layer-by-layer training. After trained the deep model by layer-by-layer optimization, fine tuning can also be performed by allowing the entire network to reconstruct the input signal.

2.2.3 Convolutional neural network

Convolutional Neural Network (CNN) (De Andrade, 2019) is a special kind of feedforward neural network with convolution layers and pooling operations. It is

capable of capturing the global and local features and significantly enhancing the efficiency and accuracy. It performs well in processing data with grid-like topology.

2.2.4 Recurrent neural network

Recurrent Neural Network RNN is a deep network structure commonly used to process time series data. RNN can not only perform feedforward calculation, but also maintain the information of the previous moment, and use historical state data and current state to predict output(Medsker and Jain, 1999), so sequence data such as text and audio can be processed.

In order to solve the problem of information loss and gradient disappearance and explosion caused by too long time interval, a new variant Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho et al., 2014) are constructed. In the recommendation system , the recurrent neural network can be used for session-based recommendation, based on the user's current session behavior, learning the user's interest transfer process, and predicting the user's next possible interaction item

2.3 Deep Learning-based Recommender Systems

Recently, deep learning has been revolutionizing the recommendation architectures dramatically and brings more opportunities in reinventing the user experiences for better customer satisfaction. Recent advances in deep learning based recommender systems have gained significant attention by overcoming obstacles of conventional models and achieving high recommendation quality. Deep learning is able to effectively capture the non-linear and non-trivial user-item relationships, and enable

the codification of more complex abstractions as data representations in the higher layers. Furthermore, it catches the intricate relationships within the data itself, from abundant accessible data sources such as contextual, textual and visual information. Existing studies can be classified into two categories based on the types of employed deep learning techniques: models using single deep learning technique and deep composite model(Zhang et al., 2017b).

Deep learning has been revolutionizing the recommendation architectures dramatically and brings more opportunities in reinventing the user experiences for better customer satisfaction (Mu, 2018). Recent advances in deep learning based recommender systems have gained significant attention by overcoming obstacles of conventional models and achieving high recommendation quality. Deep learning is able to effectively capture the non-linear and non-trivial user-item relationships, and enable the codification of more complex abstractions as data representations in the higher layers. Furthermore, it catches the intricate relationships within the data itself, from abundant accessible data sources such as contextual, textual and visual information.

The power of deep learning algorithms is that they can learn and deal with complex problems like human beings. In the face of complex data, they can analyze and calculate linear or nonlinear feature sequences from multiple dimensions, and can automatically learn from massive data. The characteristics of user needs have been successfully applied in image recognition, speech recognition, natural language processing and other fields with good results. More and more researchers are also trying to apply deep learning in recommendation systems. The effective combination and in-depth study of deep learning technology and recommendation technology has become a new research direction. Existing studies can be classified

into two categories based on the purposes of employed deep learning techniques: models using deep learning technique for feature representation and models using deep learning technique for latent relationship modeling.

2.3.1 Embedding Techniques for Recommender Systems

Recommender systems usually use embedding technique to represent an entity with a low-dimensional dense vector, which can be an item, a user, etc. Embedding technique has become an indispensable component in recommender systems which mainly dealing with sparse features and inputting it into the neural network to train the model. It is also possible to use the relationship between these vectors, as a recall strategy, to filter out candidate items that match user interests(Jiang et al., 2022; Liu et al., 2022; Yu et al., 2022).

Grbovic and Cheng (2018) applied the embedding technique to characterize users and recommendation lists on Airbnb platform. On the basis of Skip-Gram, the embedding vectors of lists and users were designed for the platform in search ranking and real-time recommendation personalization. The data is used as similar sequence information, and the embedding vector of each listing is learned by using the word vector method, which effectively characterizes the multiple characteristics of the listing, and combines the actual business scenarios to accurately recommend high-quality listings to users.

Alibaba (Zhao et al., 2018) used embedding technology to learn the representation of ID type data, which is used for recommendation in e-commerce scenarios, including user ID, product ID, category ID, etc. The traditional one-hot encoding method will lead to too sparse data, and The potential relationship between objects cannot be represented. In the e-commerce platform, the ID data is very sparse, often

reaching hundreds of millions of dimensions. It is necessary to use low-dimensional vectors to efficiently express the ID data. This paper proposes an embedding-based framework based on Item2Vec(Barkan and Koenigstein, 2016) , by collecting the ID sequence of user behavior, combined with the structured relationship between IDs, a low-dimensional vector can be learned for different types of IDs to represent. On this basis, Alibaba (Wang et al., 2018) proposed a graph-based embedding. The method is used in the recommendation system, in order to solve the problem of sparse data, large amount of data and the existence of cold start of products in Alibaba's e-commerce. The method first constructs a directed graph of products based on sessions, and constructs behaviors that interact with products based on the graph. Sequence, combining the features to generate the graph embedding vector of the item, and weighting the features of each vector. This algorithm is mainly used in the recall phase, based on the products that have interacted with the user, recall the relevant candidate items.

Hidasi et al. (2015) proposed the SR-GNN model, considering the complex process of converting items into vectors, and proposed a new embedding method, which uses graph data structure to model user sessions, and uses graph neural network to learn graph nodes and embedding vector. Finally, each session is represented by the attention mechanism as the composition of the interest of the current session and the global interest, and based on each session, the interaction probability of the next item is predicted. The model overcomes the difficulty of representing items with latent vectors, using The graph-structured model generates accurate item embedding vectors, providing a new method for session-based recommendation scenarios.

Feature representation refers to learning features from content, which is generally used in content-based recommender systems. The performance of content-based recommendation methods heavily depends on efficient data feature extraction. The biggest advantage of deep learning is that it can learn data features through a general end-to-end process, and automatically obtain high-level representations of data without relying on artificially designed features. Therefore, deep learning is mainly used in content-based recommendation to extract the latent representation of the item from the content information of the item, and then generate the recommendation by calculating the matching degree between the user and the item based on the latent representation. Under the assumption that users and items carry auxiliary information, deep neural network models are used as effective feature extraction tools.

The convolution and pooling of convolutional neural network mainly learns the local features of data(Wang et al., 2020a), can extract unstructured multimedia data, and perform representation learning on multi-source heterogeneous data. The network can integrate diverse information, such as item images and comments, and is able to extract user visual interests or user preferences from text information. Convolutional neural network improves the scalability of the model, and integrating more information allows the model to capture user interests from more aspects. In recommender systems, convolutional neural networks Neural networks are suitable for multimodal recommendation, image recommendation and text recommendation tasks.

Usually users' behavior is easily affected by images, and bright product pictures can often attract users' attention. Zhou et al. (2016) tried to capture and analyze users' favorite images with convolutional neural networks to extract user visual

interest portraits. The system By calculating the cosine similarity of the visual interest vector, find the housing that matches the user's visual interest. The model is applied to the hotel reservation system, using image features to predict the user's favorite housing style, and realize personalized recommendation. Tang and Wang(Tang and Wang, 2018) proposed convolution The sequence embedding recommendation model regards the items that the user has interacted with in the past as a sequence, and predicts the items that the user may interact with in the future. Local features of images. The model uses convolutional neural networks to learn sequence features and latent factor models to learn user features.

Some related studies use convolutional neural networks to extract text features, integrate the model into textual information, and recommend relevant textual content of interest to users. Shen et al. (2016) used CNN for online learning resource recommendation, and the model used convolutional neural The project features are extracted from the text information such as the introduction and content of the learning resources, the language model is used for the input, and the latent factor model regularized by the L1 norm is used for the output. Gong and Zhang (2016) used CNN with attention to deal with the label recommendation problem. The whole model consists of two parts. The former part is used to obtain text features, and the latter part performs Softmax multi-label classification on the representation of each text. The CNN The model convolutional layer is applied to the pre-trained word vector, and the attention mechanism is added, and the attention layer is used to generate the weight of a word relative to its surrounding words.

Zheng et al. (2017) build a DeepCoNN model to jointly model user behavior and item attributes using textual reviews. An additional shared layer on top of the two neural networks connects the two parallel networks, so user and item

representations can interact with each other to predict clicks The whole model consists of 3 layers, the Lookup layer, the CNN layer, and the output layer. The Lookup layer converts user reviews and product reviews into corresponding word vectors, input them into CNN, and finally outputs the prediction results, and trains the model to reduce the error.

In 2020, JD.com proposed the CSCNN model Liu et al. (2020), which effectively utilizes the rich commodity category information in the e-commerce platform, uses convolutional neural networks to extract image information, and innovatively uses commodity information and commodity main images as the image feature extraction module. Input, extract the rich visual features in the main image of the product, effectively mine the visual attributes of the product, learn the impact of the product image on user behavior, and improve the accuracy of click probability prediction.Wu et al. (2021) leverage the stability of graph structure to incorporate a contrastive learning framework to assist representation learning. The proposed method is able to capture high-order similarity among users and items as well as structural connetivity. In this way, the semantics that users with similar interactions will have similar preferences are extended through multiple times of information propagation.

2.3.2 Models using Deep Learning Technique for Latent Relationship Modeling

Collaborative filtering is a widely used approach in recommender systems to solve many real-world problems. Traditional CF-based methods employ the user-item matrix which encodes the individual preferences of users for items for learning to make recommendation. In real applications, the rating matrix is usually very

sparse, causing CF-based methods to degrade significantly in recommendation performance. Some improved CF methods utilize the increasing amount of side information to address the data sparsity problem as well as the cold start problem. However, the learned latent factors may not be effective due to the sparse nature of the user-item matrix and the side information. Some researchers utilize advances of learning effective representations in deep learning, propose deep learning-based collaborative filtering methods, which is a kind of model based collaborative filtering recommendation methods for latent relationship modeling.

(1) **Multilayer Perceptron-based recommendation**

MLP can easily model the non- linear interactions between users and items. In 2016, the YouTube team (Covington et al., 2016) applied DNN to the video recommendation service, used neural networks to predict and score candidate videos, and generated a recommendation list according to the scores. The whole system is divided into a matching stage and a sorting stage. The matching stage uses an efficient recall strategy. Recalling candidate items that may be of interest to users from a video library with a scale of millions The neural network integrates user features, video attributes and scene information into the model, and predicts the score of candidate videos, sorts according to the scores, and selects high-scoring videos as the recommendation list.

Cheng et al. (2016) proposed a Wide & Deep model. The model consists of a Wide part and a Deep part. The Wide part uses a linear model to extract the first-order features of the data, and the Deep part uses neural The network automatically learns high-order features to improve the generalization ability, and finally integrates the results of the two parts through the Sigmoid activation function and outputs the prediction result. The Wide part in the model corresponds to the

memory ability of the model, and finds the difference between the features from the user's data. Relevance, which is biased towards recommending content related to user historical behavior. The Deep part corresponds to the generalization ability of the model. The sparse feature forms a low-dimensional dense vector through the embedding layer and is input into the hidden layer, using the learning ability of the neural network to capture new potential The combination of high-order features, the generalization ability is conducive to the personalization of the recommendation results, so that the recommendation results have diversity.

The multi-layer perceptron model is widely used in click-through rate estimation tasks. It can make full use of user portrait features, item attribute features and context information for feature extraction, and can alleviate the problems of data sparseness and high-order feature combination. The multi-layer perceptron and The combination of factorization machines can make up for the disadvantage that the feature combination in the FM and FFM models cannot be extended to third-order and above. Usually, after the sparse features are converted into low-dimensional vectors through the embedding layer, low-order cross-feature combinations are performed and DNN is used to extract high-level features. First-order feature combination, output click probability through Sigmoid function, such as DeepFM (Guo et al., 2017), FAT-DeepFFM (Zhang et al., 2019), NFM (He and Chua, 2017) and other models. Collaborative filtering is combined with neural network to alleviate the difficulty of training caused by sparse features, He et al. (2017b) proposed a neural collaborative filtering model, which combines the processing method of matrix decomposition with deep learning. The neural collaborative filtering model mainly models the implicit feedback data and uses embedding vectors to represent users and items which input these features into a multi-layer

neural network, and get the output layer predictions of user ratings, and uses a square loss function to train the model. This method uses the neural network to learn latent vectors to represent the potential relationship between users and items, and maps users and items to the latent vector space. The distance between vectors reflects the potential relationship between users and items, which can be used in the recall stage to calculate the correlation recall and the set of item candidates related to the target user.

Wang et al.(Wang et al., 2020b) discussed the impact of normalization operations on the CTR prediction effect, such as layer normalization, batch normalization, and variance-only normalization, for the CTR estimation task. Through comparative experiments , will normalize the vectorized features, use layer normalization for continuous numerical features, batch normalization for sparse categorical features, and use bias normalization in multilayer perceptrons to improve click-through rates Prediction accuracy. Huang et al.(Huang et al., 2020) borrowed the gate mechanism in computer vision and natural language processing to improve the trainability of non-convex neural networks, and added a gate mechanism in the embedding layer to select more important features from features. The hidden layer joins the gate mechanism to filter more important feature interactions and pass them to the deeper network. The idea of the gate mechanism is similar to the attention mechanism, which enhances effective features and suppresses data noise.

(2) Autoencoder-based recommendation

Autoencoder (AE) is an effective tool for learning low-dimensional feature representation, which can learn richer feature information and has been widely used in recommender systems. In the field of recommendation, the combined application

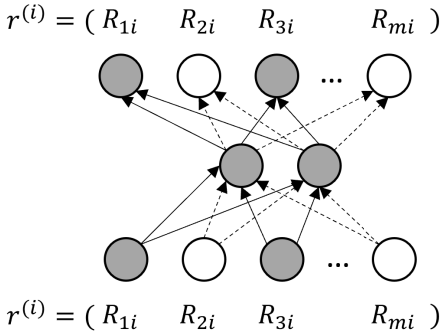


Figure 2.2 Illustration of AE-based Recommendation model.

of autoencoders and collaborative filtering methods can effectively alleviate the data sparsity problem.

Sedhain et al.(Sedhain et al., 2015) developed an autoencoder-based collaborative filtering recommendation method(AutoRec), the input of model is user based ratings or item-based ratings in the rating matrix *R*, produces an output through encoding and decoding process and optimizes the model parameters by minimizing the reconstruction error.

Figure 2.2 presents an example of AE-based collaborative filtering model. The grey entries indicate that the user has rated that item as the corresponding rating. The cost function of autoencoder-based collaborative filtering recommendation method aims at reducing the mean squared error. The rating prediction of autoencoder based collaborative filtering recommendation method is calculated by summarizing each entry of , then scaled by the maximum rating K. It uses RBM to pretrain the parameters as well as to avoid local optimum. Stacking several autoencoders together also enhances the accuracy slightly. However, there are two demerits of autoencoderbased collaborative filtering recommendation method: it fails to deal with non-integer ratings; the decomposition of partial observed vectors increases the sparseness of input data and leads to worse prediction accuracy.

The **C**ollaborative **D**eep **L**earning (CDL) model (Wang et al., 2015) is a classic hybrid recommendation model that effectively combines **S**tacked **D**enoising **A**uto**E**ncoder (SDAE) and **P**robabilistic **M**atrix **F**actorization (PMF) methods . SDAE is responsible for integrating auxiliary content information in the whole model, and the combination with PMF effectively balances the influence of auxiliary information and interaction data, and improves the recommendation accuracy of the model. This model alleviates the sparsity problem in collaborative filtering well and bridges the gap between autoencoders and collaborative filtering. However, the implementation environment of the model is severe, and the dependence on auxiliary information makes the load of the model face challenges. Based on the above model, Zhang et al.(Zhang et al., 2016) proposed the **C**ollaborative **K**nowledge based **E**mbedding (CKE) model. The model structure of CKE incorporates 3 types of auxiliary information. In order to fully learn auxiliary information features, the model uses the Bayesian embedding model to learn the vector representation of structural information, the Bayesian stack noise reduction autoencoder to learn the vector representation of text information, and the Bayesian stack volume **S**tacked **C**onvolutional **A**uto**E**ncoder (SCAE) learns vector representations of image information.

(3) Sequential recommendation

Sequential recommendation approaches utilize the sequential data to predict the users next action based on the last few actions. Depending on how these user historical behaviors are organized, sequential recommendations can be divided into two categories, user-based sequential recommendation and session-based recommendation.

User-based sequential recommendation treats user's complete historical behaviors as a sequence to generate recommendation. Some early machine learning methods can be adapted to perform user-based sequential recommendation. For example, some researchers try to model the user behavior sequence by Markov chains or Markov decision process. (Cheng et al., 2013) proposed extended factorized personalized Markov chain (FPMC) into FPMC-LR model to perform POI recommendation by adopting **B**ayesian **P**ersonalized **R**anking (BPR) on a user-location matrix. They model the sequential data via a stochastic process over discrete random variables. As known, the Markov property limits the dependencies of the process to a finite history. For example, in first-order Markov chains, the transition probability only depends on the previous state. That is to say, the next user action only depends on the previous one. Besides, the effectiveness and efficiency are hindered by the data sparsity and the explosion of the state space.

Despite these methods show promising results by utilizing users' sequential behaviors, which reflect users' dynamic and evolving interests, these methods overlook the intrinsic that the sequences are composed of sessions.

Different from above approaches, session-based recommendation regards the sequence of historical behavior in a continuous period of time as a session, and generates the recommendation result according to the state of the current session. In this setting, the users are usually assumed to be anonymous, and the personalized preferences (such as user profiles and ratings) are not provided explicitly(He et al., 2016).

In recent years, deep learning techniques have shown their promising effects on the accuracy of recommendations. Some researchers proposed to model the user behavior sequence by recurrent neural networks. For example, Hidasi et al.

(2015) proposes a recommender system based on Gated Recurrent Neural Network Architecture (GRU4Rec), which builds a ranking algorithm to model the user's historical behavior sequence. In subsequent papers, various variants of recurrent neural networks are proposed to improve the performance of GRU4Rec and adopt in many different scenarios. Some of these variants consider auxiliary information, e.g., personalized information (Quadrana et al., 2017), attribute information (Gu et al., 2016) and context information (Smirnova and Vasile, 2017). Some other variants introduce attention mechanism (Cui et al., 2019; Li et al., 2017) and different ranking loss functions (Hidasi and Karatzoglou, 2018) to improve the performance of recommendation task. In addition to recurrent neural networks, some approaches solved user-based sequential recommendation based on convolutional neural networks. Caser(Tang and Wang, 2018) is proposed to apply convolutional neural network instead of recurrent neural network to compress a sequence of items' embeddings into a low-dimension vector. Tuan and Phuong (2017) proposed using CNNs to learn feature representations form item content information (e.g., name, descriptions, identifier and category) to enhance the accuracy of session based recommendation. Pang et al. (2022) proposed to construct a heterogeneous graph in which the item transitions are captured by two types of item-item edge: in-type and out-type. Pan et al. (2022) also proposed to construct a global-level graph and a session-level graph, but it considers the learning on two graphs as two tasks, based on which multi-task learning is adopted.

Chapter 3

Enhancing Fashion Recommendation with Visual Compatibility Relationship

3.1 Introduction

A large portion of sales in the e-commerce are affected by fashion and lifestyle, which constitute apparel, footwear, bags, accessories, etc. Intelligent fashion recommendation received a lot of attention in computer vision and machine learning community (Ge et al., 2017; Kang et al., 2017; Wang et al., 2017). They have huge potential profits for the fashion industry. A lot of companies have established their own recommender systems to give users advice to enhance their shopping experience, such as Amazon, Alibaba and eBay (Mao et al., 2019; Wu et al., 2015).

Many approaches have been proposed to analyze user preferences on fashion criterion and generate personalized recommendation. Most of fashion recommen-

(a) Compatible outfits.

(b) Incompatible outfits.

Figure 3.1 Examples of compatible and incompatible outfits.

dation approaches take into account characteristics of image, as visual information is one of the most important factor in describing fashion items (He and McAuley, 2016b; Kang et al., 2017; Lynch et al., 2016). Such approaches can substantially improve recommendation accuracy than others without visual information.

However, a few of them considers the problem of compatibility of fashion items. We know that when we choose a piece of clothing, it is not just a matter of considering the style of the dress. We also need to consider its effect with other clothes we wear. Some examples of compatibility and incompatibility outfits have been shown in Figure 3.1. Normally one would not pair a red T-shirt with green pants. Moreover, a black robe is incompatible with a pair of pink running shoes. This is partially because it is difficult to model compatibility relationship between fashion items.

When designing this recommendation system, we mainly consider the problem of learning visual compatibility relationship of items on pixel level. The

visual compatibility relationship that needs to be learnt is whether fashion matching between one item and another item conforms to the human aesthetic by understanding the picture. In the traditional fashion recommendation, authors often only consider the styles and categories of clothes and ignore the sense of harmony between items as an outfit. In this chapter, our approach considers visual compatibility relationship to recommend fashion items, which is closer to the actual needs of people.

In order to learn the knowledge of the matching model between fashion items, we face two challenges: 1) How to learn the common domain knowledge about fashion compatibility relationship between items. In other words, there are a few of outfits that we observed on the online shopping website for a single person and fashion concept is often subtle and subjective for different customers. 2) How to incorporate the learnt domain knowledge into our recommender system. For the first challenge, we propose a novel method to incorporate the compatibility relationship knowledge into the image representation. Our method allows learning an embedding from the images of the fashion items to a latent space, so that two items that is a good match are close in this latent space and items that don't match are far apart. An external dataset which contains a number of outfits being given by experts is also been used to train our model. For the second challenge, we adjust the popular BPR (Rendle et al., 2012) model to include the compatibility relationship knowledge that we learnt. Moreover, because we use the external dataset to learn the domain knowledge between the items in order to solve the problem of the distribution gap between the source domain and target domain, we propose a domain adaptation method to alleviate this difference.

