Deep Representation for Social-aware Recommendation via Neural Network

神経ネットワークによる ソーシャルリコメンデーションの深層表現

February 2021

Munan LI
李 沐南

Deep Representation for Social-aware Recommendation via Neural Network

神経ネットワークによるソーシャルリコメンデーションの深層表現

February 2021

Waseda University
Graduate School of Fundamental Science and Engineering
Department of Computer Science and Communication, Research on Software Development Engineering

Munan LI
リ　モクナン

WASEDA UNIVERSITY

Abstract

Faculty of Science and Engineering

Doctor of Engineering

Deep Representation for Social-aware Recommendation via Neural Network

by LI MUNAN

With the rapid popularization of Youtube, Amazon, Facebook and other online platforms, recommendation system has become an essential service in daily life. Accurate recommendation system can bring huge revenue for enterprises. Therefore, how to make reasonable use of historical information to provide accurate recommendation results has attracted more and more attention. The traditional collaborative filtering recommendation systems which mainly based on user-item history interactions often fall in with the data sparsity problem. For the purpose of overcoming this trouble, social recommendation methods have become one of the successful methods. Although the exited work has achieved success, there are still some limitations. For example, most existed works directly utilized the features of friends and ignore that the invalid feature will bring noise the model. This is due to the fact that most existed works supposed that social neighbors share similar partiality. In fact, users' hobbies are diverse, and not all of their social neighbors' hobbies will have an impact on users. Another problem of the influence in social effect is that the influence of social relations is not universally applicable to any context. We refer to the user's choice of different items as a context. When user faced with different items, the social influence changes dynamically. Besides, few works focused on the sparsity problem of social data. As real-world social networks are usually sparse, the observed relationships in social networks were quite limited. The sparse social data will degrade the performance of the existing social-based algorithms. To this end, a novel deep learning-based framework was proposed to obtain useful user representations. In particular, the contributions of the proposed approach could be concluded as the followings: First, we use network embedding technology to obtain more effective embeddings of users to promote the convergence speed of the proposed model and improve the effectiveness of the recommendation system. Besides, through the factor-level attention model, we can learn specific preference feature of users to reduce the input noise. Also, we leverage attention neural networks to model the social dynamic influences. Finally, we propose to use heterogeneous information network to find users with similar preferences, and use these social relations to enrich the users' features. A two-layer attention structure model was proposed to strengthen the interpretability for why such prediction results were provided. Finally, for the purpose of verifying the effectiveness of the proposed approach, a large number of experiments were carried out on reliable data sets.

Acknowledgements

The completion of this paper condensed the painstaking efforts of many people, without them, this paper could not be completed. I would like to thank my advisor, Professor Fukazawa Yoshiaki, and my associate advisor, Associate Professor Tei Kenji. I would also like to thank Professor Washizaki Hironori and Professor Honiden Shinichi. I am very grateful for their valuable comments on my thesis, which helped me complete my PhD thesis.

First of all, I would like to thank my advisor, Professor Fukazawa Yoshiaki. During the three years from my doctoral study to the completion of my thesis, Professor Fukazawa Yoshiaki has been giving me a lot of help. With the help of Professor Fukazawa Yoshiaki, I was lucky enough to start my doctoral study at Waseda University. It was also with the help of Professor Fukazawa Yoshiaki that I was able to keep moving forward in my research. Fukazawa Yoshiaki has impressed me with his profound knowledge, approachable personality, rigorous academic attitude. Here, I would like to express my heartfelt gratitude to my advisor, Professor Fukazawa Yoshiaki.

Secondly, I'd like to thank my associate advisor, Associate Professor Tei Kenji. My associate advisor, Professor Tei Kenji, devoted a lot of effort to my paper. From the topic selection, implementation to the final draft of the paper, Professor Tei Kenji gave me careful guidance and support in all aspects. Without him, I could not completed this paper.

Also, I would like to express my gratitude to the friends I have met in Waseda University, who cheered me up when I was not confident, helped and encouraged me in every aspect of my life, and made my days happy and fulfilling. I will always be grateful to them for their help and company in my doctoral career.

Finally, I would like to thank my family, who have given me great support and encouragement during the three years of my doctoral life.