In this chapter, the main contributions are summarised as follows::

- Proposing a fashion compatibility relationship learning method that incorporates visual compatibility relationships as well as style information into a visual embedding.
- Proposing a fashion recommendation method with domain adaptation strategy to alleviate the distribution gap between the items in target domain and the items of external compatible outfits.
- Conducting a case study to illustrate how our method understands images. Furthermore, through an extensive set of experiments on several datasets, we demonstrate our method significantly outperforms several alternative methods.

3.2 Notations and Problem Formulation

We first explain the symbols used in this chapter. We will consider $\mathcal{U}$ be the set of all users and $\mathcal{I}$ the set of all items. For each item, we have a corresponding images $\mathcal{V}_i$ which can represent this product. $R \subseteq len(\mathcal{U}) \times len(\mathcal{V})$ is the implicit feedback matrix whose rows correspond to customers and whose columns correspond to products. This means that $R_{ui} = 1$ stands for user u has bought item i, and 0 otherwise. We also have a hand-crafted clothing collocation dataset $X = \{x_1, x_2, ..., x_n\}$ labelled by experts which contains a set of clothing images and a clothing collocation pair list $C = \{(x_i, x_j) | x_i, x_j \in X\}$. Note that despite in the item set $\mathcal{V}$ and clothing collocation set X are both contains clothing images, the images in this two datasets are differences and no overlapped items. Furthermore, we can incorporate additional information like category data of products or demographic data about customers. However, we just focus on sales and visual information which

Table 3.1 Major Notations Used in This Chapter

Notation	Description
$\mathcal{U}, \mathcal{I}$	user set, item set
$\mathcal{V}$	image set of items
R	implicit feedback matrix
X	the set of images in clothing collocation dataset
C	clothing collocation pair list
$\mathcal{P}_u, \mathcal{V}_u, T_u$	positive item set of user u in training/validation/test sets
f_i	image representation vector of image i
γ_u, γ_i	latent features of user u, item i
θ_u, θ_i	visual features of user u, item i
W_{enc}, W_{dec}	weights of encoders and decoders, respectively

is very important in fashion recommendation. Table 3.1 lists the major notations used throughout this chapter.

The fashion recommendation task with visual compatibility relationship to be solved in this chapter is to provide a personalized ranking list to each user with the help of visual information. First, given a set of fashion items $X = \{x_1, x_2, ...\}$ and collocations using these items C, learning visual compatibality knowledge is to learn an embedding $\mathcal{F}$ where the distance between item i and j, $d(\mathcal{F}(x_i), \mathcal{F}(x_i))$, is as small as possible if $(x_i, x_j) \in C$. After that, with user interaction records and item images, we try to learn the user's preference towards collaborative information and visual information to generate a ranking list for each user.

3.3 Visual Compatibility Relationship Modeling and Recommendation

We propose a fashion recommendation method which considers compatibility relationship between fashion items. In this method, we combine the collaborative

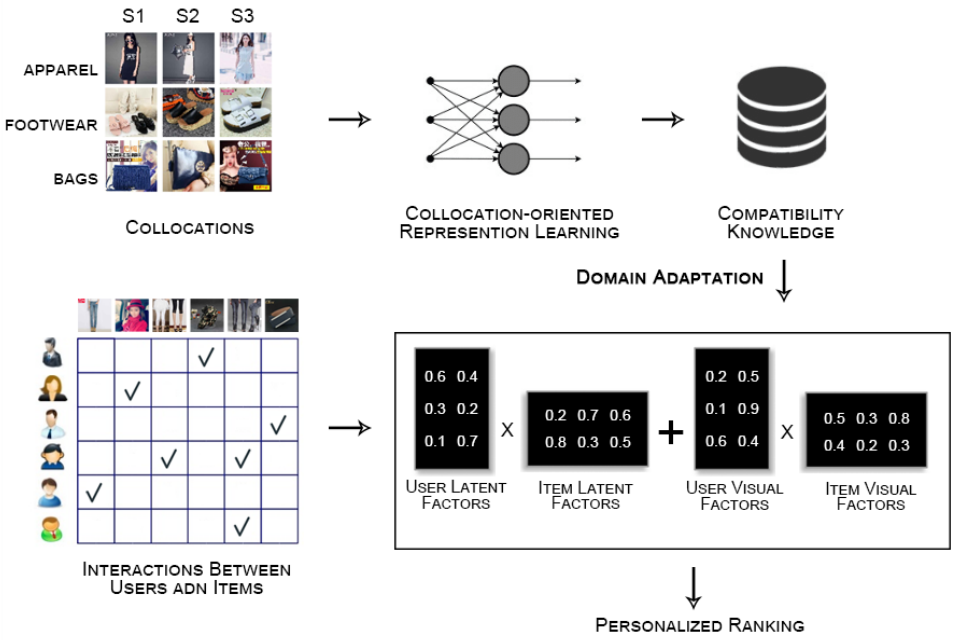


Figure 3.2 The proposed framework of our method.

information among users and items with compatibility knowledge. In order to allow the algorithm to understand the aesthetics of humans, we carefully construct an image representation model, through which we can determine what kind of information the resulting representation contains. This makes the computer learn the compatibility knowledge that people understand. After that, we incorporate the generated compatibility knowledge into our recommendation framework with a domain adaptation strategy. The framework of the entire recommender system is shown in Figure 3.2.

3.3.1 Learning visual compatibility knowledge from fashion items

In this section, our goal is to learn the visual compatibility relationships between fashion items. Conventional methods are mostly relying on category information to learn image representations. Instead of annotating images with labels or categories, which is costly, we leverage the weakly-labeled web data provided by the external dataset to learn compatibility knowledge.

Since there is no fixed category for the tasks we are going to perform, we cannot use the softmax-based cross entropy loss function for training. So, we chose triplet network to learn the image representation. The advantage of the triplet network is the distinction of details, that is, when the two inputs are similar, the triplet network can better model the details, which is equivalent to adding two measures of the difference of the input differences to learn a better representation of the input. The structure of the network is shown in the Figure 3.3.

In our task, we take the first item in an item pair in list C as an anchor, the second item as a positive sample, and select an item that is not in the list as the negative sample. More specifically, we can't randomly select negative samples on the entire candidate set, because this will cause $d(A,N)$ to be much larger than $d(A,P)$, which will make the model unable to fully train and enter the prematurely. The state of the joint. So at each training, we need to choose a negative sample with $d(A,P)$ as close as possible to $d(A,N)$. This may make the model as difficult as possible to reduce the risk of overfitting.

Given a fashion image set, $X = \{x_1, x_2, ...\}$, x_i is a picture containing the t-th item. Also give a list $C = \{(x_i, x_j)|x_i \in X\}$ which denotes clothing collocation pairs. We need to learn a x_i to f_i mapping function, so that x_i and x_j are as close

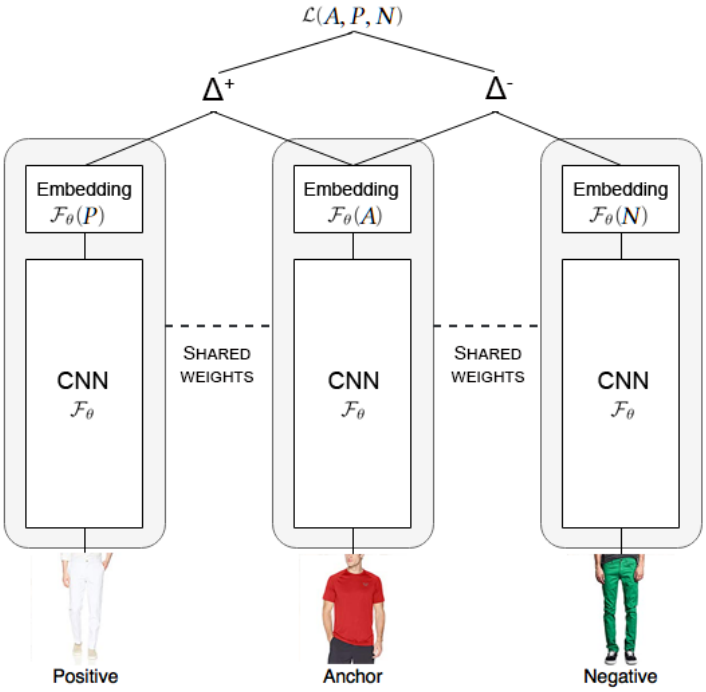


Figure 3.3 The illustration of our model for learning compatibility knowledge.

as possible, if the two items have good compatibility, and by contrast, the distance between f_i and f_j is as far as possible is i and j are not good compatibility. In other words, we try to learn a new representation of images. More formally, we minimize the following objective function:

$$\mathcal{L}(A,P,N) = max(||\mathcal{F}(A) - \mathcal{F}(P)||_2 - ||\mathcal{F}(A) - \mathcal{F}(N)||_2 + \alpha, 0) \tag{3.1}$$

where $\mathcal{F}$ is the mapping function. A is the anchor item, P is the item which have good compatibility with A, and N is not a good compatibility. α is the threshold parameter.

As shown in Figure 3.3, the network contains three sub-networks which shares weights with each other. In each sub-network, we encode the item image with

Convolutional Neural Networks. There are many ConvNets architectures to choose from, and we use a variant of AlexNet for simplicity. The AlexNet variant, is the same as the original, except that we do not use pretrained weights and we replace local response normalization with batch normalization. We use the output of the fc6 layer as the encoding feature of the input image. The dimension of the image encoding is 4096.

Using more powerful architectures (e.g. Szegedy et al. (2017)), may achieve better performance; however, we found that AlexNet is sufficient to show the effectiveness of our method. Because this network has fewer parameters, it can be trained more easily and reduces the risk of overfitting during training.

3.3.2 Fashion recommender system with visual compatibility knowledge

As mentioned above, the recommendation task can be regard as a ranking problem according to the user's preference. Our preference predictor is based on the basis of Matrix Factorization, which is the most promising model for rating prediction as well as modeling implicit feedback. The most related work to this problem is the VBPR model proposed in He and McAuley (2016b), which learns the visual user preference predictor using a pairwise ranking optimization framework.

We conduct our model based on pairwise learning. We defined the preference predictor as same as VBPR except for the reduce dimension method. To avoid missing information, we use an autoencoder to process dimension reducing. The formulation of encoder and decoder of autoencoder are as follows:

$$Enc(\mathcal{I}) = W_{enc} \cdot \mathcal{F}(\mathcal{I}) + b_{enc} \tag{3.2}$$

$$Dec(\mathcal{I}) = W_{dec} \cdot Enc(\mathcal{I}) + b_{dec} \tag{3.3}$$

Thus, the final preference predictor are as follows:

$$\hat{r}_{u,i} = \alpha + \beta_u + \beta_i + \gamma_u^T \gamma_i + \theta_u^T Enc(f_i) \tag{3.4}$$

For this implicit feedback ranking problem, we conduct the pairwise ranking optimization framework to train the model. The objective is as follows:

$$\max_{\theta} \sum_{(u,i,j) \in D_S} \ln \sigma\left(\hat{r}_{uij}\right) - \lambda_\Theta \|\Theta\|^2 + ||f_i - \hat{f}_i||^2 + ||f_j - \hat{f}_j||^2 \tag{3.5}$$

where $\hat{r}_{uij} = \hat{r}_{u,i} - \hat{r}_{u,j}$.

Because the compatibility knowledge is learned from an external dataset also called source domain, images from the external dataset and the target dataset belong to different feature space. For example, in our experiments, the backgrounds of images in the source domain are very excursive. But in the target domain, the backgrounds are very clean and neat. Thus, we propose a domain adaption method which uses the domain adaptation technique to ensure that knowledge extracted from the source domain is consistent with the target domain and that knowledge transfer is positive.

When two items are bought by a customer at the same time, it regards as a co-occurrence pair. Most of the time, in the domain of fashion recommendation, we can assumption that the co-occurrence items should be a good clothing matching. Thus, we add the co-occurrence similarity into the objective above as follows to

alleviate the distribution gap between source domain and target domain:

$$D_{i,j} = Frequent_{i,j} * ||Enc(f_i - f_j)|| \tag{3.6}$$

where $Frequent_{i,j}$ is the frequency of co-occurrence in the train set. When item i and item j are bought in a same bundle, we assume that they are a good match, and the distance between them should be very closer.

In the training procedure, the training set D_S consists of triples in the form of (u,i,j,c), where u denotes the user and item i which they expressed positive feedback, and a non-observed item j. c is the co-occurrence frequency. It can be formalized by:

$$D_S = (u,i,j,c)|i \in I_u^+ \wedge j \in I_u^+ \tag{3.7}$$

The final formulation is as follows:

$$\sum_{(u,i,j)\in D_S} \ln \sigma\left(\hat{r}_{uij}\right) - \lambda_\Theta \|\Theta\|^2 + ||f_i - \hat{f}_i|| + ||f_j - \hat{f}_j|| + D_{i,j} \tag{3.8}$$

3.4 Experiments and Analysis

We perform experiments on several datasets to evaluate the performance of the proposed method. All experiments were conducted on a workstation with a 6-core Intel CPU and two Titan-X (Pascal) graphics cards. Although there is a huge number of images and transaction records, it is still possible to train our model in half of day.

3.4.1 Datasets and evaluation metrics

The first dataset was provided by *Taobao.com* which is one of the most famous Chinese website for online shopping. It consists of clothing collocation suggestions from fashion experts, image data of Taobao items, and user behavior data. In this dataset, each line represents an item list which delimited by semicolon, every semicolon refers to a Collocation set. Every collocation set includes several goods, delimited by comma. We formatted this dataset into a pair-wise format, which means these two items is a good matching.

Another group of datasets contains user transaction records from two different sources. The first one were introduced in He and McAuley (2016b) and consist of reviews of clothing items crawled from *Amazon.com*. It was separated into 4 subcategories, named *Amazon Fashion*, *Amazon Women* and *Amazon Men*. The other one was crawled from *Tradesy.com*, which includes serveral kinds of feedback, like clicks, purchases, sales, etc.

The statistical information for the four datasets is provided in Table 3.2.

Table 3.2 Dataset statistics

Dataset	Users	Items	Interactions
Amazon Fashion	64,583	234,892	513,367
Amazon Women	97,678	347,591	827,678
Amazon Men	34,244	110,636	254,870
Tradesy.com	33,864	326,393	655,409

We measure recommendation performance of our method by calculating AUC and diversity. The AUC measures the quality of a ranking based on pairwise comparisons. Formally, we have

$$AUC = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{D}_u|} \sum_{(i,j) \in \mathcal{D}_u} \xi \left(r_{u,i} > r_{u,j} \right) \tag{3.9}$$

where $\mathcal{D}_u = \{(i,j) | (u,i) \in \mathcal{T}_u \wedge (u,j) \notin (\mathcal{P}_u \cup \mathcal{V}_u \cup T_u)\}$. In other words, we are calculating the fraction of times that the 'observed' items i are preferred over 'non-observed' item j.

AUC is a typical metric to assess the recommendation performance in reproducing known user opinions that have been removed from the test dataset. The risk of such a metric is that, with recommendations based on similarity and overlap, customers will be exposed to a narrowing band of popular commodities. In other words, we also need other metrics to evaluate the recommendation performance.

Personalization, also named Inter-user diversity, considers the uniqueness of different customers' recommendation list. Given two users i and j, the different between their recommendation lists can be measured by the inter-list distance,

$$h_{ij}(L) = 1 - \frac{q_{ij}(L)}{L} \tag{3.10}$$

where $q_{ij}(L)$ is the number of common items in the top L places of the both lists: if the two lists are identical, $q_{ij}(L) = 0$ will equals 0 whereas completely different lists have $q_{ij}(L) = 1$.

Averaging $h_{ij}(L)$ over all pairs of users we obtain the mean distance $h(L)$, for which greater or lesser values mean, respectively, greater or lesser personalization of users' recommendation lists.

3.4.2 Experimental settings and baselines

Table 3.3 Recommendation Performance in Terms of AUC and Diversity with different sparsity

		AUC			Diversity		
		D=0.0010	D = 0.0005	D = 0.0001	D=0.0010	D = 0.0005	D = 0.0001
Amazon Fashion	*POPRANK*	0.5553	0.5627	0.6298	0.0000	0.0000	0.0000
	BPR-MF	0.5866	0.5951	0.6163	0.5621	0.5285	0.3093
	VBPR	0.6953	0.7116	0.7503	0.9715	0.9861	**0.9945**
	DVBPR	0.6134	0.6199	0.6497	**0.9826**	**0.9920**	0.9856
	CO-BPR	**0.7126**	**0.7242**	**0.7723**	0.9806	0.9891	0.9926
Amazon Women	*POPRANK*	0.5534	0.5897	0.6426	0.0000	0.0000	0.0000
	BPR-MF	0.5884	0.6176	0.6437	0.6335	0.5548	0.3786
	VBPR	0.6747	0.6861	0.7161	0.9777	0.9868	**0.9935**
	DVBPR	0.6209	0.6305	0.6714	0.9859	0.9890	0.9871
	CO-BPR	**0.6792**	**0.6901**	**0.7295**	**0.9952**	**0.9921**	0.9898
Amazon Men	*POPRANK*	0.5607	0.6118	0.6538	0.0000	0.0000	0.0000
	BPR-MF	0.5969	0.6269	0.6447	0.5655	0.4583	0.3501
	VBPR	0.6754	0.6857	0.7164	0.9716	0.9760	0.9870
	DVBPR	0.6270	0.6471	0.6726	0.9796	0.9788	0.9797
	CO-BPR	**0.6815**	**0.6982**	**0.7358**	**0.9836**	**0.9902**	**0.9961**
Tradesy.com	*POPRANK*	0.4105	0.3939	0.4756	0.0000	0.0000	0.0000
	BPR-MF	0.5830	0.5763	0.5317	0.8522	0.8813	0.8694
	VBPR	0.6553	0.6819	0.6927	**0.9923**	**0.9964**	**0.9931**
	DVBPR	0.6134	0.6199	0.6497	0.9826	0.9920	0.9856
	CO-BPR	**0.6718**	**0.6873**	**0.7106**	0.9874	0.9901	0.9920

We compared the proposed method in terms of accuracy and diversity against the following baselines:

- PopRank: Always recommends the top-k most popular items to users.
- BPR-MF (2009): This is a content-free algorithm based on matrix factorization which is designed for top-k recommendation tasks Rendle et al. (2012). It optimizes pair-wise preferences between observed and unobserved items.
- VBPR (2016): It is a state-of-the-art image-based recommender system proposed by He and McAuley (2016b) for implicit feedback. The authors incorporate visual information provided by a pre-trained CNN.
- DVBPR (2017): This is the extension of VBPR by learning 'fashion aware' image representations directly (Kang et al., 2017).
- CO-BPR: Our method proposed in this chapter.

We carefully choose the hyper-parameters and tuned them via grid search for each baseline method. For BPR-MF, VBPR, DVBPR and CO-BPR, we used a mini-batch size of 32 for all experiments. The number of latent factor selected from $\{6, 8, 10, 12, 14, 16, 18, 20\}$. We set it to be 12 in all experiments.

3.4.3 Results

We evaluated our proposed method by comparing it to state-of-the-art methods using some real-world datasets. We report recommendation performance in terms of the AUC and diversity in Table 3.3. Data with three sparsity ratios in target domain are chosen as training set. Comparing all the methods on these four datasets, we make the following observations:

1) Compared with POPRANK method on Amazon datasets, the value of AUC increase with the increase of sparsity ratio. It tells us that customer more likely to purchase popular items on Amazon. However, on Tradesy.com, customers prefer to choose unpopular items.

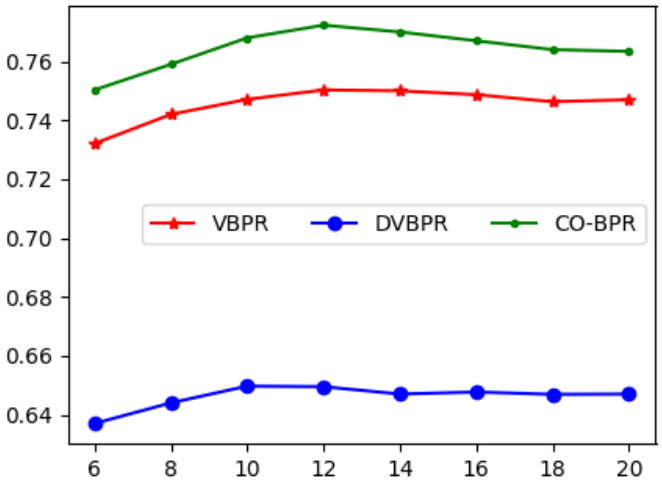
2) Compared with methods without visual information, we can see that visual information substantially improve recommendation accuracy and diversity.

3) Compared with VBPR and DVBPR, we can see that learning visual compatibility relationship from an external dataset is very effective. Our proposed method CO-BPR outperforms all the comparison methods on all the four datasets. This demonstrates the significant benefits of generating recommendations with visual compatibility relationship.