Contents

Abstract **iii**

Acknowledgements **v**

1 Introduction **1**
1.1 Recommendation System 1
1.2 Social Recommendation System 2
1.3 Prior Works and Limitations 3
1.4 Objectives 5
1.5 Outline 6

2 Existing Approaches in Recommendation System **7**
2.1 Social Recommendation 7
2.2 Deep Learning-based Recommendation System 9
2.2.1 Deep Learning-based Collaborative Filtering 10
2.2.2 Deep Hybrid Models for Recommendation 12
2.2.3 Personalized Recommendation Based on Graph Neural Network 12

3 Technical Background **15**
3.1 Model-based Recommendation System 15
3.1.1 Learning Model of Matrix factorization 17
3.1.2 Matrix Factorization using Bayesian Personalized Ranking 17
3.2 Deep Neural Network 19
3.2.1 A Multi-Layer Perceptron 19
3.2.2 Learning Process of a Neural Network 21
3.3 The Basic Framework of Recommendation System based on Deep Learning 23
3.4 Embedding and Representation Learning 24
3.4.1 Embedding Matrix 25

3.4.2 Network Embedding 26
3.4.3 Attention Model 28
3.5 Heterogenous Information Network 29

4 Representation Learning Framework for Social Recommendation 31
4.1 Problem Definition 31
4.2 Notations and Formulation 32
4.3 Probabilistic Model in RLSeSys 33
4.4 The RLSeSys Model 34
4.5 A Whole Architecture of RLReSys 36
4.6 Model Optimization 37

5 An Adaptive Attentive Model for Social Recommendation 41
5.1 Motivations and Rationales 41
5.2 Adaptive Neural Network for Social Recommendation(ANSR) 44
5.2.1 Model Framework 44
5.2.2 Network embedding-based representation learning . . 48
5.2.3 Attention-based social effects modeling 50
5.2.4 Optimization and Training 53
5.3 Experiments Settings 56
5.3.1 Datasets 56
5.3.2 Baselines 57
5.3.3 Evaluation protocols 58
5.3.4 Parameter settings 60
5.4 Performance Evaluation 60
5.4.1 Overall Performance Comparison 60
5.4.2 Model Analysis 64
Attention Module Analysis 64
Network Embedding Module Analysis 67
5.5 Discussion 69
5.5.1 Data Sparsity Problem Study 69
5.5.2 Discussion 70
5.5.3 Threats to the Experiment 72

6 Adaptive Social Influence Learning for Recommendation via Heterogeneous information Networks 75
6.1 Motivations and Rationales 75
6.2 Methodology 79

6.2.1 Overview of HASRec 79
6.2.2 Missing Links Identify Module 79
6.2.3 Attention-based Recommendation Module 84
6.3 Optimization and Training 87
6.4 Experimental Settings 89
6.4.1 Datasets 89
6.4.2 Baselines 89
6.4.3 Evaluation protocols 90
6.4.4 Hyper-parameter Settings 91
6.5 Performance Evaluation 92
6.5.1 Performance Comparison of Recommendation Systems 92
6.5.2 Model Analysis 94
6.5.3 Case Study 96

7 Conclusions and Future Work **99**

List of Figures

3.1 An example of the user-item interaction matrix. 15
3.2 An example of matrix factorization: decompose the rating matrix R into two smaller matrices P and Q 16
3.3 An example of machine learning process of MF. 17
3.4 An example of user implicit feedback. 18
3.5 The network structure of a full-connected feedforward neural network 20
3.6 A single neuron 20
3.7 An example of a three-layer perceptron 21
3.8 The basic framework of RS based on DL 23
3.9 An example of the user embedding matrix. 26
3.10 An illustration of attention mechanism 28

4.1 An example of the input data in social recommendation system. 33
4.2 An illustration of integrating social information into MF-based model. 35
4.3 The overall architecture of representation learning framework in Social Recommendation. 36

5.1 An illustration of social influences, (a) demonstrates that users may not have similar preferences in all aspects and (b) demonstrates different social neighbors contributes different to users' decision 43
5.2 The overall Architecture Framework of ANSR. 45
5.3 An example of utilizing one-hot-encoding to represent an entity. 46
5.4 An example of mapping the identity of item to a low-dimensional space to obtain a dense representation of an entity. 46
5.5 An example of the explanation of an item vector. 47
5.6 The illustration of Network Embedding. The connected users will have a similar embedding vector in the latent space. . . . 48