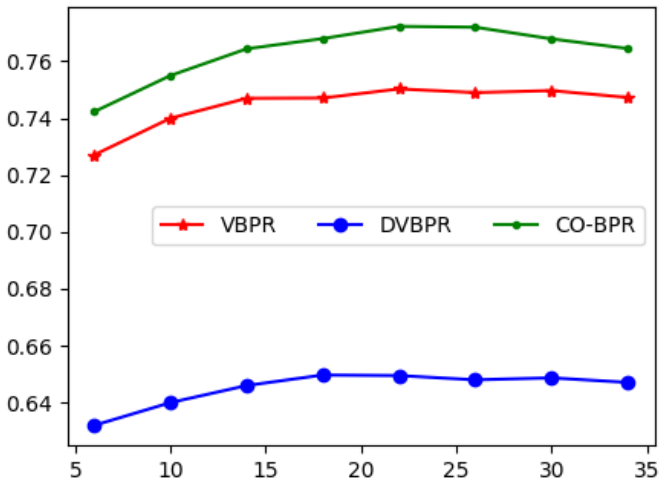
Furthermore, to investigate the dimensionality sensitivity, we illustrate the performance of VBPR, DVBPR and CO-BPR with varying dimensionality in Figure 3.4. It is clear that both latent dimensionality and visual dimensionality are note very sensitive.

3.5 Summary

In this chapter, we have introduced a novel method for the fashion recommendation task with learning compatibility knowledge in visual aspect. A triplet network is used to learn compatibility knowledge from an external dataset. Domain adaptation strategy is used to alleviate the distribution gap between source domain and target domain. The experimental results show that our method is superior to several baselines in the AUC and diversity indicators.



(a) latent dimensionality



(b) visual dimensionality

Figure 3.4 Performance of VBPR, DVBPR and CO-BPR with varying dimensionality measured by AUC.

Chapter 4

A Deeper Graph Neural Network for Recommender Systems

4.1 Introduction

Recommender systems have become increasingly important in recent years due to the problem of information overload. Recommender systems allow individuals to acquire information more effectively by filtering information. Over the years, collaborative filtering has become the most successful and widely used recommendation technique (Shi et al., 2014). The core assumption is that users who have expressed similar interests in the past will share similar interests in the future. Popularized by the Netflix Prize, **M**atrix **F**actorization (MF) has become the de facto approach to collaborative filtering-based recommendation. Much research effort has been devoted to enhancing the MF method, one of the most powerful collaborative filtering techniques, such as integrating it with neighbor-based models (Koren, 2008), extending it to factorization machines for the generic modeling of

features (Rendle, 2010a), and optimizing it with Bayesian personalized ranking objective to adapt it to implicit feedback recommendation (Rendle et al., 2012). However, despite these efforts, sparsity is still one of the most challenging issues facing us today.

Recent years have seen a surge in research on graph neural networks, leading to substantial improvements in the performance of tasks with graph-structured data, which is fundamental for recommendation applications. One of the most prominent approaches is **G**raph **C**onvolutional **N**etwork (GCN) (Defferrard et al., 2016). The core idea behind GCNs is finding a way to iteratively aggregate feature information about graph structure and the structure of the node's local graph neighborhood into a machine learning model. The goal is to learn a mapping that embeds nodes as points in a low-dimensional vector space R^d. The primary contribution of representation learning approaches is that of finding a way to represent, or encode graph structure which geometric relationships in the embedding space reflect the structure of the original graph. However, GCNs need to operate on the Laplacian eigenbasis which leads to a huge time consumption on large graphs.

In this chapter, we view the recommendation task as a link prediction problem on a bipartite graph: the interaction data in collaborative filtering-based methods can be represented by a bipartite graph between user and item nodes as shown in Figure 4.1. The MF approaches can then be considered as learning a mapping from users/items to a low-dimensional vector, where the interaction information is contained in the vectors.

We propose a general framework named GCF, short for Graph neural network-based Collaborative Filtering, which builds on recent progress in graph neural networks. The framework contains a larger receptive field with iterative information

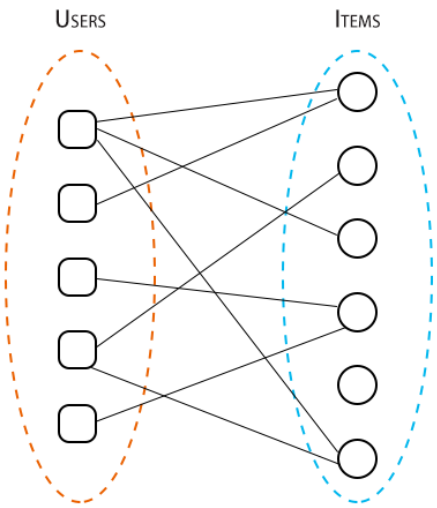


Figure 4.1 Illustration of the bipartite graph of a user-item interaction matrix.

propagation enabling our method to access more information in the process of making decisions. The concept of receptive field on graph neural networks will be given in the following section.

We present an attention-based message-passing method to carry out the information propagation process. In the recommendation task, the variable size input for each layer is a challenge, because the number of neighbors for each node is different. To solve this problem, we assign different weights for neighbors.

The main contributions of this chapter are summarised as follows:

- Presenting a general framework GCF to model the latent features of users and items. We also show that MF is a special case of GCF when the number of hidden layers is no more than one.
- Proposing an attention-based message-passing method to solve the variable size input problem.

- Performing extensive experiments on three real-world datasets to demonstrate the performance of our GCF method. The results show that the proposed method outperforms the state-of-the-art methods in terms of HR@k and NDCG@k.

4.2 Problem Formulation and Motivation

In this section, We formalize the problem and discuss existing solutions for collaborative filtering with implicit feedback.

4.2.1 Recommendation and Link Prediction in Bipartite Graphs

Let N_u and N_v denote the number of users and items, respectively. We define the user–item interaction matrix $Y \in \{0,1\}^{N_u \times N_v}$ from users' implicit feedback, where y_{ui} equals 1 if interaction $(user\ u,\ item\ i)$ is observed, 0 otherwise. Here $y_{ui} = 1$ indicates that there is an interaction between user u and item i. This means that user u likes item i. However, a value of 0 does not necessarily mean that user u does not like item i, it can be that the user is not aware of the item.

The recommendation problem with implicit feedback is formulated as the problem of estimating the scores of unobserved entries in Y. Formally, they can be abstracted as learning $\hat{y}_{ui} = f(u, i \mid \theta)$, where $\hat{y}_{ui}$ is the predicted score of interaction y_{ui}, θ denotes the model parameters, and f denotes the function that maps the model parameters to the predicted score.

We can also treat the user-item interaction matrix as a bipartite graph, where users and items are nodes in the graph as shown in Figure 4.1, and interactions are edges in the graph. The edge between node u and node i indicates that an interaction

between user u and item i is observed, and vice versa. Thus, the recommendation problem can be represented by the problem of predicting the probability of an edge between a user node and an item node. Formally, the users' implicit feedback can be represented by an undirected graph $G = (\mathcal{U}, \mathcal{V}, \mathcal{E})$, where $u_i \in \mathcal{U}$ is a collection of user nodes with $i \in \{1, ..., N_u\}$, and $v_j \in \mathcal{V}$ is a collection of user nodes with $j \in \{1, ..., N_v\}$. The edge $(u_i,\ v_j) \in \mathcal{E}$ denotes that the interaction $(user\ u,\ item\ i)$ is observed, and vice versa.

4.2.2 Factorization Models

Machine learning algorithms in recommender systems are typically classified into three categories: content-based, collaborative filtering-based and hybrid methods. Content-based methods follow the assumption that a user prefers items that have similar attributes to items previously preferred by that user. In contrast, collaborative filtering-based methods do not rely on item content description, but generate recommendations according to users who have shared similar interests in the past (Mao et al., 2017). Hybrid methods combine both content-based and collaborative filtering-based methods. Collaborative filtering-based methods have been widely used because they enable users to discover new content that is dissimilar to items viewed in the past. In this study, we also concentrate on collaborative filtering-based techniques.

The factorization model is one of the most promising categories of collaborative filtering-based recommendation. One of the most famous methods used in the context of recommendations and usually referred to as "SVD" is not strictly speaking the mathematical Singular Value Decomposition of a matrix but rather an approximate way to compute the low-rank approximation of the matrix by

minimizing the squared error loss. The basic idea is to decompose the original and very sparse matrix into two low-rank matrices that represent user factors and item factors (Koren, 2009). User and item factors are simply multiplied to predict the score of interaction y_{ui}.

$$\hat{y}_{ui} = b_u + b_i + p_u^T q_i \tag{4.1}$$

where b_u and b_i is the bias of user u and item i, p_u is the user latent factors of user u and q_i is the item latent factors of item i.

Another well-known algorithm is SVD++, which is the enhanced model of SVD (Koren, 2008). In this method, user latent factors are combined with item latent factors. This is because the author believes that a user-rated item is in itself an indication of preference. This has a positive effect on new users for whom there might not be enough data points to generate good user factors. The formulation of this method is as follows:

$$\hat{y}_{ui} = b_u + b_i + q_i^T \left(p_u + |N(u)|^{-1/2} \sum_{j \in N(u)} y_i \right) \tag{4.2}$$

where $N(u)$ is the positive feedback of user u and y_i is the personalized preference bias of who likes item i.

SVD and SVD++ are the simplest algorithms of collaborative filtering-based methods. The matrix factorization methods have been proven to be efficient and effective in many situations (Koren et al., 2009; Symeonidis, 2016; Wang et al., 2016a). However, one of the challenges of these methods is the sparsity problem. As a result of this problem, the collaborative filtering-based method cannot find enough similar users to support the decision-making process. At present, a large number of methods based on the idea of matrix decomposition (Bobadilla et al., 2013; Lu et al.,

2015) have been proposed to address this problem. A non-uniform item sampler has been proposed to address the problem in which the convergence of stochastic gradient descent learning algorithms slows down if the item popularity has a tailed distribution (Rendle and Freudenthaler, 2014). Because a good recommender particularly emphasizes accuracy near the top of the ranked list, a new pairwise ranking loss has been proposed to reduce computational complexity (Yuan et al., 2016). A series of cross-domain techniques have been proposed to utilize data from an external domain to alleviate the sparsity problem (Zhang et al., 2018, 2017a). All of these methods attempt to address the sparsity problem in different aspects.

4.2.3 Graph Neural Network

Our work builds upon a number of recent advancements in deep learning methods for graph-structured data. Graph neural networks consist of an iterative process, which propagates the node information until equilibrium and produces an output for each node based on its information. It was first outlined in (Gori et al., 2005) and further elaborated on in (Scarselli et al., 2009). (Bruna et al., 2013) proposed a graph convolution based on spectral graph theory. Following on from this work, a number of authors have proposed improvements, extensions, and approximations of these spectral convolutions (Bronstein et al., 2017; Defferrard et al., 2016) which have proved the effectiveness on node classification and link prediction, as well as recommender systems (Monti et al., 2017). These approaches have consistently outperformed techniques based upon matrix factorization or random walks. However, the learnt filters in the spectral approaches depend on the Laplacian eigenbasis and spectral decomposition, which is prohibitively expensive on large graphs.

There are also several non-spectral approaches (Atwood and Towsley, 2016; Duvenaud et al., 2015), which define convolutions directly on the graph. (Hamilton et al., 2017) introduced GraphSAGE, a method for computing node representations in an inductive manner. This approach has yielded impressive performance across several large-scale inductive benchmarks, however, the technique only deals with a fixed size neighborhood of each node.

4.3 Graph Neural Network-based Collaborative Filtering

In this section, we first present the general GCF framework. We then show that SVD and SVD++ can be expressed under GCF with node embedding via graph neural network. To address the problem of dealing with variable size inputs in the information propagation process, we propose a new method with an attention mechanism which assigns different weights to the neighbors of each node.

4.3.1 General framework

Factorization models essentially learn the representation of users p_u and the representation of items q_i with a distance metric d to represent the similarity between p_u and q_i. Recommendations are then generated based on the similarity ranking.

In this chapter, we use a graph neural network to learn p_u and q_i. In Figure 4.2, the boxes represent users and the circles represent items. A bipartite graph consists of two parts: node information, also known as latent factors and structural

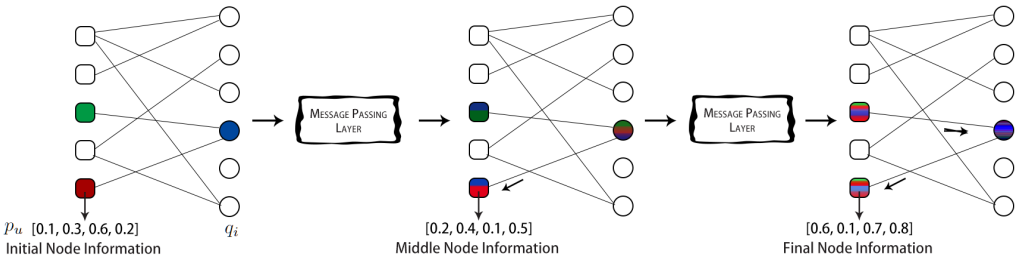


Figure 4.2 Illustration of the proposed framework.

information. We use p_u and q_i to represent the node information in the graph, and a user-item interaction matrix to represent the structure information of the graph.

The node information can be broadcasted along the edges on the graph. In the first step, as shown in Figure 4.2, the user node receiving a message from the item nodes means that the items related to the user also represent the user's preference. The latent factors of these items are therefore contained in the user's latent factors. At the same time, item nodes in the graph also receive a message from the user nodes for the same reason. In the second step, the user receives a message from the related items again, but the message not only contains the latent factors of items but also the latent factors of users related to this item. Node information will be broadcasted several times on the graph with multiple message-passing layers. Finally, we can obtain p_u and q_i with a wider scope of information by integrating the messages from the user nodes and the item nodes.

In convolutional neural network, the receptive field is defined as the region in the input space that a particular CNN's feature is looking at. A larger receptive field means that more information is integrated into the particular feature. In the same situation, the concept of receptive field in graph neural network is defined as the region on the graph that the target node receives messages from. With a

larger receptive field on graphs, representations of nodes are able to integrate more information.

4.3.2 Node embedding via Graph Neural Network

The core of our framework is how to aggregate information from a local graph region using neural networks. We know from the previous section how to predict the interaction score between user u and item i in SVD and SVD++. In SVD, the interaction score is directly calculated by the inner product of the latent factors of users and items. It is a special case of GCF without aggregating information from local graph regions. SVD++ incorporates the information from neighborhoods of users to its latent factors on the basis of SVD. This operation can be regarded as a process in which information propagation occurs from items to users, which can be formulated as $p_u + |N(u)|^{-1/2} \sum_{j \in N(u)} y_i$.

Based on this idea, we consider aggregating feature information iteratively from local graph neighborhoods, so that information from larger regions can be incorporated into latent factors. A node receives messages from its neighbor nodes and incorporates them into itself as new latent factors. The formulation can be expressed as:

$$h_i^{(k+1)} = h_i^k + |N(i)|^{-1/2} \sum_{j \in N(i)} h_j^k \tag{4.3}$$

where h_i^k is the latent factors of node i in layer k. Note that there are also some differences between one-layer GCF and SVD++. In SVD++, y_i are independent parameters, meaning that they are an indication of preference. However, in our method, y_i equals to q_i. Another difference is that messages are only passed from items to users in SVD++, but in our method, messages are passed to both sides.

In the above formula, we hypothesize that user latent factors and item latent factors are in the same feature space, and that the corresponding dimensions in both feature spaces have the same meaning, so their corresponding dimensions can be directly summed. However, latent factors in these two features space may not have the same meaning. Therefore, we concatenate the user latent factors with the item latent factors, then add a linear transform and use a nonlinear activation function to ensure they are transformed to the same feature space. The final formula is as follows:

$$h_i^{(k+1)} = \sigma(W^k \times concat(h_i^k,\ |N(i)|^{-1/2} \sum_{j \in N(i)} h_j^k)) \tag{4.4}$$

where W^k is the linear transformation matrix in layer k. Weights in the transformation matrix will be shared on all nodes. σ is the activation function such as $ReLU(\cdot) = max(0, \cdot)$.

4.3.3 Attention Mechanism

Attention mechanism has become one of the most important elements in many sequence-based tasks, because it allows for dealing with variable size inputs, focusing on the most relevant parts of the input to make decisions. In recommendation tasks, the node degree on the bipartite graph is variable. We introduce an attention-based architecture to handle this problem.

We add a measure of the importance of the neighbor nodes to each node to replace the fixed experienced number $|N(i)|^{-1/2}$, as shown in the following formula:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k \in N(i)} exp(e_{ik})} \tag{4.5}$$

where $e_{ij} = a \cdot concat(h_i, h_j)$ and $a \in R^{2F}$, F is the dimension of latent factors.

The message passing layer can be expressed as:

$$h_i^{(k+1)} = \sigma(W^k \times concat(h_i^k,\ |N(i)|^{-1/2} \sum_{j \in N(i)} \alpha_{ij} h_j^k)) \tag{4.6}$$

4.3.4 Model Training

As previously indicated, it is only positive classes in the data are observed in implicit feedback recommendation systems. The remaining data is a mixture of negative classes and missing values. In our method, we use item pairs as training data to optimize a pair-wise ranking list. The basic idea is to maximize the distance between a positive example and a negative example for each item pair in the training data. The loss function is as follows:

$$L = \max_{\theta} \sum_{(u,i,j) \in D_S} \ln \delta\left(\hat{y}_{ui} - \hat{y}_{uj}\right) - \lambda_\theta \|\theta\|_2 \tag{4.7}$$

where δ is the logistic sigmoid function, $\delta(x) = \frac{1}{1+exp^{-x}}$, θ is all the trainable parameters in our method, and D_S is the sampled data point, which means that $(user\ u,\ item\ i)$ is observed in the datasets and $(user\ u,\ item\ j)$ is the missing value. We assume that the user prefers the positive items over all other non-observed items. For items that both appear as positive classes, we cannot infer any preference. The same is true for two non-observed items.

Algorithm 4.1 details how optimization can be achieved by performing stochastic gradient descent. We first initialize the parameters with random values in the model. The parameters include the user latent factors p_u, the item latent factors q_i, weights of linear transform W^k and weights of attention mechanism a. Then we

compute the loss value for each example and apply the stochastic gradient descent strategy.

Algorithm 4.1: Optimization

htb
Input:
Set of training data $(u,\ i,\ j) \in D_S$;
Hyper-parameters: dimension of latent factors, number of hidden layers, λ_θ
Output:
An optimized model

Initial parameters in the model;
repeat
for each $(u,\ i,\ j) \in D_S$ **do**
for each layer in hidden layers **do**
for each node on the graph **do**
Compute forward propagation on the graph h_i^k;
end for
end for
Compute loss value for current example $L_{(u,\ i,\ ,\ j)}$;
Apply stochastic gradient descent to optimize parameters;
end for
until convergence

The process of generating recommendations is detailed in Algorithm 4.2. It shows how to obtain the ranking list of all the items about each user.

4.4 Experiments

4.4.1 Experimental Settings

Experiments are conducted on three datasets namely MovieLens 1M, MovieLens 10M and Taobao. The basic statistics are listed in Table 4.1. We select 60% of records as the training set. Some records contain explicit feedback such as ratings.

Algorithm 4.2: Generating recommendations

htb
Input:
User id: u;
optimized model;
Output:
Ranking list for user u

for each item $i \in \mathcal{V}$ **do**
Perform forward propagation to obtain the representation of user u and item i;
Compute score $\hat{y}_{ui}$;
end for
Ranking according to scores;
Recommend the top K items for user u;

As our focus is on the implicit feedback task, we remove the ratings from these datasets.

Table 4.1 Statistics of the two datasets

	users	items	feedback	sparsity
ML-1M	6,040	3706	939,809	0.9580
ML-10M	69,878	10,677	104,000,054	0.9865
Taobao	8,349	5,701	321,976	0.9932

MovieLens is a common benchmark dataset which consists of user ratings for items. Many versions have been released on the GroupLens website. We select MovieLens 1M (ML-1M) and MovieLens 10M (ML-10M) to evaluate our method.

Taobao is a dataset for competitively matching clothing on the Tianchi platform. It contains basic item data and data on the historical behavior of users. We use only the historical behavior data to make recommendations. We remove users with less than 10 items ($|N(u)| < 10$) and items with less than 20 users ($|N(i)| < 20$) from this dataset.

For the recommendation task with implicit feedback, we evaluate the performance of each method using the following metrics averaged over all users:

- Hit ratio at K (HR@K) that is equal to 1/K if the test item appears in the top K list of recommended items.
- Normalized Discounted Cumulative Gain (NDCG@K) favors higher ranks of the test item in the list of recommended items.

4.4.2 Performance Comparison

In this subsection, we compare the proposed model with the methods below.

- ItemPop: Always recommends the top-k most popular items to users.
- SVD: The most famous matrix factorization method which maps the interaction between users and items to a latent space of lower dimension.
- SVD++: The enhanced model of SVD.
- GCF-NA: The proposed model without attention mechanism.
- GCF-YA: The proposed model with attention mechanism.

To ensure all these experiments are evaluated equally, they are all conducted under the Bayesian personalized ranking optimization framework. The hyper-parameters for each baseline method are carefully chosen. The number of layers in both GCF-NA and GCF-YA is set to three to balance efficiency and effectiveness. The overall performance of the compared approaches is shown in Table 4.2.

Comparing all the methods on these four datasets, we make the following observations:

Table 4.2 Recommendation Performance in Terms of AUC and Diversity

		@1		@5		@10	
		HR	NDCG	HR	NDCG	HR	NDCG
ML-1M	*ItemPop*	0.1611	0.1308	0.3679	0.2727	0.5003	0.3899
	SVD	0.2465	0.2032	0.5885	0.3956	0.7510	0.4574
	SVD++	0.2522	0.2082	0.5972	0.4029	0.7690	0.4646
	GCF-NA	0.2591	0.2128	0.6029	0.4143	0.7656	0.4724
	GCF-YA	0.2763	0.2158	0.6150	0.4394	0.7818	0.4873
ML-10M	*ItemPop*	0.1427	0.1277	0.3485	0.2621	0.4507	0.3463
	SVD	0.2550	0.2090	0.5308	0.3439	0.7219	0.4077
	SVD++	0.2657	0.2130	0.5249	0.3575	0.7485	0.4249
	GCF-NA	0.2886	0.2279	0.5404	0.3782	0.7525	0.4432
	GCF-YA	0.2982	0.2293	0.5673	0.3893	0.7642	0.4677
Taobao	*ItemPop*	0.0191	0.0113	0.0354	0.0182	0.0542	0.0259
	SVD	0.0900	0.0699	0.2148	0.1317	0.3011	0.2048
	SVD++	0.1246	0.1051	0.2352	0.1405	0.3310	0.2200
	GCF-NA	0.1309	0.1150	0.2465	0.1558	0.3361	0.2254
	GCF-YA	0.1562	0.1264	0.2888	0.1819	0.3662	0.2491

- The collaborative filtering methods are more effective than ItemPop.
- The proposed GCF method has obvious advantages over the SVD method. This indicates that aggregating information on a larger regional scope on the graph is effective.
- It is evident that GCF-YA is 6% better than SVD on ML-1M, 14% on ML-10M, and 20% on the Taobao dataset, which means that the proposed method is more powerful in dealing with sparse data.
- Comparing GCF-YA and GCF-NA, it is clear that attention mechanism has certain advantages.

4.4.3 Discussion

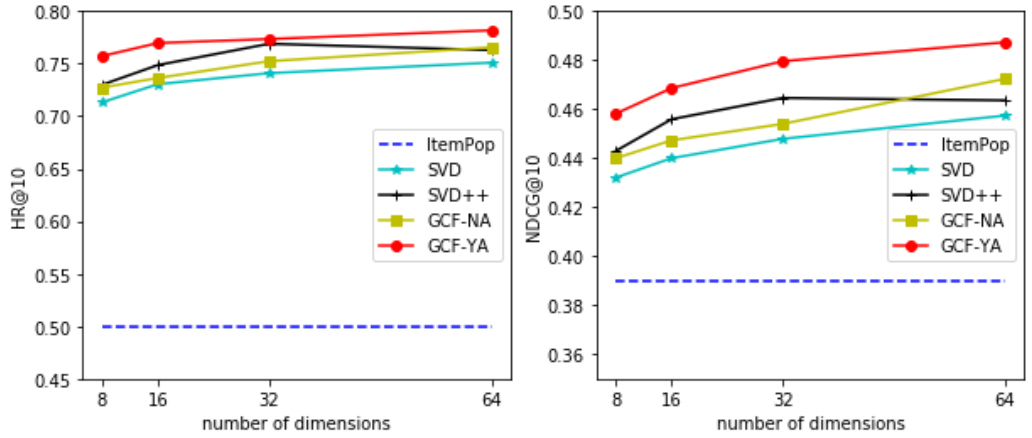


Figure 4.3 Performance of HR@10 and NDCG@10 with different numbers of latent factors on ML-1M.

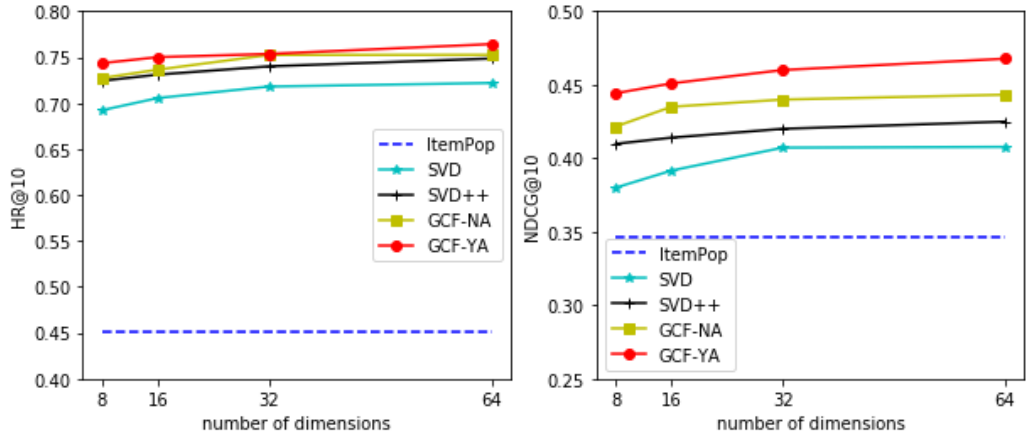


Figure 4.4 Performance of HR@10 and NDCG@10 with different numbers of latent factors on ML-10M.

The dimension analysis is shown in Figure 4.3, 4.4 and 4.5. We evaluate the performance with HR@10 and NDCG@10 on three datasets. The best dimension is set up to 64. It is clear that GCF-YA outperforms all the other models on all dimensionalities. The results also indicate that our method is not sensitive to dimensionality.

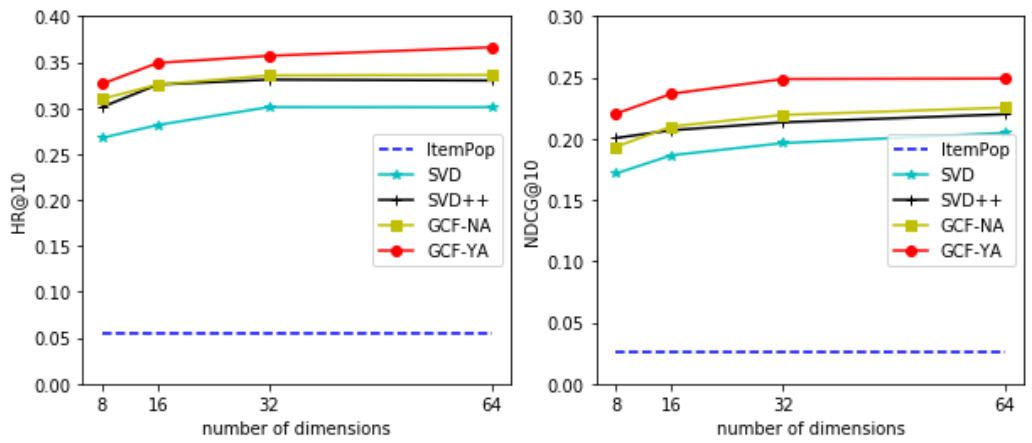


Figure 4.5 Performance of HR@10 and NDCG@10 with different numbers of latent factors on Taobao dataset.

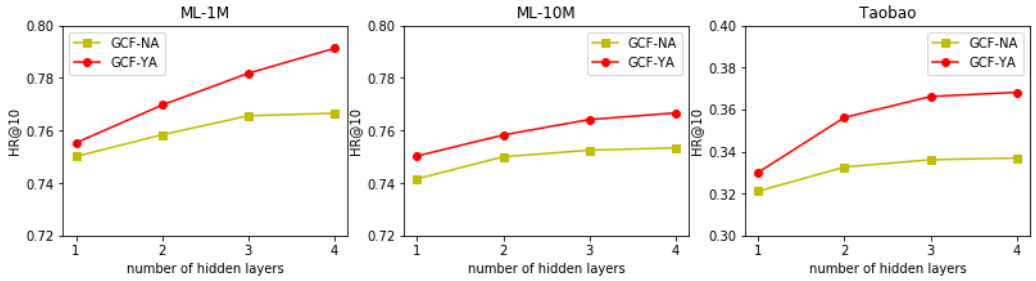


Figure 4.6 Performance of HR@10 with different numbers of hidden layers.

We also analyze the impact of the number of hidden layers in terms of HR@10 and NDCG@10. The results are shown in Figures 4.6 and 4.7 and show that GCF-YA achieves better results than GCF-NA, especially when the number of hidden layers increases.

4.5 Summary

In this study, we have introduced a general framework GCF and an information propagation-based graph neural network. GCF is a representation learning framework for learning a mapping that embeds users and items as points in a low-

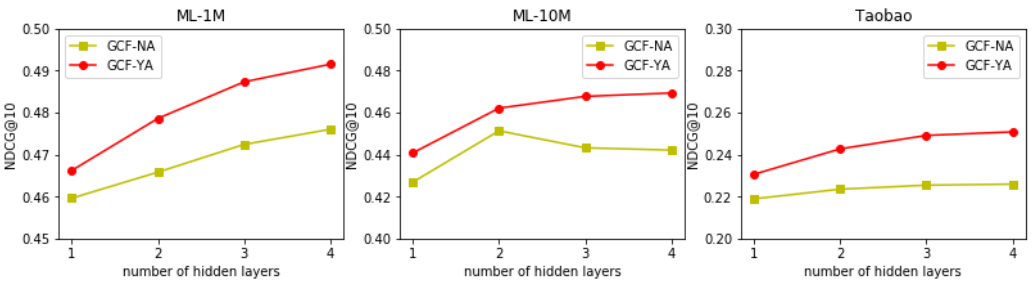


Figure 4.7 Performance of NDCG@10 with different numbers of hidden layers.

dimensional vector space with geometric relationships in the embedding space that reflect the preference relationship between users and items. To address the problem of variable size inputs for each node on a bipartite graph, we have proposed an attention-based information propagation method. The primary contribution of GCF is that a larger receptive field size can be used to obtain more information to support the users' decision process. We also proved that SVD and SVD++ can be expressed under GCF with node embedding via graph neural network. We conducted experiments on several real-world datasets to reveal the relationship between the number of iterations of information propagation and the recommendation performance. The experimental results show that the proposed method outperforms all the other models.

Chapter 5

Long- and Short-term User Interest Network for Personalized Recommendation

5.1 Introduction

Recommender systems have been studied by many researchers and companies as an effective way to alleviate the information overloading. Recommender systems learn diverse personalized interest from historical behaviors and select items that the user may like for target user in numerous of goods or services. It is well known that there are two types of user interest: long-term user interest and short-term user interest(Guo et al., 2019). The former refers to users' inherent and stable interest contained in the user's inherent attributes and historical behaviors. For example, the interest related to the users' gender and occupation are long-term interests. The short-term user interest convey users' purchasing intention in a relatively short

period. It is affected by incidentally transient events, such as the fashion trends change or different personal mood.

According to different assumptions of user interests, recommendation approaches can be classified into two categories: traditional recommendation approaches and sequential recommendation approaches. Traditional recommendation approaches mainly focuses on the long-term user interest. These methods treat all the user's historical behaviors as an unordered collection, and treat all the records in the collection equally. MF is one of the most famous methods in recommender systems(Rendle et al., 2012). By learning latent user factors and item factors directly, MF achieves good performance in many different scenarios. SVDFeature (Chen et al., 2012) incorporates side information into matrix factorization to predict the bias term and to reweight the matrix factorization. Wide&Deep(Cheng et al., 2016) considering low- and high-order feature interactions simultaneously brings additional improvement. However, these approaches cannot fully exploit contextual information which is important to identiy users' intention. These approaches also assumpt that user interests are stable, they cannot adaptive user's interests changes with time.

Different from traditional recommendation approaches, sequential recommendation provides an effective way to model the sequential relationship between historical behaviors of users and generate personalized recommendation lists for them. Sequential recommendation approaches assume that the more recent items in a sequence have a larger impact on the next item. FPMC(Rendle et al., 2010) uses the Markov chains to model the sequence data, so as to calculate the next click action through the state transition probability, and combines the matrix decomposition algorithm to learn the latent factors of users and items. In recent years, with

the development of deep learning, more and more sequential recommendation algorithms based on deep learning have been proposed. Caser(Tang and Wang, 2018) uses a convolutional neural networks to learn the sequential patterns. It effectively solves the problem that markov chain approaches failed to model union-level sequential patterns and did not allow skip behaviors in the item sequences. However, in the real-world recommender systems, the number of items is often very large, the number of states will increase exponentially with the number of items. Sequential recommendation approaches also cannot effectively use all the historical information of users, because it always relies on the recent user behavior data and ignores the long dependencies in long sequences.

Session-based recommendation approaches is a subarea of sequential recommendation. A session is a list of user behaviors that occur within a given time frame. These methods divide the user's complete historical behavior sequence into multiple sessions which consist of consequent items in a short period. In this setting, the recommender systems make recommendations based only on the behavior of users in the current browsing session. Session-based recommendation approaches pay more attention to the short-term user interest, not only because the user ID cannot be obtained in many scenarios, but also by analyzing and using the user's short-term interests, the user's specific needs in the current context can be better obtained. Benefit from the sequential modeling capability of RNN, **G**ated **R**ecurrent **U**nit (GRU) is widely used in session-based recommender systems. GRU4REC (Hidasi et al., 2015) is the earliest attempt to apply RNN with GRU to the session-based recommender system. They treat items in the session as a sequence, and introduce session-parallel mini-batch, mini-batch based output sampling and ranking loss function into session modeling and achieve better performance than conventional

methods. (Wu et al., 2019) are the first to introduce GCN into recommender systems. They use GNN to capture complex transition relationships in session sequences. Although these methods have achieved better results in session-based recommendation tasks, they ignore the long-term user interests which are able to contribute more to learn users' interests exactly.

In order to solve the problems that the traditional recommendation approaches cannot dynamically adapt to the dynamic changes of user interests over time and the sequential recommendation approaches cannot effectively utilize the user's long-term interest, there is also a small amount of work that takes into account the user's long-term interests and short-term interests. For example, the literature (Devooght and Bersini, 2017) comprehensively considers the user's long- and short-term interests in recommender systems, and proposes a method based on recurrent neural networks which allow recommender systems to manually control the impact of long- and short-term interests to generate recommendation results. However they did not take into account the long-term interests, beacause not all user historical behaviors is used as the context of the current session which resulting in the long-term interests can not have a sufficient beneficial impact on the current recommendation. (Villatel et al., 2018) uses recurrent neural networks to model the sequential data in recommender systems. As a study of long- and short-term interests, the author only compared the effects of different sequence lengths in their article, and not really use all the user's historical behaviors to learn long-term user interests.

Based on the abovementioned analysis, we propose a long- and short-term user interest recommendation method that exploits long-term user interest and short-term user interest to perform more precise recommendation. Specifically, inspired by

the GNN method based on collaborative filtering, we propose L-UIN to learn the latent real-value low-dimension feature representations of long-term user interest. We incorporate side information the complete user's historical behaviors and side information into this network, such as the brands, price and categories of goods. In addition to L-UIN, the latent feature representation of short-term user interests, which refers to the temporary and dynamic user intention, are learnt by the proposed S-UIN. Especially, S-UIN can learn the sequential relationship from the session sequence via network embedding and attention based recurrent neural network and is able to capture the user's intention according to the browsing sequence. We use common embedding layers for both L-UIN and S-UIN, which brings two important benefits: (1) knowledge could be shared between L-UIN and S-IN; (2) it can reduce the number of parameters, which speed up training process and reduce the risk of overfitting. Finally, we aggregate the long- and short-term user interests and generate recommendation for users.

The main contributions of this chapter are summarised as follows:

1. Presenting L-UIN on the basis of graph convolutional neural network to learn long-term user interest from rich historical behaviors and side information.

2. Presenting S-UIN to learn user's dynamic interest and intention from the context. Based on the dynamic and temporary characteristics of the short-term user interest, a recurrent neural network was built for quickly obtaining the user's intention and generating adaptive recommendation according to the user's surfing process.

3. Proposing a method called LSREC to combine long-term user interest and short-term user interest with sharing the feature embedding between the L-UIN and S-UIN component.

4. Comprehensive experiments are conducted on real-world datasets, which demonstrate that our method outperforms other state-of-the-art baselines.

5.2 Preliminaries and Problem Formulation

Formally, we define the set of all users as $U = \{u_1,\ u_2, \ldots, u_N\}$, the set of items as $X = \{x_1, x_2, \ldots, x_M\}$, where N and M represent the number of unique users and items in the dataset respectively. Suppose we know that the set of session sequences is $S = \{s_1, s_2, \ldots, s_K\}$, in which a session sequence of interactive actions are denoted as $S_i = \{x_1, x_2, \ldots, x_n\}$, where K represents the number of session sequences in the dataset and $x_i \in X$ denotes the ID of the i-th item that the uesr interacted with in the current session. From this, all sessions that the user u_i interacted in the dataset can be represented by $S_{u_i} \subset S$, where $u_i \in U$ denotes the ID of the i-th user in the dataset. It is noteworthy that there are some anonymous sessions that do not belong to any user in some dataset. The symbols we used to introduce our model are listed in Table 5.1.

Usually, the objective of recommendation task based on short- and long-term user interest is to predict what a user would like to click next when the current session sequence and the user's side information are given. The formulation of the recommendation task in this chapter is that we build a model for any given session sequence s_i, user historical behaviors S_{u_j} and other auxiliary information $aux(u_j)$ of u_j and items in the dataset $aux(X)$, we get the output

Table 5.1 Notations

Notations	Description
U	set of users
X	set of items
S	set of sessions
S_{u_i}	set of sessions which are generated by user u_i
$aux(u_i)$	side information of user u_i
$aux(X)$	side information of all items
y_{u_j,s_i}	ranking metrics of user u_j for all items on state of s_i
$p_{u_i}^{long}$	latent vector of long-term perference of user u_i
$p_{s_j}^{short}$	latent vector of short-term perference of session s_j

of $y_{u_j,s_i} = f(s_i, S_{u_j}, aux(u_j), aux(X))$, where $y = [y_1, y_2, ..., y_M]$. We view y_{u_j,s_i} as the ranking list over all the next items that can occur in that session, where y_j corresponds to the recommendation score of item j. Since a recommender system typically needs to make more than one recommendations for the user, thus the top-k items in y are recommended.

Speicifically, in the process of ranking predicting, we aim to learn a user's latent vectors from the current session sequence, the user's attribute information and historical behaviors. In detail, given a user $u_i \in U$, we use S_{u_i}, $aux(u_i)$ and $aux(X)$ to learn the long-term user interest vector $p_{u_i}^{long}$. Then, we aim to use s_j and $aux(X)$ to predict the short-term user interest as a latent vector $p_{s_j}^{short}$. Finally, we aggregate the short- and long-term uesr interest to generate the final prediction y.

5.3 Methodology

The framework of LSRec is illustrated in Figure 5.1. As we see, it can be divided into three parts: long-term user interest learning, short-term user interest learning, and the fusion of long- and short-term user interest. The lower right part of the

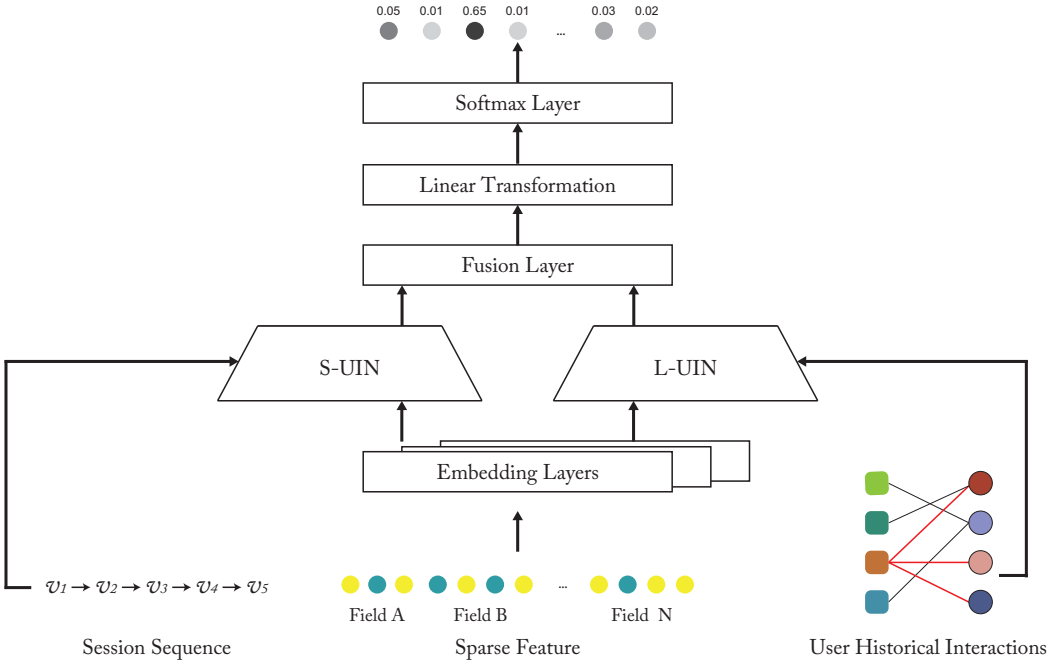


Figure 5.1 Overall framework of the proposed LSRec method.

framework is L-UIN. It is a GNN model. we feed the bipartite graph of user historical behaviors and entity embeddings (i.e., user ID, item ID and item profiles) into the network and it will generate a latent vector to represent the long-term user interest. The details of this part will be introduced in Section 5.3.1. Then, we further learn the short-term user interest shown in the lower left part of the framework, which takes the item sequences of session and entity embeddings as the input. We will introduce this part in Section 5.3.2. To enable knowledge sharing between L-UIN and S-UIN, we use common embedding layers to represent discrete features in the datasets. Finally, to obtaion the complete user representation, we combine the short- and long-term user interest into a final user latent vector and generate recommendation list according to this vector. This part will be introduced in Section 5.3.3.