5.7 The illustration of the Deep Walk Methods. First, generate a social corpus with random walk. Then, learn the node representation using the Skip-Gram Model. 49
5.8 The illustration of the framework of aspect-aware module. . . 51
5.9 The illustration of the influence calculating module. 54
5.10 The overall comparison results. 62
5.11 The illustration of the variant models. 65
5.12 Performance of HR@10 and NDCG@10 varying iterations on Epinions and Douban 67
5.13 Performance of HR@10 and NDCG@10 varying iterations on $Douban_5$ and $Douban_10$ 69
5.14 (a), (b) are HR@10 and NDCG@10 of NeuMF, CUNE-BPR, ANSR on the Douban dataset w.r.t. different data sparsity levels. . . 70
5.15 (a) and (b) are HR@10 and NDCG@10 of ANSR on the Ciao dataset w.r.t different embedding dimension size D, respectively. (c) and (d) are HR@10 and NDCG@10 of ANSR on the Ciao dataset w.r.t different regularization parameter λ, respectively. 72

6.1 Example of the social friends' size of users grouping by the number of the observed rating data. 76
6.2 Example of an HIN in a recommendation system. The dotted blue line represents that a user has rated an item, and the solid yellow line represents the user trust his/her friends. 78
6.3 Example of the social influence. 79
6.4 The framework of Missing Links Identify Module 80
6.5 the Construction of a Heterogeneous Information Network from a user-item rating network and a user-user social network. . . 81
6.6 The meta path designed in our work. U denotes the user and I denotes the item. 82
6.7 The architecture framework of Attention-based Recommendation Module. 84
6.8 The framework of the attention layer. 85
6.9 The variant models in ablation study. 94

6.10 (a) and (b) are Recall@10 and NDCG@10 of the variant models of HASRec on the delicious dataset, respectively. (c) and (d) are Recall@10 and NDCG@10 of the variant models of HASRec on the douban, respectively. 96
6.11 (a) and (b) are Recall@10 and NDCG@10 of NeuMF, CUNE-BPR, ANSR and HASRec on the douban dataset w.r.t. different data sparsity levels. 97
6.12 Case Study: The attention distribution of sampled friends of $user(1408)$. 98

List of Tables

5.1 Data set statistics 57
5.2 The comparison results on variant models. 66
5.3 The comparison results varying different network embedding methods. 68
5.4 Leave-one-out vs Hold-out 73

6.1 Recall@K and NDCG@K comparisons for different top-k values on different datasets 93

Chapter 1

Introduction

1.1 Recommendation System

In general, a recommendation system is an algorithm that recommend relevant items to users. Depending on the online platform, the item can be a music to listen, a news to read, or a friend to follow. With the rapid popularization of Youtube, Amazon, Facebook and other online platforms, recommendation system has become an essential service in daily life. For example, E-commerce platforms recommend items to the target users in line with their preferences through the recommendation system. An accurate recommendation system is important to both the enterprise and the user. An efficient recommendation system can generate huge revenue for enterprises and help them stand out from their competitors. Users can also save considerable time to catch the information that they are satisfied with, and increase their trust in that online platform. Therefore, how to boost the effectiveness of the recommendation system has become the focus referring to the industry.

Until now, the most generally practiced algorithm in recommendation domain is the model-based Collaborative Filtering [1], also known as Matrix Factorization(MF) [2]. MF generates new recommendations depended on "user-item rating matrix" [3] which store the rating history of the users given to the items. The core concept of MF lied in decomposing the high-dimensional "user-item rating matrix" into different smaller low-dimensional matrices: one is called user-factor matrix and another is factor-item matrix. The row vectors in the user-feature matrix can represent users' preference feature, while the row vectors in the item-feature matrix can denote the attributes of items. In this way, MF can learn the missing rating in the "user-item rating matrix" through computing the dot product of these two matrices.

Although MF is successfully used in recommendation system, the data

sparsity problem seriously affects its performance. In some large online platforms like Amazon, the number of users are in millions the variety of items are tens of thousands. In this case, the "user-item rating matrix" is extremely sparse, which means that there are very few elements in the matrix whose value is not 0. This will degrade the performance of recommendation system. One extreme problem with data sparsity is the cold start problem.[4] It refers to a situation where a new user or item has just entered the system. Collaborative filtering are not able to generate useful recommendations for that there is no interaction records for that user or item. Therefore, to mitigate the data sparsity problem, many studies have tried to supplement the data of users and items through expoliting side informations such as social networks, reviews.