5.3.1 Long-term User Interest Network

In this section, we mainly introduce L-UIN in detail. The long-term user interest reflects the fairly stable and static interests of users based on their historical behaviors and attributes. Though many existing works focus on long-term user interest, learning long-term user interest is still facing challenges: most of these methods have lower performance in cold-start situations and it is difficult for us to provide accurate recommendations for inactive users. Most existing approaches considering short- and long-term user interest model the long-term uesr interest by using sequential models, because these methods are similar with short-term part. However, as we all know, the users' history is not continuous in time and there is no sequential dependency between items. Therefore, we use the collaborative relationship between items to build a graph neural network to learn the long-term user interest.

Collaborative filtering methods assume that people who have the same interest in the past will also have the same interest in the future. In conventional collaborative filtering methods, recommendations are generated according to similar users or neighbor items of the target user. And the similarity between each user are computed accrodingto the historical behaviors, e.g. the users who rated many common items or the user has rated a series of items. According to the method proposed in (Koren, 2008), the long-term user intrerest can be represented by aggregating the neighbor items. It can be formulated as:

$$p_u^{long} = \sum_{i \in B(u)} x_i \tag{5.1}$$

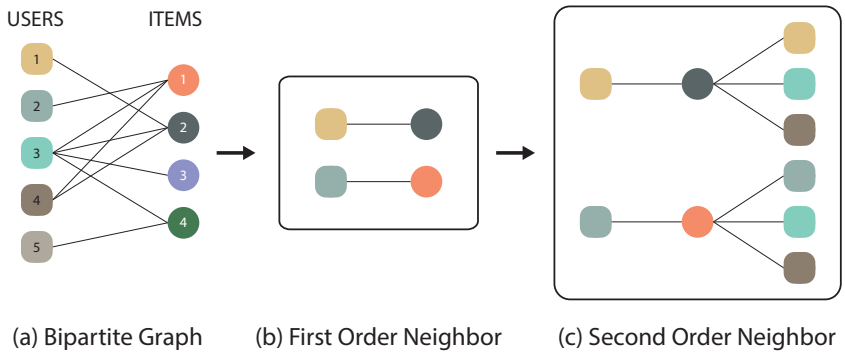


Figure 5.2 Illustration of long-term interest network

where x_i is the latent vector of item i and $B(u)$ is the neighbor items of user u_i. In real scenarios, the latent vector often consist of variety property of items, such as ID, prices and categories. However, we are unable to get enough information to support the prediction of long-term user interest in cold-start situation. Because there are only few neighbor items in many cases.As shown in Figure 5.2 (b), imagine that we aims to generate a recommendation list for user u_1 whose neighbor only has x_2. Then we can compute the similarity between user u_1 and user u_2 according to the neighbors they shared by using Equation 5.1. The similarity will be 0 because they have no common neighbor. To alleviate the problem of insufficient information, we introduce a graph neural network to utilize local structure information and neighbor information. Specifically, we introduce high-order neighbors to support the recommendation decision making process. As shown in Figure 5.2 (c), we can find the common second-order neighbors u_3 and u_4 between the user u_1 and the user u_2 through the second-order adjacency relationship, so as to obtain the reasonable similarity metric between the user u_1 and the user u_2. Since the graph is an asymmetric non-Euclidean space, it is often necessary to search the solution space on the whole graph. By introducing the graph convolutional neural network, information can be aggregated locally at relatively low computational cost. The

spatial graph convolution function can then be written as:

$$p_u^{l+1} = W * \left[p_u^l; \sum_{v \in N(u)} \alpha_{u,v} p_v^l \right] \tag{5.2}$$

where p_u^l is the feature expression of the node u at the l level, assuming its dimension is d. W is a linear transformation that maps the aggregated features from the $2d$ dimension back to the d dimension. $\alpha_{u,v}$ is used to select which neighbor information is more important. The formulations can be expressed as follows:

$$\alpha_{u,v} = \frac{e^{\left(p_u^l\right)^T \; p_v^l}}{\sum_{x \in N(u)} e^{\left(p_u^l\right)^T \; p_x^l}} \tag{5.3}$$

As in convolution neural network, we hope to increase the receptive field with more graph convolutional neural network layers. However, the high-level abstraction often leads to the loss of details in the original features which results the oversmoothing problem. Therefore, we retain the latent feature of each layer in the graph convolutional neural network to improve the expressive ability of the proposed method. To this end, for k layers graph convolutional network, we consider the following long-term user interest representation:

$$p_u^{long} = \left[p_u^0; p_u^1; \ldots; p_u^k \right] \tag{5.4}$$

5.3.2 Short-term User Interest Network

A user usually show a clear intention in one session, and the intention often change sharply when the user start a new session. Thus, for the purpose of learning session-level user interest from sequential behaviors, we establish a short-term user interest

network based on the basis of RNN. RNNs have been devised to model sequential data in many real-world scenarios. There are many different variants of RNNs (e.g., LSTM and GRU). For the balance between efficiency and performance, we take the GRU variant to model the dependency between sequential behaviors. The GRU variant reduced the vanishing gradients problem of vanilla RNN. The input of the network is ordered behaviors in which the item IDs have been represented by the common embedding layers. The formulations of GRU can be expressed as follows:

$$R_t = \sigma(X_t W_{xr} + H_{t-1} W_{hr} + b_r) \tag{5.5}$$

$$Z_t = \sigma\left(X_t W_{xz} + H_{t-1} W_{hz} + b_z\right) \tag{5.6}$$

$$\widetilde{H}_t = \tanh\left(X_t W xh + \left(R_t \odot H_{t-1}\right) W_{hh} + b_h\right) \tag{5.7}$$

$$H_t = Z_t \odot H_{t-1} + (1 - Z_t) \odot \widetilde{H} \tag{5.8}$$

$$Y_t = H_t W_{hq} + b_q \tag{5.9}$$

where X_t is the input at the time t, which is the representation of a commodity in the user's conversation sequence. W is the weight matrix in the network, and b is the bias of the neurons in the network.

However, the short-term user interest which only captures the users current state cannot represent the browsing intention of current session. For example, a customer who intends to buy a computer is attracted by a advertising of mouse and

clicks the detail page of the mouse. In this case, we cannot just predict that the user's intention is to by a mouse according to the last state. Thus, the short-term user interest should contains two level: the last state representation and the user intention representation. To obtain the user intention of the whole session, interest state at each time step is aggregated.

As we all know, not all user's behaviors are strictly dependent on each adjacent behavior. With the help of attention mechanism, we can arrange different attention score to reflect the relationship between user's behavior and his/her intention. The user intention α_u can be formulated as:

$$\alpha_u = \frac{1}{t}\sum_{i=1}^{t} y_i (W_{at} \sum_{j=1}^{t} y_j)^T y_i \tag{5.10}$$

where $W_{at} \in R^{d \times d}$ is used to transform the average of hidden states into a latent space, d is the dimension of hidden state; t is the length of the session.

Finally, we combine the user's current state and user intention by concatenate operation to represent the short-term user interest:

$$p_u^{short} = [y_t; \alpha_u] \tag{5.11}$$

5.3.3 Long- and short-term interest fusion and recommendation

The user's long- and short-term interests can be represented by two vectors respectively with L-UIN and S-UIN. To obtain the user's universal interest with long- and short-term interest, we concatenate these two feature vectors, and then a fully connected layer is used to learn the final representation of the user. The

formulations can be expressedas follows:

$$p_u = W * \left[p_u^{short}; p_u^{long} \right] \tag{5.12}$$

Finally, through the learned user's interest and item feature, the user's preference on each item can be predicted as following:

$$\hat{y}_i = \frac{e^{p_u^T x_i}}{\sum_{x \in I} e^{p_u^T x_x}} \tag{5.13}$$

5.3.4 Loss function

The problem of click prediction can be regarded as a multi-classification problem. In this chapter, we use the softmax cross-entropy loss to optimize the proposed model. It can be formulated as:

$$J = -\sum_{u \in U} \sum_{i=1}^{n} y_i \log(\widehat{y_i}) + (1 - y_i) \log(1 - \widehat{y_i}) + \lambda |\theta|^2 \tag{5.14}$$

where θ is all the parameters in the model.

5.4 Experiments

This section presents the experimental settings, results and the related analysis. The datasets will be introduced first, followed by the evaluation metrics. Then we give the baseline methods to be compaired with our method. Finally, we present the results of the empirical experiments with a comprehensive analysis to the results.

Table 5.2 Statistics of datasets used in the experiments

	Diginetica	Retailrocket
# of clicks	999,198	1,270,194
# of training sessions	181,165	330,885
# of test sessions	24,532	40,039
# of items	44,526	79,851
Average length	4.86	3.42

5.4.1 Dataset and Data Preparation

We study the effectiveness of the proposed approach LSRec on two real e-commerce datasets, i.e., Diginetica[1] and Retailrocket[2]. Diginetica comes from CIKM Cup 2016. We used the released transaction data and item side information in this study. Retailrocket dataset is published by a personalized e-commerce company, which contains six months of user browsing activities. We also used the released user behaviors data and item properties in the following experiments.

To filter noisy data, we filter out all session sequences with a length shorter than 2 items and items appearing less than 5 times(Li et al., 2017). In addition, similar to (Tan et al., 2016), we use the data augmentation method to generate sequences and corresponding labels by splitting the input session. Then, a session sequence of length n is divided into $n-1$ sub-session sequences. For side information, we discretize all continuous variables to improve the generalization and stability. The data statistics are shown in Table 5.2.

[1] http://cikm2016.cs.iupui.edu/cikm-cup/
[2] https://www.kaggle.com/retailrocket/ecommerce-dataset

5.4.2 Evaluation metrics

In most recommendation scenarios, the number of items that can be recommended is limited by certain factors. Therefore, the items that meet user's interests should appear in the limited number of items listed in the recommendation list. To evaluate the quality of the recommendation list generated by these methods, we adopt two common metrics, i.e., Predictive accuracy (P@K) and Mean Reciprocal Rank (MRR@K). The former one is an evaluation of unranked retrieval results, while the latter one is evaluations of ranked lists. Here, we consider Top-K (K = 5, 10) for recommendation.

P@K: The P@K score is widely used as a measure to evaluate the quality of the recommendation lists. P@K represents the proportion of items that should be recommended in a top K products in a ranking list. The calculation process is as the following equation:

$$P@K = \frac{n_{hit}}{N} \tag{5.15}$$

where N denotes the number of items in the test set, n_{hit} represents the number of items that appears in the recommendation list. We calculate P@K of each user, and then take the average as the P@K of the method.

MRR@K: MRR@K is the average of the inverse of the ground-truth item ranking. If the rank of an item is greater than K, the value is set to 0. This indicator takes the position of the item in the recommendation list into account, and is usually important in some order-sensitive tasks. The following equation shows its calculation process:

$$MRR@K = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{Rank_i} \tag{5.16}$$

where $Rank_i$ is the rank of the first item in the recommendation list that appears in the test result for the i-th user, and $|Q|$ represents the number of users.

5.4.3 Baseline Methods

The evaluation the proposed method, we compare our method LSRec with the following methods. Our method considers long- and short-term user preferences simultaneously, and there is no other methods consider both of them. These methods are divided into two parts, the methods focusing on long-term user preferences and the methods focusing on short-term user interests.

The baseline methods focusing on long-term user interests are as follows:

- POP: A ranking algorithm that ranks the items according to the popularity and recommend the top K items to users. Although this is a simple method, it is still effective in some scenarios and is widely used as a benchmark in recommender systems.
- Item-KNN: A baseline method that recommends items similar to the candidate item based on the cosine similarity.
- BPR-MF: BPR-MF proposed a pairwise ranking objective function to model ranking relationship between implicit feedbacks.
- GC-MC: GC-MC view matrix completion as link prediction on interaction graph and introduce a graph auto-encoder framework for the matrix completion task in recommender systems.

- FPMC: This is a method that combines Markov chain model and matrix factorization for the next-basket recommendation. The authors assume that users could continuously review two closely related items.

The baseline methods focusing on short-term user interests are as follows:

- S-POP: This method recommends the most popular products for the current session. Compared with the POP method, the recommendation list changes during the session gains more items.

- GRU4REC: This method uses recurrent neural network to model user's behavior sequence and generate session-based recommendations. Especially, a session-parallel min-batch traing process is declared to model user action sequences.

- NARM: This model combines the attention mechanism with RNN, while capturing the sequential behavior characteristics and main purpose of users.

- STAMP: STAMP focuses on strengthening the influence of user's recent behaviors when modeling long-term behaviors. It not only considers the general interest from long-term historical behavior, but also considers the user's last click to mine the immediate interest.

- GC-SAN: It is a graph contextualize self-attention network based on graph neural network. GC-SAN dynamically construct a graph structure for session sequences and capture rich local dependencies via graph neural network and long-range dependencies in sessions is learnt by applying the self-attention mechanism.

5.4.4 Experimental Setup

The parameters used in the model are set as follows. For all datasets, the dimensions of the embedding vector are set to $d = 100$. All parameter matrices in the model are initialized using a Gaussian distribution with a mean of 0 and a standard deviation of 0.1. The mini-batch Adam optimizer is exerted to optimize these parameters, where the initial learning rate is set to 0.001. In addition, the training batch size and L2 penalty are set to 128 and 10^{-5} respectively. We select hyperparameters based on the validation set, which is a 10% random subset of the training set.

5.4.5 Comparison with baseline methods

To demonstrate the performance of the proposed model, we compare CaSe4SR with the other state-of-art methods, which include methods considering short-term user interest and methods considering long-term user interest. We report the comparison results of P@K, MRR@K and NDCG@K. For anonymous sessions in Diginetica dataset, the long-term user interest is set to the average user interest of the whole dataset. The specific comparison results are shown in Table 5.3. In addition, we will give a detailed analysis of the results.

Table 5.3 The performance of different methods on the two datasets.

	Diginetica				Retailrocket			
	P@5	P@10	MRR@5	MRR@10	P@5	P@10	MRR@5	MRR@10
POP	0.0025	0.0072	0.0017	0.0024	0.0111	0.0195	0.0074	0.0081
BPR-MF	0.0917	0.1210	0.0623	0.0801	0.2002	0.2501	0.1251	0.1312
Item-KNN	0.1283	0.1818	0.0667	0.0826	0.1857	0.2212	0.1100	0.1225
GC-MC	0.1649	0.2099	0.0731	0.0892	0.1923	0.2282	0.1126	0.1148
FPMC	0.1792	0.2193	0.0801	0.0953	0.1944	0.2314	0.1127	0.1153
S-POP	0.0130	0.0229	0.0100	0.0207	0.0109	0.0193	0.0077	0.0119
GRU4REC	0.2367	0.3151	0.1186	0.1348	0.2111	0.2750	0.1215	0.1367
NARM	0.3183	0.4668	0.2301	0.2290	0.3096	0.3587	0.1858	0.2044
STAMP	0.3420	0.4809	0.2313	0.2382	0.3286	0.3842	0.2190	0.2418
GC-SAN	0.3473	0.4927	0.2332	0.2403	0.3325	0.4011	0.2237	0.2491
LSRec	0.3691	0.5148	0.2574	0.2629	0.3861	0.5222	0.2778	0.3097

First, it can be seen that the recommend performance of all long-term user interest methods on Diginetica is lower than Retailrocket dataset. This is mainly because about 70 percent of sessions are anonymous sessions. We cannot obtain enough personalized information from these anonymous sessions. Moreover, it can be found from the experimental results in Table 5.3 that the recommend performance of all short-term user interest methods on Diginetica is higher than Retailrocket dataset. This is mainly because the average length of sessions in the former dataset is longer than the latter one. LSRec achieves the better performance on Retailrocket than Diginetica. We conjecture the main reason is that LSRec is able to utilize the whole user historical behaviors information of Retailrocket, and there are too many anonymous sessions which leads to the lower performance.

Second, the most traditional and the simplest algorithms (i.e., POP and S-POP) has the most unfavorable performance on both datasets. This is because it only considers the popularity of items and does not consider the user's personalized interactive behaviors. The significant gap between long-term user interest and short-term user interest is due to the short-term user interest model is more adaptive for recommendation task.

Finally, our proposed model LSRec outperforms all baseline methods. Compared with the best long-term user interest method in the results, the performance of LSRec is improved by about 100% in terms of P@K and MRR@K on both dataset. This confirms the necessity of considering sequential relationship between historical behaviors. Compared with the best short-term user interest method, the performance of our proposed method increased by about 5% on Diginetica dataset and about 20% on Retailrocket dataset. This is mainly because all users in Retailrocket dataset is nonanonymous and we can utilize more global information

about users. It indicates that LSRec can achieve better results compared with other methods that do not consider long-term user interest.

5.4.6 Components Analysis

We also conduct extensive experiments to show the effect of different components in the proposed method. In order to show the effectiveness of the long-term user interest network, we compared the performance of nonanonymous and anonymous users in Diginetica dataset. We also proved the effectiveness of the graph convolutional neural network component which was introduced in the long-term user interest network. In order to learn the impact of the dimension of latent vector, we conduct experiments on different dimension of latent vector. We also conduct experiments of with side information to show the impact of auxiliary information. Finally, we illustrated the learning curves on the two datasets.

1) Effects of L-UIN: To evaluate the efeectiveness of L-UIN, we divide Diginetica dataset into two parts: anonymous and nonanonymous. Table 5.4 shows the performance of P@5, P@10, MRR@5 and MRR@10 on LSRec. Compared to results on anonymous users, the results on nonanonymous users obtains obvious improvement. This shows that considering the long-term user interest in recommender systems can significantly improve the quality of the recommendation list.

Table 5.4 The performance of nonanonymous and anonymous users

	P@5	P@10	MRR@5	MRR@10
Nonanonymous	0.3437	0.4872	0.2312	0.2375
Anonymous	0.3771	0.5267	0.2605	0.2764

2) Effects of the number of Graph Convolutional Layers: We now explore how the number of graph convolutional layers affects the preformance of LSREC. It can be found from Figure 5.3 that the performance of LSREC increase as the number of graph convolutional layers increase. It indicates that the aggregate strategy in L-UIN is able to alleviate the over smoothing problem. However, more graph convolutional layers in L-UIN will result a high computational complexity. We can find that the gradient of performance cureve decreases with the number of layers increases. Therefore, to balence between accuracy and runtime, we choose three layers in other experiments in this section.

3) Effects of dimension: The dimension of the latent vector of user interests decides the fitting and modeling ability of the proposed method. Generally, LSRec can depict more useful information with a higher dimension and have better performance. However, the higher dimension may cause overfitting problem limited by optimization techniques. Therefore, we evaluate the proposed method on different dimension which increase from 10 to 200. We assume that the dimension in different part (e.g. ID embedding, latent vector of short- and long-term user interest p_{short} and p_{long}) is equal in our experiments. As shown in Figure 5.4, the performance in terms of P@K and MRR@K tends to increase and then stabilize as the dimension increase. The reason is a higher dimensional is able to incorporate more information and 100 dimension is enough to represent the knowledge in the dataset effectively.

4) Effects of side information: To compare the proposed approach with the baseline methods, we only consider the ID of users and items to generate recommendations. However, the proposed method LSRec is able to incorporate a variety of side information into the decision making process. Note that the side

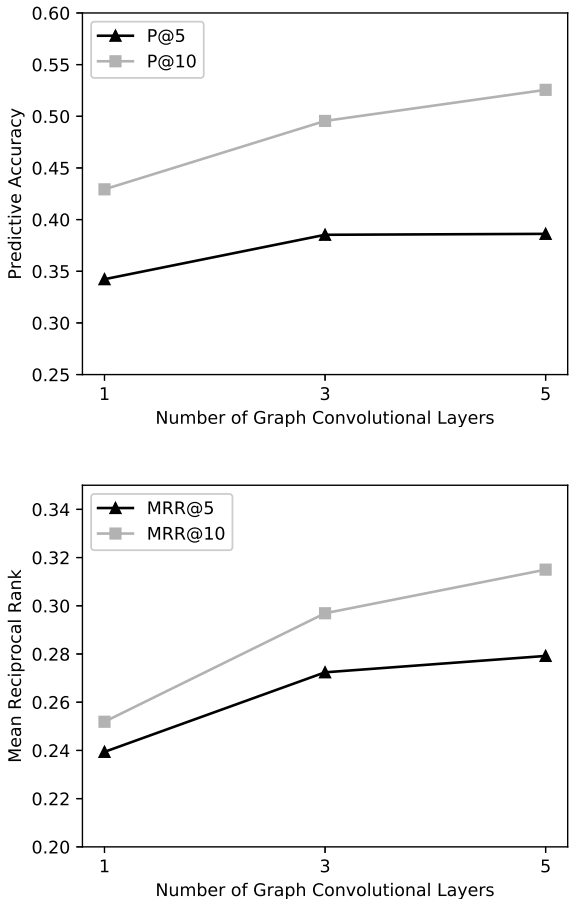


Figure 5.3 Performance comparison with different number of Graph Convolutional Layers.

information of the item in both datasets are concatenated to represent itself in the following experiments. As shown in Table 5.5, the performance of LSRec with side information outperforms LSRec without side information in two datasets.