1.2 Social Recommendation System

The recommendation system based on users' social data provided an important direction for solving the data sparsity problem. Nielsen, a well-known third-party research agency in the United States, surveyed the causes that affect users to believe in the results provided by the system. The outcomes show that ninety percent of the users trust their friends' suggestions and seventy percent trust other online users' comments on advertised products. It can be seen from the survey that friends' recommendations are certainly critical to escalation the users' confidence to the results provided by the system. The popularity of web-based social media has made it simpler for users to express their preferences and develop social relations online. These social relation information can effectively enrich the user's personal characteristics and provide important assistant for the recommendation system. The reason why social recommendation is so important in many websites is due to the following advantages: Social associates' suggestion can raise the expectation of the recommendation system; Besides, utilizing social relations can manage the "cold start" issue. When a new client logs into the website via Facebook account, we can get the list of friends of the user in the online media site, and then recommend to the user the items that his social associates appreciated on that online website. Thus, we can provide users with tremendous results when there are no user behavior records, which partly solves the problem of cold start of the recommendation system.

Social scientists have found that the connected users in social network will share similar preference.[5] Users' preferences can spread through the social relations and influence other users. Many studies have successfully exploited users' social information to recommendation systems to minigate the data sparse issue. In the previous passage, we introduced the collaborative filtering (CF) algorithm commonly employed in the recommendation system. Unlike the CF model, social recommendation system has two inputs: the user's rating information and the user's social data. Most of the existing social recommendation systems choose CF model as the basic model to develop their own framework, and integrate the user's social information into the CF model. Provided with the social relation data and a data set of user behavior. The social network defines the relationship between users, while the user behavior data set defines the historical behavior of users. The simplest algorithm is to recommend the user a collection of items that friends like. At present, social recommendation system has attracted wide attention of researchers, and many algorithms have been proposed and achieved great success.

1.3 Prior Works and Limitations

In the early attempts, most social recommendation systems tried to integrate social information into matrix factorization. Their common theoretical basis is based on the social homogeneity attribute: the preferences of users tend to be related to or influenced by their friends. According to this theory, MF-based social recommendation methods could be divided into the follwing categories: co-factor decomposition methods [6, 7], integration methods [8, 9] and regularization methods [10]. The co-factor decomposition method assumes that there is a rating latent space and a social latent space, in which the users' preference vectors are similar through the decomposition from rating matrix and social matrix. One of the representative method in co-factor decomposition methods is SoRec [1], which can learn users' preference vector from the rating matrix and social matrix at the same time. The essential theory of the integration methods is that the user should share similar ratings with their friends. Then the missing ratings are learned through a linear function where the variable is the rating matrix and the friend list of

the users. The assumption of regularization methods is that the user's preferences need to be as close as his or her social relations' preference. Regularization means the strategy that forcing user's latent representation vector to be approximation to the preference of his/her social associates. SocialMF [11] as one of the representative method of regularization methods forces the user's preference vector to approximate to the average preference vector of his or her social associates.

However, part of researches have found that the strength of social relations also influence the prediction accuracy of the social recommendation system. Users with strong associations are more likely to be real-life friends. They are in a tendency to share similar interest with the target user compared with the weak associations. Therefore, when exploit social relations to the recommendation system, all social relations cannot be treated equally. Otherwise, some weak social relations may become the social noise to recommendation system which will degrade the recommendation performance. Many studies have attempted the evaluate this particular influence value of different social relations. For example, [32] measured the social influence value according to the similarity of user behaviors. SPF[33] proposed a social Poisson Factorization Probabilistic model that estimate the "influence" value in regard to different social relations and learn the missing rating through different "influence" value factors.

Most of the above methods directly use the adjacency matrix, and combine the adjacency matrix decomposition with the rating matrix decomposition to learn the user's preference vector. The development of network embedding brings new possibilities to the social recommendation system. By combining social recommendation system with network embedding technology, we can learn more information from social network to improve the accuracy of recommendation system. For example, while extending MF, CUNE[9] utilized network embedding to learn the top-k semantic friends of users, and makes recommendations to users with these reliable implicit social relations. IF-BPR[15] utilized network embedding to learn from heterogeneous information networks and identified users' reliable social associates to reinforce the recommendation system. Although the above methods can mitigate the data sparsity dilemma, the social recommendation system still faces some challenges. For example, not all social information can be directly applied to a recommendation system. Poor quality links can make social networks noisy. The indiscriminate adoption of these social relationships will affect

the performance of the recommendation system. Besides, users' relations are also faced with the problem of data sparsity. These problems have not been well addressed.