5) Effects of data sparsity: We now explore how the sparsity of data influences the performance of the proposed method LSRec as well as the baseline methods. Science there is anonymous users in Diginetica datasets, experiments are conducted on Retailrocket dataset for fair evaluation. The concept of sparsity is different in long-term user interest and short-term user interest. In long term user interest

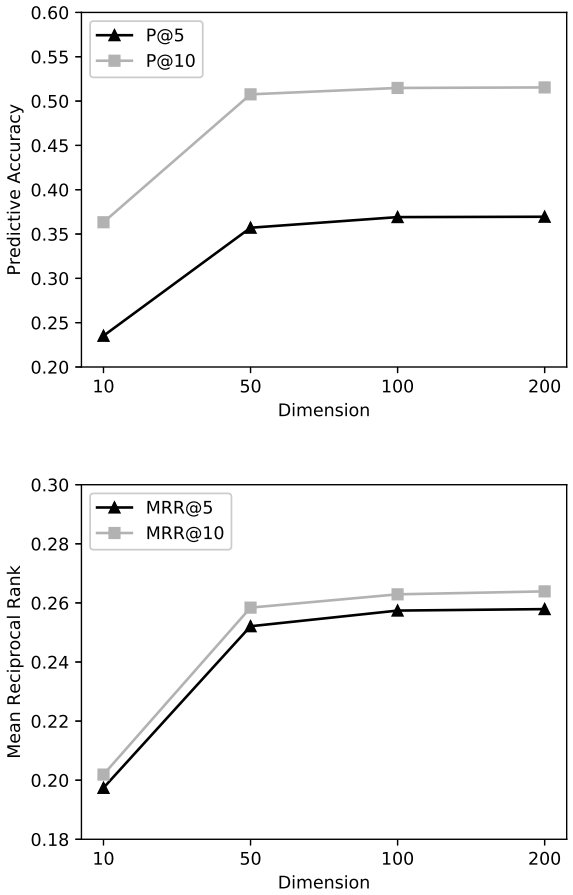


Figure 5.4 Experimental results of the dimension's effects on Diginetica dataset.

Table 5.5 Experimental results of LSREC with side information.

		P@5	P@10	MRR@5	MRR@10
Diginetica	with feat	0.3437	0.4872	0.2312	0.2375
	without feat	0.3437	0.4872	0.2312	0.2375
Retailrocket	with feat	0.3771	0.5267	0.2605	0.2764
	without feat	0.3771	0.5267	0.2605	0.2764

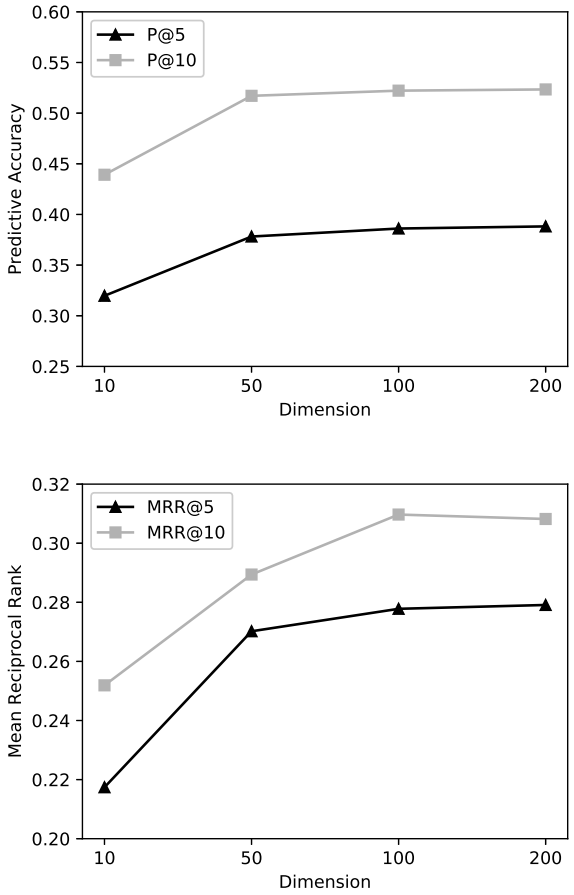


Figure 5.5 Experimental results of the dimension's effects on Retailrocket dataset.

scenario, sparsity refers to the proportion between the observed entries and unobserved entries in the user-item interaction matrix. Specifically, we construct the datasets via filtering users with low click frequency. The results are shown in Figure 5.6, and LSRec outperforms all baselines. The reason is that LSRec with higher oreder neighbors is able to utilize more information in histroical behaviors. The concept of sparsity in short-term user interest scenario is the average length of sessions. Because we can obtain more information from a long session sequence. The dataset of different sparsity in this scenario are generated via filtering short sessions. The results are given in Figure 5.7. The proposed approach LSRec outperforms all baselines in variety sparsity, which shows that LSRec is able to model short-term user interest with different data sparsity effectively.

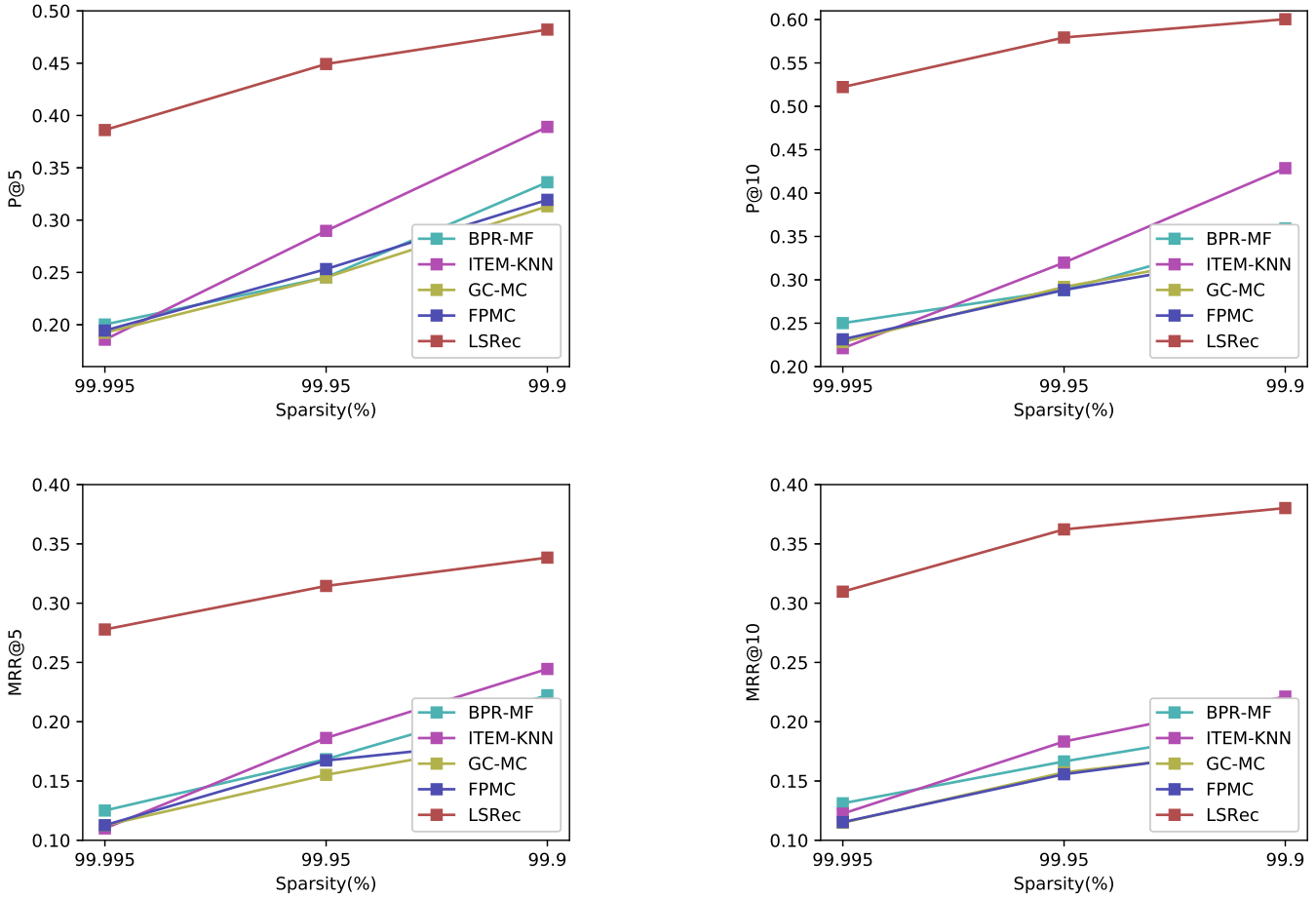


Figure 5.6 Performance over Retailrocket dataset with a different user-item interaction sparsity.

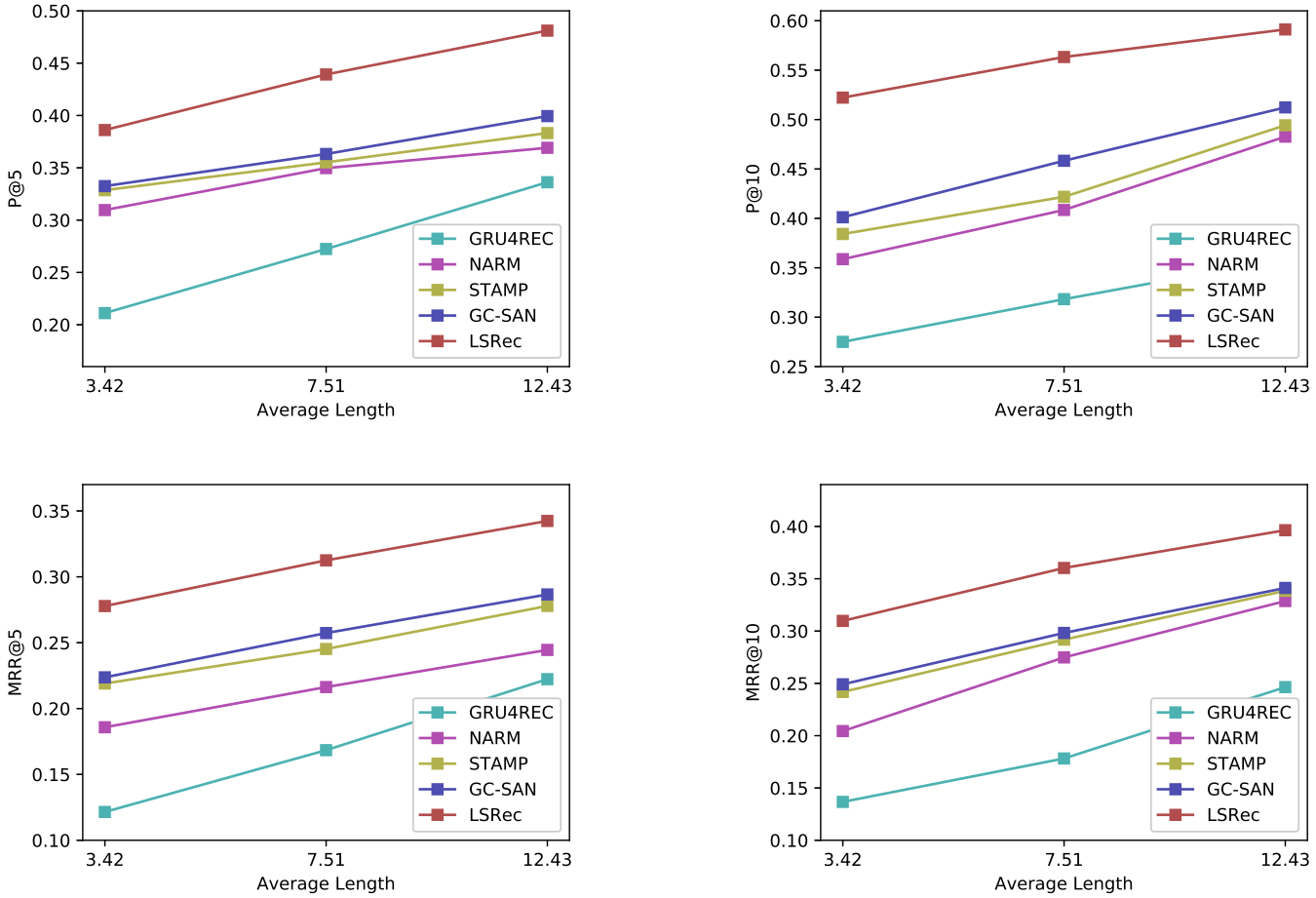


Figure 5.7 Performance over Retailrocket dataset with a different average length of sessions.

5.5 Summary

In this aritcle, we present a recommendation method LSRec based on long- and short-term user interest. Specifically, the proposed method is composed of three components: L-UIN for learning long-term user interest from historical behaviors and user profiles, S-UIN for modeling short-term uesr interest from sesssion sequence, and combination for recommendation generating. We particularly devise a graph convolutional neural network to enrich the user's local sturcture information by considering the high order neighbors in L-UIN. We have further developed an attention based RNN to learn short-term user interest in S-UIN. The experimental results on two datasets have demonstrated the superiority of LSREC over methods considering short-term user interest and long-term interest.

Chapter 6

RsyGAN: Generative Adversarial Network for Recommender Systems

6.1 Introduction

Recommender systems have become increasingly important in recent years due to the problem of information overload in e-commerce (Lu et al., 2015). The use of recommender systems allows individual searches to be more effective by filtering information. Many companies are also interested in using recommender systems to target their customers and recommend products. Recommender systems model the preferences of users through their click history, purchase records or list of favourites. The recommendation task is to predict missing user-item preferences given the observations of these historic records (Bobadilla et al., 2013).

Existing methods for recommender systems can be divided into three classes: content-based methods, CF methods, and hybrid methods. Many recommender systems use collaborative filtering methods to make recommendations (Georgiev

and Nakov, 2013). The most successful CF methods try to learn latent factors according to user-item interactions such as user-item rating or user purchase history (Su and Khoshgoftaar, 2009).

A severe problem with CF methods is that it is difficult to train the model with sparse datasets. The collaborative filtering approaches, especially matrix factorization methods, rely on factorizing the user-item matrix into two latent factor matrices to represent users and items. However, the factorization could be very non-robust when the user-item matrix is very sparse. It always causes a lower quality local optimum in the experiments.

Many works attempted to address this problem. Several authors have merged models to obtain more robust results. (Koren, 2008) combined latent factor models and neighbourhood models to build a more accurate combined model. (Sedhain et al., 2015) employed an autoencoder model to learn latent user preferences. These methods try to learn the latent factors through user-item interactions. But these methods cannot converge to an optimal solution because of the severe sparsity of the dataset, which results in an inability to provide ideal recommendation results.

There are also some methods use auxiliary information. Item content information and an item-tag matrix are combined in collaborative topic regression to address the sparsity problem in (Wang et al., 2013). A hierarchical Bayesian model has been proposed to address the auxiliary information sparse problem (Wang et al., 2015). However, auxiliary information is unavailable in some scenarios.

In this chapter, we develop a generative network and a discriminative network inspired by generative adversarial networks (Goodfellow et al., 2014) to train a property model for recommender systems. The generative network is able to generate the missing preferences for users and the discriminative network is

established to evaluate the generative network and guide training process. The model is trained using an adversarial training strategy.

In the experiments, our model demonstrates significant improvements in performance on common datasets such as movieLens (Harper and Konstan, 2015) and Taobao. The main contributions of this chapter are summarised as follows:

- Proposing a novel recommendation model in which the adversarial training strategy is used for the first time to improve the recommendation quality. We treat recommendation generation as a generative process and utilize a discriminator to help it escape from lower-quality local optima.
- Developing an efficient adversarial optimization algorithm with two loss functions to ensure that this model can be trained efficiently until it converged.
- Conducting experiments on three real-world datasets to evaluate the effectiveness of our method. Experimental results reveal that our method outperforms six state-of-the-art methods in terms of precision, recall and mean average precision metrics.

6.2 Preliminaries and Problem Formulation

In this chapter, we focus on making recommendation according to implicit feedback. Suppose there are M users $U = \{u_1, \ldots, u_M\}$, N items $V = \{v_1, \ldots, v_N\}$. Let $R \in R^{M \times N}$ denote the implicit feedback matrix, where R_{ij} equals 1 when interactions exist between user i and item j, and 0 otherwise. The formulation R_{i*} can be used to represent a user feature in which some elements are missing. Given a history of user actions, the recommendation task tries to predict a list of items which the

user might like. A recommender system is commonly formulated as the problem of estimating the missing values in user feature vector R_{i*}.

6.3 Generative Adversarial Network for Recommender Systems

In this section, we first give the problem formulation of the recommendation task. We then introduce our proposed generative adversarial model (RsyGAN) and give details of the loss functions, followed by the model optimization algorithm.

6.3.1 Proposed Model

Inspired by GAN, we combine a generative network and a discriminative network to train a property model for the recommendation task. Figure 1 illustrates our proposed model.

Two networks have been contained in the boxes delineated by a dashed line. The portrait box is a generative network and the landscape box is a discriminative network. The input of the generative network is the user feature vector R_{i*} and the input of the discriminative network is the combination of real user feature vector R_{i*} and the generated user feature by the generative network.

The recommendation task attempts to predict the missing elements in the feature vector. The generative network accepts the feature vector with missing values and returns a similar vector with all missing positions are filled. We can use the output vector to predict the user preferences more accurately.

We apply an autoencoder neural network structure as the generative network. Our network has a number of differences from the classical autoencoder. In this

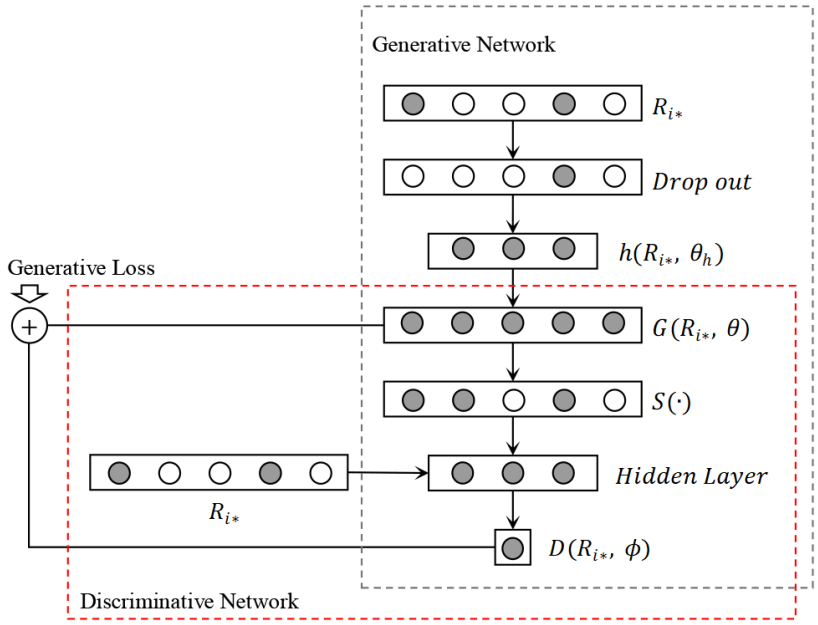


Figure 6.1 The architecture of the RsyGAN model

neural network, the feature vector with high dimension will be mapped into a hidden layer which is a lower feature space. The process of dimensionality reduction can be regarded as the extraction of features for user embedding. The hidden layer is computed as follows:

$$h(R_{i*}, \theta_h) = \sigma(W_g \times g(R_{i*}) + b_g) \tag{6.1}$$

where $\sigma(\cdot)$ is the activation function and $\theta_h = \{W_g, b_g\}$. $g(\cdot)$ is the dropout function. The dropout function is required to avoid over-fitting, because the dataset is too sparse.

We then use an output layer to recover the original user feature vector from the hidden layer. The missing values in the feature vector are filled in the output vector. The output value $\widetilde{R}_{i*}$ can be described as follows:

$$G(R_{i*}, \theta) = \sigma(W'_g \times h(R_{i*}, \theta_h) + b'_g) \tag{6.2}$$

where $\theta = \{W'_g, b'_g, \theta_h\}$. The reverse mapping may optionally be constrained by tied weights where $W = W'$ in the autoencoder, but different weights are used in our method. For the generation process, we first downsample the user's original rating vector and map it to a low-dimensional space through an encoder. After we get the user preference in the fixed feature space, we map it back to the original dimension through a decoder, so as to obtain the user's preference for other items. In order to simulate the process of user selection of items, we also need to sample this generated vector during the evaluation of the discriminator. Finally, a vector of length N is obtained to represent the user's preference for different items.

As mentioned in Section 1, we always get a lower quality local optimum due to the sparsity of the dataset. We have therefore designed a discriminative network as an quality indicator of our recommendation model, the generative network. It can be used to help the parameters be trained on the property direction. Because the evaluation function, discriminative network, can be updated according to the convergent recommendation model to help the training algorithm escaping from local optimum.

The discriminative network contains three layers: the input layer, the hidden layer and the output layer. The discriminative network can be described as follows:

$$D(R_{i*}, \phi) = \sigma(W'_d \times \sigma(W_d \times R_{i*} + b_d) + b'_d) \tag{6.3}$$

where $\phi = \{W_d, b_d, W'_d, b'_d\}$ and $\sigma(\cdot)$ is also the activation function.

We have attempted two activation functions which are defined by the formulae (4) and (5). The impaction of the activation functions is discussed in the experiments.

$$sigmoid(x) = \frac{1}{1+e^{-x}} \tag{6.4}$$

$$ReLU(x) = max(0,x) \tag{6.5}$$

6.3.2 Loss Function

Another key problem is to design a proper objective function according to the input data and output values. In GANs, the generative network samples synthetic data from a hidden feature space represented by a multilayer perceptron. However, it cannot be used directly in the recommendation task, because this task is a prediction problem, therefore we employ the incomplete user history as the input of the generative network. The generative network can only be used to predict missing values, whereas the discriminative network will try to distinguish between real users and users generated by the generative network. Two different loss functions are utilized to conduct the two step optimization.

In the discriminative network training process, we have:

$$J^D = \max_{\phi} \sum_{i}^{M} \log D(R_{i*}, \phi) + \log(1 - D(G^*(R_{i*}, \theta), \phi)) \tag{6.6}$$

where $G^*(R_{i*}, \theta) = S(G(R_{i*}, \theta))$, $S(\cdot)$ is a sample function with Bernoulli distribution $x \sim B(1, \widetilde{R}_{ij})$. The output of the generative network is a vector

containing continuous values in which $\widetilde{R}_{ij} \in (0,1)$; however, the ground truth is the binary value $R_{ij} \in \{0,1\}$.

We append J^D into the loss function of the generative network to influence the training of the model. The loss function of the generative network is as follows:

$$\begin{aligned} J^G = & \min_{\theta} \sum_{i}^{M} (\|W_{i*} \circ (R_{i*} - G(R_{i*}, \theta))\|_F^2 \\ & + \lambda_D (\log D(R_{i*}, \phi) + \log(1 - D(G^*(R_{i*}, \theta), \phi)))) + \frac{\lambda_\theta}{2} \cdot \|\theta\|_F^2 \end{aligned} \tag{6.7}$$

where $W \in \{0,1\}^{M \times N}$ is a non-negative weight matrix. Because there are positive examples in missing values, the weight matrix means the confidence of the examples. In our experiments, we set $W_{ij} = 1$ if $R_{ij} = 1$ and a low confidence level $W_{ij} = 0.1$ otherwise.

6.3.3 Optimization Algorithm

Our model contains two parts: the discriminative network and the generative network. The parameters in both networks are initialized randomly before training commences. During the adversarial training stage, the generative network and the discriminative network are trained alternately with Eqs. (6) and (7).

The model for RsyGAN is built using Tensorflow and trained with synchronous stochastic gradient descent updates. We have also open-sourced our implementation on GitHub.

We describe the detailed optimization process in Algorithm 6.1.

Algorithm 6.1: Optimization Algorithm of the Proposed Model

Input: user-item matrix R
Output: approximated user-item matrix $\widetilde{R}$
Initialize $G(R_{i*}, \theta)$ and $D(R_{i*}, \phi)$ with random weights θ, ϕ.
repeat
  for d-step **do**
    Sample a batch R_t from training set R
    Calculate the filled matrix $\widetilde{R}_t$ using $G(R_{i*}, \theta)$
    Sample from $\widetilde{R}_t$ which is subjected to Bernoulli distribution
    Update parameters ϕ by using Eq. (6)
  end for
  for g-step **do**
    Sample a batch R_t from training set R
    Update parameters θ using Eq. (7)
  end for
until converges

6.4 Experiments

6.4.1 Datasets

Experiments are conducted on three datasets namely MovieLens 1M, MovieLens 10M and Taobao. The basic statistics are listed in Table 6.1. We select 60% of records as the training set. Some records contain explicit feedback such as ratings. As we want to solve an implicit feedback task, we remove the ratings from these datasets.

Table 6.1 Statistics of the two datasets

	users	items	feedback	sparsity
ML-1M	6,040	3706	939,809	0.9580
ML-10M	69,878	10,677	104,000,054	0.9865
Taobao	8,349	5,701	321,976	0.9932

MovieLens is a widely used dataset in many researches, and many versions have been released on the GroupLens website. We choose MovieLens 1M (ML-1M) and MovieLens 10M (ML-10M) to evaluate our method.

Taobao is a dataset for competitively matching clothing on the Tianchi platform. It contains basic item data and data on the historical behaviour of users. We use only the historical behaviour data to make recommendations. We remove users with less than 10 items ($|R_{i*}| < 10$) and items with less than 20 users ($|R_{*j}| < 20$) from this dataset.

6.4.2 Evaluation for Recommendation

For top-k recommendation, we evaluate the performance of each approach using metrics precision (Prec@k), recall (Recall@k) and mean average precision (MAP@k).

Given a top-k recommendation result C_k, we can compute precision and recall as follows:

$$Precision@k = \frac{\sum_{i=1}^{|U|} |C_{k,i} \cap T_i|}{|U| \times k} \tag{6.8}$$

$$recall@k = \frac{\sum_{i=1}^{|U|} |C_{k,i} \cap T_i|}{\sum_{i=1}^{|U|} |T_i|} \tag{6.9}$$

where $C_{k,i}$ is the top-k recommendation list of user i and T_i is the items that user i has adopted in the test set.

Average precision (AP) is a ranked precision metric which is used to score information retrieval. AP@k is the average precision of all positions, which is defined as follows:

$$AP@k = \frac{\sum_{n=1}^{k} Precision@n \times rel(n)}{\min\{k, |T_i|\}} \tag{6.10}$$

where $rel(n)$ is an indicator function equalling 1 if the item at rank k is contained in the test set, otherwise 0. MAP is the mean of the AP scores for all users.

It is difficult to optimize these metrics directly because they are discontinuous. The loss function in our method is used in learning to approximate these metrics. In our experiments, we mainly show the result of top-k when $k = \{5, 10, 20, 50\}$.

6.4.3 Performance Comparison

In this subsection, we compare the proposed RsyGAN with the methods below. As our proposed model aims to make user recommendations by considering only the relationship between users and items, we mainly compare RsyGAN with user-item models.

- ItemPop: Always recommends the top-k most popular items to users.
- ItemKNN: The classical memory-based collaborative filtering method. Pearson correlation is used in our experiment and the top 50 most similar users are selected as the nearest neighborhood.
- BPR-MF: This is also a content-free algorithm based on matrix factorization which is designed for top-k recommendation tasks (Rendle et al., 2012). It optimizes pair-wise preferences between observed and unobserved items.
- CDAE: Collaborative denoising auto-encoders (Wu et al., 2016) learn latent representations of corrupted user-item preferences which can reconstruct the full input. This model is similar to our generative network.

- NCF: Neural network-based Collaborative Filtering (NCF) is a general framework for replacing the inner product with a neural architecture that can learn an arbitrary function from data.
- RsyGAN: Our method proposed in this chapter.

We cannot compare our method with RBM because the result of RBM is a binary list. It cannot be evaluated by the metrics in our experiments.

We carefully choose the hyper-parameters for each baseline method. The overall performance of the compared approaches is shown in Tables 6.2.

Table 6.2 Recommendation Performances in Terms of Precision and Recall

		Precision				Recall			
		Prec@5	Prec@10	Prec@20	Prec@50	Recall@5	Recall@10	Recall@20	Recall@50
ml-1m	*POPRANK*	0.2085	0.1911	0.1868	0.1506	0.0742	0.1211	0.1736	0.2530
	ItemKNN	0.2466	0.2351	0.2263	0.2021	0.0833	0.1367	0.1978	0.2632
	BPR-MF	0.4932	0.4617	0.4026	0.3224	0.0853	0.1495	0.2149	0.3058
	CDAE	0.5699	0.5183	0.4876	0.4592	0.0900	0.1556	0.2472	0.3826
	NCF	0.5920	0.5222	0.4895	0.4611	0.0920	0.1623	0.2676	0.3974
	RsyGAN	0.6632	0.6105	0.5087	0.3918	0.1091	0.1775	0.2702	0.4072
ml-10m	*POPRANK*	0.1934	0.1873	0.1628	0.1347	0.0307	0.0743	0.1008	0.1941
	ItemKNN	0.2404	0.2269	0.2124	0.1817	0.0384	0.0942	0.1392	0.2057
	BPR-MF	0.4153	0.3892	0.3260	0.2287	0.0612	0.1266	0.1800	0.2928
	CDAE	0.4674	0.4118	0.3643	0.2816	0.0752	0.1853	0.2200	0.3733
	NCF	0.4962	0.4759	0.3993	0.3082	0.0782	0.1832	0.2342	0.3542
	RsyGAN	0.5333	0.4854	0.4097	0.3007	0.0885	0.2094	0.2480	0.4023
Taobao	*POPRANK*	0.0040	0.0036	0.0030	0.0066	0.0024	0.0053	0.0069	0.0283
	ItemKNN	0.0517	0.0501	0.0466	0.0194	0.0267	0.0376	0.0582	0.0416
	BPR-MF	0.1047	0.0920	0.0897	0.0544	0.0408	0.0706	0.1370	0.2079
	CDAE	0.0889	0.0766	0.0608	0.0367	0.0362	0.0585	0.1196	0.1565
	NCF	0.1108	0.1023	0.0856	0.0586	0.0492	0.0774	0.1402	0.2152
	RsyGAN	0.1392	0.1314	0.0937	0.0624	0.0544	0.1038	0.1428	0.2343

We can see from the experimental results that RsyGAN achieves significant improvements across all the evaluation metrics and all the datasets. Note that the generative network is similar to CDAE, but we obtain better performance than it does. Our explanation is that there are too many local minimums in the solution space, and it is very easy to converge to a lower quality local optimum in the CDAE method. The discriminative network can be regarded as a strong constraint for the generative network when the entire solution space is searched.

We also observe that the results on MovieLens are much better than those on Taobao, because Taobao dataset is sparser than the MovieLens. The reasons can be divided into two aspects: (1) From the perspective of ensemble learning, the users in the recommender system are modeled using the generative model and the discriminative model respectively. The data generated by the generative model can alleviate the original sparse problem with a certain extension of original data. Thus, the performances of our method are better than a traditional method. (2) On the other hand, from an optimization point of view, sparse data makes it easier for the model to fall into a local optimal solution during the training process. We use an adversarial training strategy, especially go to the next epoch of training without fully convergence, which makes it easier for the model to escape from the local optimal and thus achieve better results.

6.4.4 Components in RsyGAN

In this Section, we study the influence of several main components, including the types of activation functions, the number of hidden units, and the hyper-parameter λ_D.

Table 6.3 Performance comparison of the activations function on MovieLens 1M

	MAP@5	MAP@10	MAP@20	MAP@50
Sigmoid	0.4834	0.3973	0.3420	0.3176
ReLU	0.4561	0.3800	0.3315	0.2939

Table 6.4 Performance comparison of the activations function on MovieLens 10M

	MAP@5	MAP@10	MAP@20	MAP@50
Sigmoid	0.4617	0.3804	0.3292	0.2919
ReLU	0.4401	0.3721	0.3122	0.2881

As mentioned in Section 3, we have two different types of activation function. We study their influence separately on two datasets. We show the results for the sigmoid function and the ReLU function on the hidden layer in Tables 6.3, 6.4 and 6.5.

We can see from the three tables that the sigmoid function performs better than the ReLU method in each case in the experiment, but that ReLU improves more on the larger dataset than on the smaller dataset. One possible cause is that the non-linear part in our model performs better in a small dataset. However, the model with ReLU function can be trained more efficiently, so if we have a very large dataset, we should perhaps choose the ReLU function, and if we need greater precision in a small dataset, we should choose the sigmoid function. To overcome the weakness of the linear activation function, we can also choose multiple layers for model construction.

Table 6.5 Performance comparison of the activation functions on Taobao

	MAP@5	MAP@10	MAP@20	MAP@50
Sigmoid	0.1018	0.0869	0.0715	0.0573
ReLU	0.0982	0.0827	0.0706	0.0416

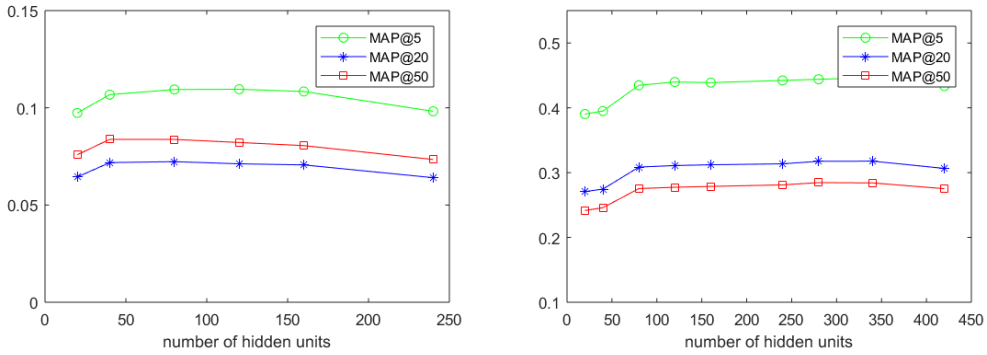


Figure 6.2 MAP@k of RsyGAN showing variations in the number of hidden units

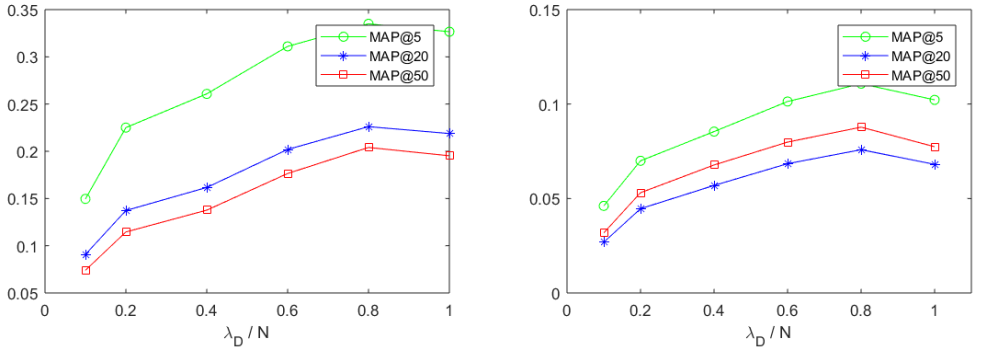


Figure 6.3 The effects of parameter λ_D

The number of hidden units is possibly another sensitive parameter in addition to the activation function. In Figure 6.2, we evaluate the performance of our method as the number of hidden units varies. We observe that the best performance is obtained when the number of hidden units is around 80 on the Taobao dataset and 350 on the MovieLens 10M dataset. This illustrates that the number of hidden units should increase with the increase in the size of the dataset.

Lastly, we study the effect of λ_D in our proposed method. We can see in Eqs. 6.6 and 6.7 that the value of the loss function of the discriminator network is very small compared to the loss function of the generative function. The output of the discriminative network is just one value range from 0 to 1, and the output of the generative networks have N numbers ranging from 0 to 1, thus we times N when we apply λ_D. Figure 6.3 shows the predictive performance for RsyGAN on the two dataset.

We find that we obtain the best performance on Taobao dataset when λ_D equals 0.8. In our extensive experiments, we observe that the value of λ_D is same on MovieLens 10M.

Since adversarial training is widely regarded as an effective but unstable technique, we further investigate the learning trend of our proposed method. Figure 4 shows the learning curves of the generative network and the discriminative network on MovieLens 10M dataset. Here we only show the performance measure by the value of the loss function. The results show that while we cannot prove that the loss function will ultimately converge, we can achieve better recommendation performance than other methods.

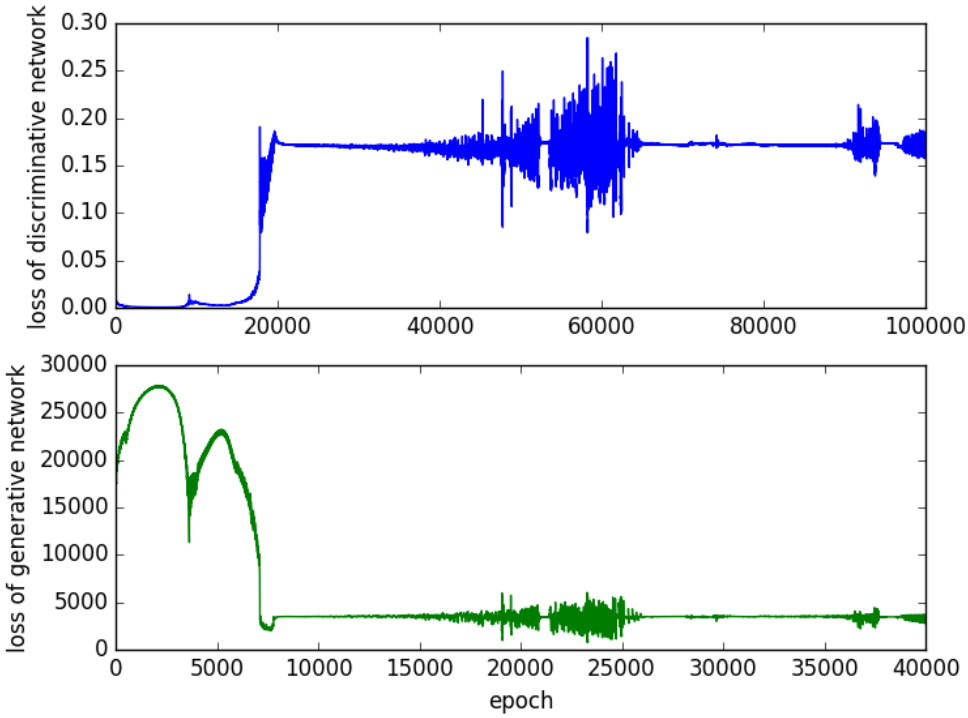


Figure 6.4 Learning curves of RsyGAN on MovieLens 10M

6.5 Summary

In this chapter, we have introduced a novel method for the top-k recommendation task which can be used in a real recommendation scenario with sparse data. The model in our method contains a generative network and a discriminative network. We utilized the adversarial strategy to train this model. The adversarial training framework takes advantage of both networks: the generative network is guided by the signals from the discriminative network, and the discriminative network can be enhanced by the generative network. We also conducted experiments on several datasets and compared the results with state-of-the-art methods. Significant performance gains were observed in each set of experiments.

Chapter 7

Conclusion and Future Research

This chapter concludes the thesis and provides further research directions for this topic.

7.1 Conclusions

Recommender systems are one of the most important techniques to help users alleviate the problem of information overload and achieve gread success in the past few years. Recommender systems help users to find the most interesting items by using information filtering technology. However, the shortcomings brought by the complexity of recommender systems and the sparsity problem of users have not been resolved so far. This research focuses on solving the following four questions concerning recommender systems: 1) take advantage of multimedia information; 2) high-order similarity information between users and items; 3) incoporate long and short-term user preferences; 4) effectively optimize with cold start users. These are still challenging problems and an investigation to provide new features to recommender systems will improve recommendation accuracy. Therefore, this

research conducts a comprehensive analysis of each of the aforementioned aspects and develops a set of recommendation methods.

The main contributions of this research are as follows:

1. It develops a novel method for the fashion recommendation task with learning compatibility knowledge in visual aspect (to achieve Objective 1).

   In the learning stage of clothing matching knowledge, a supervised clothing matching mode learning algorithm is proposed, which maximizes the matching mode that conforms to the mainstream aesthetics and the matching mode that does not conform to the mainstream aesthetics through the triple network. A map of low-dimensional embedding representations of collocation patterns. In the recommendation result generation stage based on collaborative filtering, two autoencoders are used to map the matching patterns learned in the clothing matching knowledge learning stage and the product images in the recommended products to the same feature space, reducing the visual impact between different domains. distributional differences between information, and apply this knowledge to a collaborative filtering-based product recommendation model. The experimental results show that in a large-scale real data set, the algorithm proposed in this chapter can effectively learn the knowledge of clothing matching, and transfer the learned knowledge to the target domain to be recommended, and the recommendation effect of the model that integrates the knowledge of clothing matching is excellent. Compared with other personalized clothing recommendation algorithms that regard clothing products as independent individuals that are not related to each other.

2. It develops a general framework GCF and an information propagation-based graph neural network (to achieve Objective 2) for learning a mapping that embeds users and items as points in a low-dimensional vector space with geometric relationships in the embedding space that reflect the preference relationship between users and items.

   The traditional algorithm only considers the low-order similarity between users and users, and between users and products, so that enough information cannot be obtained for new users and non-popular products to generate recommendations. A graph convolutional neural network is introduced to extend the low-order similarity in traditional algorithms to high-order similarity through layer-by-layer local information aggregation, so that more information can be used when targeting new users and non-popular products. At the same time, for the over-smoothing problem caused by the multi-layer graph convolutional neural network, we proposes an information propagation method based on the attention mechanism, which can effectively alleviate the over-smoothing problem when the graph convolutional neural network is too deep. Experiments show that the performance of this algorithm is better than other similar algorithms in this field. Applying it to the recommendation system can effectively alleviate the problem of poor recommendation effect for new users and non-popular products.

3. It develops a recommendation method LSRec based on long- and short-term user interests (to achieve Objective 3) to deal with the dynamic user interests in different context in Chapter 5.

   In the long-term interest preference learning model, a graph convolutional neural network-based method is used to learn the user's long-term stable

and invariant inherent preference. In the short-term interest preference learning model, a gated recurrent neural network is used to model the user's conversation sequence, quickly capture the user's current intention, and combine the current context to generate timely recommendations for the user. The experimental results show that the method proposed in this work achieves better results in comparison with methods that focus on users' long-term interest preferences and methods that focus on users' short-term interest preferences. It is fully proved that the assumption that the user's interest preference is divided into long-term stable and short-term dynamic change is in line with the actual situation, and the recommendation algorithm based on the user's long-term and short-term interest preference can better solve the problem of dynamic change of user's interest preference.

4. It develops a novel method for the top-k recommendation task (to achieve Objective 4) which can be used in a real recommendation scenario with sparse data.

   In the generative model, a stack of denoising autoencoders is used to predict the missing parts in the user's history vector and produce recommendations accordingly. After that, the discriminant model discriminates the recommendation results generated by the generative model with the real user shopping records, so as to guide the training direction of the generative model, so that the generative model can generate recommendation results that are as similar to the distribution of real shopping records as possible. Experiments show that the algorithm can effectively pass the recommendation results and generate better recommendation results than the single generation model,

thus alleviating the problem of poor optimization caused by data sparseness to a certain extent.