1.4 Objectives

Most existing studies failed to capture the complex influences of social relations on user preferences and detect the noises in some social relations. In this research, we develop an advanced recommendation framework that integrate the user-item interactions and user-user interactions into the feature learning process. To obtain a better representation of the user feature, we designed several deep learning-based modules to generate meaningful embeddings from learning process. In particular, we introduced a social-aware representation learning framework which exploit attention mechanism, and propose the following methods to solve the above problems. As the neural network embedding method has shown its powerful ability in numerous recommendation tasks as a result of its capability to distill the high-level semantic characters from unprocessed input. We came up with utilizing network embedding method to distill the latent semantic informations hidden in the users' social network structure. Through network embedding, a high-level representation could be achieved depend on the users' social connections. This approach could be viewed as a pre-train strategy. Then, we put the obtained user embeddings into the downstream framework to enhance the representation learning of the user. To solve the problem that users are only interested in certain aspects of social friends, we designed a factor-level attention model, which refines users' characteristics into different factors, and improves the accuracy of the model by learning users' preferences for specific factors. Besides, although much work has been done to explore the influence of different social neighbors on user preferences, unlike the above summarized regularization methods and other methods based on user information to mine the strong and weak relationship between users, we use the self-learning ability of neural network to learn the influence of different users. Especially, as attention network could assign different weights to different factors to learn more informative features, we designed a module which dependent on attention mechanism to learn the different contributions of different social neighbors to user preferences. Another important feature is that the

attention weight can vary from context to context. Based on this feature, the influence value between users could be obtained adaptively. When giving a different item, the attention neural network could help us learn a different influence value for different social neighbors. To tackle the issue of social data sparsity, heterogeneous information network (HIN) is proposed to find users with similar preferences, and these social relationships are used to enrich the social information of users. To enhance the interpretability of the recommended model, we also analyze and demonstrate how to improve the interpretability of the system. To validate the effectiveness of our proposed methods, extensive experiments were carried out on several public datasets. We also carried out several experiments to verify the importance of several modules for learning better user representations.

1.5 Outline

In this section, we gave an introduction of the structure in this paper. Chapter 1 gave a brief introduction about this work. Chapter 2 summarized the related works about recommendation system. Chapter 3 introduced some technical background about this paper. Chapter 4 presented the general problems and research methods of this paper. Chapter 5 introduced an adaptive attentive model for Social Recommendation. Chapter 5 introduced a recommendation system model based onHeterogeneous information Networks. Finally, Chapter 5 made a summary about this work and gave a conclusion for the future work.

Chapter 2

Existing Approaches in Recommendation System

2.1 Social Recommendation

So far, data sparsity and cold start issues remain important factors that have plagued the performance of collaborative filtering systems, which have contributed to the emergence of social recommendation studies. Based on the traditional recommendation system, social recommendations [10] are algorithms that incorporate the user's social information (such as friend relationship and trust relationship) into the recommendation methods as an important influencing factor to tackle the data sparsity issue. This research has practical implications. In the real world, people's lives are often strongly influenced by their friends, such as choosing new products. This theory has also accelerated the study of social recommendations. Many studies [3, 11–22] have tried to exploit social signals to reinforce the recommendation system and have achieved success. All of these studies are dependent on the same hypothesis that users tilt towards to share similar interests for their social relationships. These studies can be divided into three categories[10]: co-factorization [11, 12, 18, 22], ensemble [17] and regularization [23] methods. In particular, [18] proposed a probabilistic factorization model called SoRec that co-factorize the "user-item rating matrix" and "user-user trust matrix". In SoRec, Ma assumed that the users' latent factor in social space is the same with rating space. Therefore, the users' rating matrix and trust matrix can be decomposed simultaneously through the factor decomposition model to learn the user's latent features in the latent space. Different from SoRec, [11] proposed another factorization-based model called SocialMF. SocialMF argues that a user's latent feature is depended on the user's social

neighbors. Therefore, the weighted average sum of the user's friends' latent vector should be as similar as possible to the user's latent vector. Distinct from co-factorization methods, ensemble methods argued that there are two factors that influence a user's decision on the given item, one is the user's inherent interest factor, the other is the influence factor given by their social neighbors. According to this reason, [17] proposed a probabilistic framework called RSTE that integrate the preference of users and their trusted friends through an assemble parameter. The assemble parameter controlled how much information the model should learn from these two factors. If the assemble parameter equals to 1, the method could be viewed as Matrix Factorization. If the assemble parameter equals to 0, RSTE could be transformed to SocialMF. However, the experiment results have proved that assemble these two factors will perform better than only utilize either the user's inherent interest factor or the influence factor given by their social neighbors. The studies mentioned above are relied on a prevalent hypothesis that users have similar interest to their social neighbors. However, this assumption may not consistently appropriate to any social recommendation scenarios. Some friends may have completely different interest with users. Therefore, [23] proposed a regularization methods called SR. It could constrain the model based factorization methods through a parameter called social regularization term. This social regularization term could be viewed as a similarity function that imply how much a user's preference is related with his social neighbors.

The above works are based on explicit feedback, that is, the task of recommendation system concentrates on forecasting the missing rating between the user and the item. Nevertheless, many websites cannot provide specific rating. Therefore, some recommendation algorithms utilized BPR to improve the rating-based model, so that the recommendation task can be applied to the implicit rating. [20] proposed a ranking-based model called SBPR. The basic idea of SBPR is that user tilt towards to give a higher rank to the item that the user's friends has bought or clicked. [24] proposed Expectation-Maximization algorithm model called TBPR which extend the SBPR model. In TBPR, the model first utilized neighborhood overlap algorithm to identify the strong and weak connections in the users' social network. Then, it utilized the BPR idea to strengthen the objective functions. Other work has also

attempted to examine the impact of social network connections on recommendation systems. For example, [25] proposed a model called SocialMF-TM. SocialMF-TM first calculated a trust score between users. Then, it utilized this trust score to extend the SocialMF model. [26] attempted to assign different weight to different users in a different way. These weights are obtained by calculating the user's similarity to other users. Similarity is measured by the degree of overlap of the bought or clicked items of the user, or the degree of overlap of the user's social interactions. [27] proposed a network embedding(NE)-based model called CUNE-BPR. The model first calculate the similarities between users through network embedding. Then, CUNE-BPR extend SBPR by assigning different regularization terms to different friends. However, while all of these works mentioned above have been successful, they have not taken into account the fact that the regularization term will change as the user facing with different given items. The proposed approach focused on solving this limitation.

2.2 Deep Learning-based Recommendation System

Due to the rapid development of deep learning technology, research on deep learning-based personalized recommendation technology has attracted the attention of many scholars. Some recent research has shown that in personalized recommendation, applying neural network has certain advantages. For example, [28] has shown that the traditional collaborative filtering method unable to manage huge data, and utilize the Restricted Boltzmann Machines(RBM's) in the recommended system. This paper also validate that it could realize a better prediction result than the previous traditional methods(e.g., Matrix Factorization). This is the first attempt to apply neural network to recommendation system. Besides, compared with the traditional recommendation technology, the application of deep learning technology can better understand users' preferences, items' characteristics and users' and items' linear and nonlinear interactions. It is of great significance to integrate deep learning into personalized recommendation system to optimizing the deficiencies of the existing works and improve prediction quality of recommendation system. Existing methods are mainly including two categories, which are deep learning-based collaborative filtering and deep hybrid models for recommendation. The following section will give a summary of the relevant

progress of the existing works in deep leaning-based recommendation system.