7.2 Future Study

There are some limitations of this study. This thesis identifies the following directions as future work:

- For the recommendation system for clothing products, the cost of data labeling for learning clothing matching knowledge is relatively high. Researching automated methods to learn matching knowledge from unlabeled data has high economic value and practical value. Subsequent work can try to learn fashion matching knowledge with less labeled or unlabeled datasets from the perspectives of semi-supervised learning, unsupervised learning, and transfer learning.

- In a real production environment, the builders and users of recommender systems often want to know the reason for recommending a certain item. However, the interpretability of recommendation results generated by recommender systems based on deep learning models and collaborative filtering is poor. Subsequent work can apply the graph neural network to the unstructured data based on the knowledge graph in the recommendation system, and use the information in the knowledge graph to make the model interpretable to a certain extent. Therefore, how to combine traditional feature-based recommendation models with knowledge graphs to improve the interpretability of the models has high research value. On the other hand,

inference-based methods can also be considered to add some interpretability to the model.

- Considering the continuous addition of new users and new products to the recommendation system in practical scenarios, the interests of users change rapidly over time, and the use value and meaning of the products themselves are constantly changing. How to dynamically update the model through online learning? It is also an urgent problem to be solved.

- In addition, we have also started some new attempts in the field of recommender system security. For example, we propose a new method to generate our expected recommendation results by adding malicious noise to the recommender system. This method can be used to conduct targeted attacks on the target recommender system to increase or decrease the probability of certain items being recommended.

Bibliography

Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions.

Adomavicius, G. and Tuzhilin, A. (2015). Context-aware recommender systems. *Recommender Systems Handbook, Second Edition*.

Atwood, J. and Towsley, D. (2016). Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1993–2001.

Barkan, O. and Koenigstein, N. (2016). Item2vec: neural item embedding for collaborative filtering. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE.

Batmaz, Z., Yurekli, A., Bilge, A., and Kaleli, C. (2019). A review on deep learning for recommender systems: challenges and remedies. *Artificial Intelligence Review*, 52(1):1–37.

Billsus, D., Billsus, D., Pazzani, M., and Pazzani, M. (1998). Learning collaborative information filters. *Proceedings of the Fifteenth International Conference on Machine Learning*.

Bobadilla, J., Ortega, F., Hernando, A., and Gutiérrezz, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, 46:109–132.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. (2017). Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Campos, P. G., Díez, F., and Cantador, I. (2014). Time-aware recommender systems: A comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*.

Cao, H., Chen, E., Yang, J., and Xiong, H. (2009). Enhancing recommender systems under volatile user interest drifts. In *International Conference on Information and Knowledge Management, Proceedings*.

Chen, R.-C. et al. (2019). User rating classification via deep belief network learning and sentiment analysis. *IEEE Transactions on Computational Social Systems*, 6(3):535–546.

Chen, T., Zhang, W., Lu, Q., Chen, K., Zheng, Z., and Yu, Y. (2012). SVDFeature: A toolkit for feature-based collaborative filtering. *Journal of Machine Learning Research*.

Cheng, C., Yang, H., Lyu, M. R., and King, I. (2013). Where you like to go next: Successive point-of-interest recommendation. In *IJCAI International Joint Conference on Artificial Intelligence*.

Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., et al. (2016). Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Covington, P., Adams, J., and Sargin, E. (2016). Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198.

Cui, Q., Wu, S., Huang, Y., and Wang, L. (2019). A hierarchical contextual attention-based network for sequential recommendation. *Neurocomputing*.

Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., Gupta, S., He, Y., Lambert, M., Livingston, B., et al. (2010). The youtube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 293–296.

De Andrade, A. (2019). Best practices for convolutional neural networks applied to object recognition in images. *arXiv preprint arXiv:1910.13029*.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852.

Deshpande, M. and Karypis, G. (2004). Item-based top-N recommendation algorithms. *ACM Transactions on Information Systems*.

Devooght, R. and Bersini, H. (2017). Long and short-Term recommendations with recurrent neural networks. *UMAP 2017 - Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 13–21.

Diao, Q., Qiu, M., Wu, C. Y., Smola, A. J., Jiang, J., and Wang, C. (2014). Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Ding, Y. and Li, X. (2005). Time weight collaborative filtering. In *International Conference on Information and Knowledge Management, Proceedings*.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232.

Ge, T., Zhao, L., Zhou, G., Chen, K., Liu, S., Yi, H., Hu, Z., Liu, B., Sun, P., Liu, H., Yi, P., Huang, S., Zhang, Z., Zhu, X., Zhang, Y., and Gai, K. (2017). Image matters: Jointly train advertising CTR model with image representation of Ad and user behavior. *arXiv*.

Georgiev, K. and Nakov, P. (2013). A non-IID frameworkfor collaborative filtering with restricted boltzmann machines. In *International Conference on Machine Learning*, volume 28, pages 1–9.

Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using collaborative filtering to Weave an Information tapestry. *Communications of the ACM*.

Gomez-Uribe, C. A. and Hunt, N. (2015). The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):1–19.

Gong, Y. and Zhang, Q. (2016). Hashtag recommendation using attention-based convolutional neural network. In *IJCAI*, pages 2782–2788.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. In *NIPS*, pages 1–9.

Gori, M., Monfardini, G., and Scarselli, F. (2005). A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE.

Gori, M. and Pucci, A. (2007). ItemRank: A random-walk based scoring algorithm for recommender engines. In *IJCAI International Joint Conference on Artificial Intelligence*.

Grbovic, M. and Cheng, H. (2018). Real-time personalization using embeddings for search ranking at airbnb. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 311–320.

Gu, Y., Lei, T., Barzilay, R., and Jaakkola, T. (2016). Learning to refine text based recommendations. In *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*.

Guo, H., Tang, R., Ye, Y., Li, Z., and He, X. (2017). DeepFM: A factorization-machine based neural network for CTR prediction. In *IJCAI International Joint Conference on Artificial Intelligence*.

Guo, Y., Cheng, Z., Nie, L., Wang, Y., Ma, J., and Kankanhalli, M. (2019). Attentive long short-term preference modeling for personalized product search. *ACM Transactions on Information Systems*, 37(2):1–27.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034.

Han, X., Wu, Z., Jiang, Y.-G., and Davis, L. S. (2017). Learning fashion compatibility with bidirectional lstms. In *Proceedings of the 2017 ACM on Multimedia Conference*, pages 1078–1086. ACM.

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

He, R. and McAuley, J. (2016a). Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *25th International World Wide Web Conference, WWW 2016*.

He, R. and McAuley, J. (2016b). Vbpr: Visual bayesian personalized ranking from implicit feedback. In *AAAI*, pages 144–150.

He, X. and Chua, T.-S. (2017). Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 355–364.

He, X., Gao, M., Kan, M.-Y., and Wang, D. (2017a). BiRank: Towards ranking on bipartite graphs. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):57–71.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. (2017b). Neural collaborative filtering. In *International World Wide Web Conferences*, pages 173–182.

He, X., Pan, J., Jin, O., Xu, T., Liu, B., Xu, T., Shi, Y., Atallah, A., Herbrich, R., Bowers, S., and Candela, J. Q. (2014). Practical lessons from predicting clicks on ads at Facebook. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

He, X., Zhang, H., Kan, M. Y., and Chua, T. S. (2016). Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR 2016 - Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Hidasi, B. and Karatzoglou, A. (2018). Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM*

international conference on information and knowledge management, pages 843–852.

Hidasi, B., Karatzoglou, A., Baltrunas, L., and Tikk, D. (2015). Session-based recommendations with recurrent neural networks. *International Conference on Learning Representations*, pages 1–9.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*.

Hong, W., Li, L., and Li, T. (2012). Product recommendation with temporal dynamics. *Expert Systems with Applications*.

Huang, T., She, Q., Wang, Z., and Zhang, J. (2020). Gatenet: gating-enhanced deep network for click-through rate prediction. *arXiv preprint arXiv:2007.03519*.

Jamali, M. and Ester, M. (2009). TrustWalker: A random walk model for combining trust-based and item-based recommendation. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Jiang, W., Jiao, Y., Wang, Q., Liang, C., Guo, L., Zhang, Y., Sun, Z., Xiong, Y., and Zhu, Y. (2022). Triangle graph interest network for click-through rate prediction.

In *Proceedings of the fifteenth ACM international conference on web search and data mining*, pages 401–409.

Kang, W.-C., Fang, C., Wang, Z., and McAuley, J. (2017). Visually-aware fashion recommendation and design with generative image models. In *Data Mining (ICDM), 2017 IEEE International Conference on*, pages 207–216. IEEE.

Kapoor, K., Subbian, K., Srivastava, J., and Schrater, P. (2015). Just in time recommendations - Modeling the dynamics of boredom in activity streams. In *WSDM 2015 - Proceedings of the 8th ACM International Conference on Web Search and Data Mining*.

Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM.

Koren, Y. (2009). The bellkor solution to the netflix grand prize. *Netflix prize documentation*, 81(2009):1–10.

Koren, Y. (2010). Collaborative filtering with temporal dynamics. *Communications of the ACM*.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*.

Lang, K. (1995). NewsWeeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*.

Li, J., Ren, P., Chen, Z., Ren, Z., Lian, T., and Ma, J. (2017). Neural attentive session-based recommendation. *International Conference on Information and Knowledge Management, Proceedings*, Part F1318:1419–1428.

Liang, H., Xu, Y., Li, Y., and Nayak, R. (2010). Personalized recommender system based on item taxonomy and folksonomy. In *International Conference on Information and Knowledge Management, Proceedings*.

Liu, B., Xiong, H., Papadimitriou, S., Fu, Y., and Yao, Z. (2015). A general geographical probabilistic factor model for point of interest recommendation. *IEEE Transactions on Knowledge and Data Engineering*.

Liu, H., Lu, J., Yang, H., Zhao, X., Xu, S., Peng, H., Zhang, Z., Niu, W., Zhu, X., Bao, Y., et al. (2020). Category-specific cnn for visual-aware ctr prediction at jd. com. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2686–2696.

Liu, H., Wei, Y., Yin, J., and Nie, L. (2022). Hs-gcn: hamming spatial graph convolutional networks for recommendation. *IEEE Transactions on Knowledge and Data Engineering*.

Lu, J., Wu, D., Mao, M., Wang, W., and Zhang, G. (2015). Recommender system application developments: A survey. *Decision Support Systems*, 74:12–32.

Luo, X., Zhou, M. C., Li, S., You, Z., Xia, Y., and Zhu, Q. (2016). A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method. *IEEE Transactions on Neural Networks and Learning Systems*.

Lynch, C., Aryafar, K., and Attenberg, J. (2016). Images don't lie: Transferring deep visual semantic features to large-scale multimodal learning to rank. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 541–548. ACM.

Mao, M., Lu, J., Han, J., and Zhang, G. (2019). Multiobjective e-commerce recommendations based on hypergraph ranking. *Information Sciences*, 471:269–287.

Mao, M., Lu, J., Zhang, G., and Zhang, J. (2017). Multirelational social recommendations via multigraph ranking. *IEEE transactions on cybernetics*, 47(12):4049–4061.

McAuley, J. and Leskovec, J. (2013a). From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews. In *WWW 2013 - Proceedings of the 22nd International Conference on World Wide Web*.

McAuley, J. and Leskovec, J. (2013b). Hidden factors and hidden topics: Understanding rating dimensions with review text. In *RecSys 2013 - Proceedings of the 7th ACM Conference on Recommender Systems*.

McAuley, J., Targett, C., Shi, Q., and van den Hengel, A. (2015). Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '15*, pages 43–52, New York, New York, USA. ACM Press.

Medsker, L. and Jain, L. C. (1999). *Recurrent neural networks: design and applications*. CRC press.

Monti, F., Bronstein, M., and Bresson, X. (2017). Geometric matrix completion with recurrent multi-graph neural networks. In *Advances in Neural Information Processing Systems*, pages 3697–3707.

Mu, R. (2018). A survey of recommender systems based on deep learning. *IEEE Access*, 6:69009–69022.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1998). The pageRank citation ranking: bringing order to the web. *World Wide Web Internet And Web Information Systems*.

Pan, Z., Cai, F., Chen, W., Chen, C., and Chen, H. (2022). Collaborative graph learning for session-based recommendation. *ACM Transactions on Information Systems (TOIS)*, 40(4):1–26.

Pang, Y., Wu, L., Shen, Q., Zhang, Y., Wei, Z., Xu, F., Chang, E., Long, B., and Pei, J. (2022). Heterogeneous global graph neural networks for personalized session-based recommendation. In *Proceedings of the fifteenth ACM international conference on web search and data mining*, pages 775–783.

Panniello, U., Tuzhilin, A., and Gorgoglione, M. (2014). Comparing context-aware recommender systems in terms of accuracy and diversity. *User Modeling and User-Adapted Interaction*.

Panniello, U., Tuzhilin, A., Gorgoglione, M., Palmisano, C., and Pedone, A. (2009). Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *RecSys'09 - Proceedings of the 3rd ACM Conference on Recommender Systems*.

Qian, X., Feng, H., Zhao, G., and Mei, T. (2014). Personalized recommendation combining user interest and social circle. *IEEE Transactions on Knowledge and Data Engineering*.

Quadrana, M., Karatzoglou, A., Hidasi, B., and Cremonesi, P. (2017). Personalizing session-based recommendations with hierarchical recurrent neural networks. In

RecSys 2017 - Proceedings of the 11th ACM Conference on Recommender Systems.

Rendle, S. (2010a). Factorization machines. *IEEE International Conference on Data Mining*, pages 995–1000.

Rendle, S. (2010b). Time-Variant Factorization Models. *Context-Aware Ranking with Factorization Models*, pages 137–153.

Rendle, S. and Freudenthaler, C. (2014). Improving pairwise learning for item recommendation from implicit feedback. In *Web Search and Data Mining*, pages 273–282.

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2012). BPR: Bayesian personalized ranking from implicit feedback. *Conference on Uncertainty in Artificial Intelligence*, pages 452–461.

Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. (2010). Factorizing personalized Markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web, WWW '10*.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, CSCW 1994*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Salakhutdinov, R. and Mnih, A. (2009). Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems 20 - Proceedings of the 2007 Conference*.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web, WWW 2001*.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Sedhain, S., Menony, A. K., Sannery, S., and Xie, L. (2015). AutoRec: Autoencoders meet collaborative filtering. In *WWW 2015 Companion - Proceedings of the 24th International Conference on World Wide Web*, pages 111–112.

Shen, X., Yi, B., Zhang, Z., Shu, J., and Liu, H. (2016). Automatic recommendation technology for learning resources with convolutional neural network. In *2016 international symposium on educational technology (ISET)*, pages 30–34. IEEE.

Shi, Y., Larson, M., and Hanjalic, A. (2014). Collaborative Filtering beyond the User-Item Matrix : A Survey of the State of the Art and Future Challenges. *ACM Computing Surveys*, 47(1):1–45.

Singhal, A., Sinha, P., and Pant, R. (2017). Use of deep learning in modern recommendation system: A summary of recent works. *arXiv preprint arXiv:1712.07525*.

Smirnova, E. and Vasile, F. (2017). Contextual sequence modeling for recommendation with Recurrent Neural Networks. In *ACM International Conference Proceeding Series*.

Su, X. and Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*.

Symeonidis, P. (2016). Matrix and tensor decomposition in recommender systems. In *RecSys*, pages 429–430.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12.

Tan, Y. K., Xu, X., and Liu, Y. (2016). Improved recurrent neural networks for session-based recommendations. *ACM International Conference Proceeding Series*, 15-Septemb:17–22.

Tang, J. and Wang, K. (2018). Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 565–573.

Tsymbal, A. (2004). The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*.

Tuan, T. X. and Phuong, T. M. (2017). 3D convolutional networks for session-based recommendation with content features. In *RecSys 2017 - Proceedings of the 11th ACM Conference on Recommender Systems*.

Villatel, K., Smirnova, E., Mary, J., and Preux, P. (2018). Recurrent neural networks for long and short-term sequential recommendation. *arXiv*.

Wang, H., Chen, B., and Li, W. J. (2013). Collaborative topic regression with social regularization for tag recommendation. In *IJCAI*, pages 2719–2725.

Wang, H., Wang, N., and Yeung, D.-Y. (2015). Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1235–1244.

Wang, J., Chen, Y., Chakraborty, R., and Yu, S. X. (2020a). Orthogonal convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11505–11515.

Wang, J., Huang, P., Zhao, H., Zhang, Z., Zhao, B., and Lee, D. L. (2018). Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 839–848.

Wang, W., Zhang, G., and Lu, J. (2016a). Member contribution-based group recommender system. *Decision Support Systems*, 87:80–93.

Wang, W., Zhang, G., and Lu, J. (2017). Hierarchy visualization for group recommender systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(6):1152–1163.

Wang, X., Donaldson, R., Nell, C., Gorniak, P., Ester, M., and Bu, J. (2016b). Recommending groups to users using user-group engagement and time-dependent matrix factorization. In *30th AAAI Conference on Artificial Intelligence, AAAI 2016*.

Wang, Z., She, Q., Zhang, P., and Zhang, J. (2020b). Correct normalization matters: Understanding the effect of normalization on deep neural network models for click-through rate prediction. *arXiv preprint arXiv:2006.12753*.

Wu, D., Lu, J., and Zhang, G. (2015). A fuzzy tree matching-based personalized e-learning recommender system. *IEEE Transactions on Fuzzy Systems*, 23(6):2412–2426.

Wu, J., Wang, X., Feng, F., He, X., Chen, L., Lian, J., and Xie, X. (2021). Self-supervised graph learning for recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 726–735.

Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., and Tan, T. (2019). Session-based recommendation with graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:346–353.

Wu, Y., DuBois, C., Zheng, A. X., and Ester, M. (2016). Collaborative denoising auto-encoders for top-N recommender systems. In *WSDM 2016 - Proceedings of the 9th ACM International Conference on Web Search and Data Mining*, pages 153–162.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24.

Yang, B., Lei, Y., Liu, J., and Li, W. (2017). Social collaborative filtering by trust. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Yildirim, H. and Krishnamoorthy, M. S. (2008). A random walk method for alleviating the sparsity problem in collaborative filtering. In *RecSys'08: Proceedings of the 2008 ACM Conference on Recommender Systems*.

Yin, H., Cui, B., Chen, L., Hu, Z., and Zhou, X. (2015). Dynamic user modeling in social media systems. *ACM Transactions on Information Systems*.

Yu, W., Zhang, Z., and Qin, Z. (2022). Low-pass graph convolutional network for recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8954–8961.

Yuan, F., Guo, G., Jose, J. M., and Chen, L. (2016). LambdaFM : Learning optimal ranking with factorization machines using lambda surrogates. In *International Conference on Information and Knowledge Management*, pages 227–236.

Zhang, C., Wang, K., Yu, H., Sun, J., and Lim, E. P. (2014). Latent factor transition for dynamic collaborative filtering. In *SIAM International Conference on Data Mining 2014, SDM 2014*.

Zhang, F., Yuan, N. J., Lian, D., Xie, X., and Ma, W.-Y. (2016). Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362.

Zhang, J., Huang, T., and Zhang, Z. (2019). Fat-deepffm: Field attentive deep field-aware factorization machine. *arXiv preprint arXiv:1905.06336*.

Zhang, Q., Lu, J., Wu, D., and Zhang, G. (2018). A cross-domain recommender system with kernel-induced knowledge transfer for overlapping entities. *IEEE transactions on neural networks and learning systems*.

Zhang, Q., Wu, D., Lu, J., Liu, F., and Zhang, G. (2017a). A cross-domain recommender system with consistent information transfer. *Decision Support Systems*, 104:49–63.

Zhang, S., Yao, L., and Sun, A. (2017b). Deep Learning based Recommender System: A Survey and New Perspectives. *arXiv*, 1(1):1–35.

Zhao, K., Li, Y., Shuai, Z., and Yang, C. (2018). Learning and transferring ids representation in e-commerce. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1031–1039.

Zheng, L., Noroozi, V., and Yu, P. S. (2017). Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 425–434.

Zhou, J., Albatal, R., and Gurrin, C. (2016). Applying visual user interest profiles for recommendation and personalisation. In *International Conference on Multimedia Modeling*, pages 361–366. Springer.

Abbreviations

BPR	**B**ayesian **P**ersonalized **R**anking
CDL	**C**ollaborative **D**eep **L**earning
CF	**C**ollaborative **F**iltering
CKE	**C**ollaborative **K**nowledge based **E**mbedding
CNN	**C**onvolutional **N**eural **N**etwork
DBN	**D**eep **B**elief **N**etworks
GCF	**G**raph neural network-based **C**ollaborative **F**iltering
GCN	**G**raph **C**onvolutional **N**etwork
GNN	**G**raph **N**eural **N**etwork
GRU	**G**ated **R**ecurrent **U**nit
L-UIN	**L**ong-term **U**ser **I**nterest **N**etwork
LSREC	**L**ong- and **S**hort-term user interest network for personalized **rec**ommendation
MF	**M**atrix **F**actorization
MLP	**M**ulti-layer **P**erceptron
PMF	**P**robabilistic **M**atrix **F**actorization
RNN	**R**ecurrent **N**eural **N**etwork
S-UIN	**S**hort-term **U**ser **I**nterest **N**etwork

SCAE **S**tacked **C**onvolutional **A**uto**E**ncoder

SDAE **S**tacked **D**enoising **A**uto**E**ncoder