2.2.1 Deep Learning-based Collaborative Filtering

The development of deep learning technology provides a new research direction for the traditional collaborative filtering technology to alleviate the problems of data scale and scalability. Many studies [29–38] has tried to integrate MLP, AutoEncoder, CNN, RNN, Attention Mechanism, RBM, and other deep learning methods into the collaborative filtering model. For example, Neural Collaborative Filtering(NCF) [39] is the representative work that combine the matrix factorization with MLP to form a generic and extensible model. NCF overcame the problem that MF can only learn the linear interaction between users and items, and utilized MLP to enhance the non-linear interaction learning between users and items to improve the recommendation quality. In collaborative filtering recommendation based on Auto-Encoder, [40] is the representative work. It proposed a collaborative noise reduction auto-encoder(CDAE), where the input is corrupted by Gaussian noise and reconstruct the input by an auto-encoder. Taking the advantage of CNN, many studies utilized CNN to learn feature representation of unstructured data. For example, Wang et al. [41] proposed to adopt CNN to learn the latent features of image and map them to the same space to calculate the similarity and recommend them to target users who might be interested in the image. It has obtained the good performance. [42] designed ConMF by combining text representation which learned via CNN with MF to enhance the recommendation performance. Since RNN is suitable for processing sequential data, it is appropriate for extracting useful features from user behaviors with temporal characteristics. In the real world, users' interest change over time. For example, a person's interests after working may be different from those of his school days. [43] designed a session-based recommendation system based on GRU to predict what item the user will interact next in a time session. [44] proposed a LSTM-based model and compared it with standard nearest neighbor method and matrix factorization method. Experiment results has shown that LSTM is much better than other methods in terms of short-term prediction. The attention mechanism can be considered as a feature extractor, which can effectively filter out the non-informational features in the input data, which are generally considered as noise data and will

have side effects on the model. The attention mechanism is an intuitive but very effective technique that can be integrated into original models to deal with the side effects of noise data. For example, [45] integrated the attention mechanism into the LSTM model that enables the neural network to simultaneously process long time and noisy input data, which helps the model to remember more effective information. The success of attention mechanism in neural network also promotes the research of recommendation system based on deep learning. The advantage of attention mechanism lies in that it can help filter out unrelated information and has good attentive ability. Therefore, there are many attempts to apply attention network to recommendation system, among which [46] is the most representative one. In recent two years, generative Adversarial Networks (GANs) [47] is considered as the most successful technology in the field of deep learning, and has made important progress in image recognition [48] and other fields. GAN is different from other neural network structures, it is composed of two game neural networks, generator and discriminator, which played a minimax game. The generator is mainly responsible for learning the distribution of real samples and generating fake samples to spoof the discriminator. At the same time, the discriminator tries to distinguish whether the input sample is real or not. This process will continue until the discriminator was unable to distinguish between the generated sample and the real sample. Now there have been a number of works that applied GAN in the recommendation system and made progress. Some works alleviates the problem of data sparsity by generating user interaction information through GAN. [49] is the study that attempted to apply the GAN to the domain of information retrieval. It proposed a framework based on adversial learning, named IRGAN, to solve the problem of information retrieval, and this model could also be applied to the recommendation field. The main method of IRGAN is that, given a user, let generator generate items that the user may purchase, and let discriminator distinguish the probability distribution of the generated item from the probability distribution of the user's actual purchase of products, until the generator can finally capture the true probability distribution of the user's preference for items. However, the generator in IRGAN each time will generate a separate index value, which may be exactly the same as the real data, thus affecting the training effect. Therefore, different from IRGAN, [50] proposed a new gan-based framework called CFGAN. In CFGAN, the generator generated the user's possible purchase vector each time instead of a single

item index, and the discriminator will distinguish the generated purchase vector from the real purchase vector.

2.2.2 Deep Hybrid Models for Recommendation

Another advantage of using a deep neural network to build a recommendation system is that the neural network structure is highly extensible. Different neural network modules can be integrated into the same system, which making the learning ability of the model more powerful. Many studies have attempted to integrate different neural network structures into a single model to enhance the capability of the recommendation system. For example,[6] proposed a model (CKE) that integrates CNN and auto-encoder to enhance the representation learning. CKE can effectively utilize different input information sources. For instance, CNN-based embedding method is used to learn the features form visual source, while auto-encoder is used to learn the features contained in text information. Such a hybrid model is more expressive than a model with a single structure.

2.2.3 Personalized Recommendation Based on Graph Neural Network

Graph Neural Networks(GNN) [51] is a new type of deep neural networks based on deep learning and graph analysis. It is a new expansion form of deep learning theory and technology in the graph field. Graph neural network allows non-Euclidean structure data directly be used as the input of deep neural network, which has good representation ability and interpretibility. In recent years, graph neural network has become a hot spot in deep learning research. Graph Learning Based Recommendation System is a new recommendation system based on Graph Learning, which uses the ability of node representation learning in network embedding technology. In the past few years, there have seen extensive research on Graph Learning Based Recommendation System. Most of the data in recommendation system has a graphical structure. In practice, learning the relationship between such graph type data has important research significance for the recommendation system. Learning the interaction between the user and the item is the main purpose in a recommendation system. It is found that users and items

can naturally form a user-item bipartite graph based on their interaction information. Each edge between the user and the item represents that the user has bought or clicked the item. According to the structure of the bipartite graph above, Li and Chen [52] proposed a method based on link prediction, transforming the recommended task into link prediction, and predicting the possible unknown links based on the known links in the user-item bipartite graph, and thus find the unknown items that users may be interested in.

Chapter 3

Technical Background

3.1 Model-based Recommendation System

The generally employed method in recommendation systems is called Collaborative filtering(CF) [53]. The core idea is utilizing the user-item interaction matrix to find similarities between users or items and make recommendation based on these similarities. At present, there existed mainly two types of CF-based approaches in Recommendation System(RS): memory-based recommendation system [54] and model-based recommendation system [55]. In this research, we focused on the model-based recommendation system, especially the proposed methods were built on the basis of the Matrix Factorization(MF), which is commonly utilized in memory-based recommendation system. Figure 3.1 give an example of the user-item interaction matrix used in our paper, which denoted as R.

user 1	user 2	user 3	…	user n	
1	2.5	?	…	2	item 1
4	?	?	…	3.5	item 2
3.5	3	?	…	4	item 3
…	…	…	…	…	
3	1.5	?	…	?	item m

FIGURE 3.1: An example of the user-item interaction matrix.

As shown in Figure 3.1, the user's past behaviors were stored by this sparse matrix. Each columns represents a user's preference on different items. The value here can be a score. Also, it can be 1 or 0, which representing that whether the user has clicked or bought that product. What Matrix Factorization(MF) do was tried to forecast undiscovered ratings in this rating matrix. The core idea of MF assumes that there existed two low-dimensional feature matrix U and I in a latent space that represent the users' preferences and the items' characteristics, respectively. These two small matrices could be obtained through decomposing the matrix mentioned above. This method is also known as the Latent Factor Model (LFM) [56]. As shown in Figure 3.2, the main idea is to link the user's perference to the target item's attributes through the underlying factors.

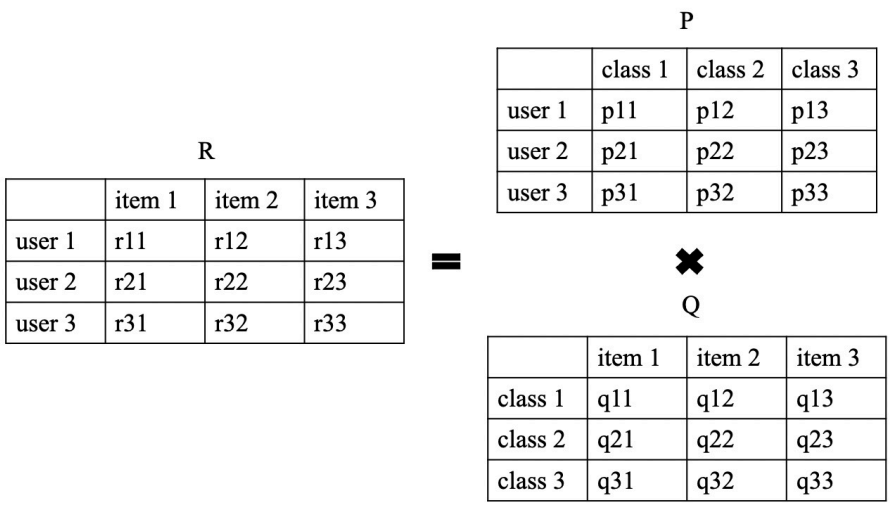


FIGURE 3.2: An example of matrix factorization: decompose the rating matrix R into two smaller matrices P and Q

Let a k dimension vector v_u and v_i represent the user's and the item's feature, respectively. Each latent factor in this feature can be viewed as an aspect that describes an item or a user's interest. MF assumes that the user's predilection on the item was the sum of preferences for various aspects of an item. It can be calculated using the dot product of the underlying factors as following equation:

$$r_{u,i} = \sum_{f=1}^{k} v_{u,f} * v_{i,f} \tag{3.1}$$

To find such matrix U and I, the model first initializes two matrices and then calculates the difference between the result of dot product between U and I and the target value in user-item matrix. By this way, the recommendation problem could be viewed as an optimization problem with loss function and constraints.

3.1.1 Learning Model of Matrix factorization

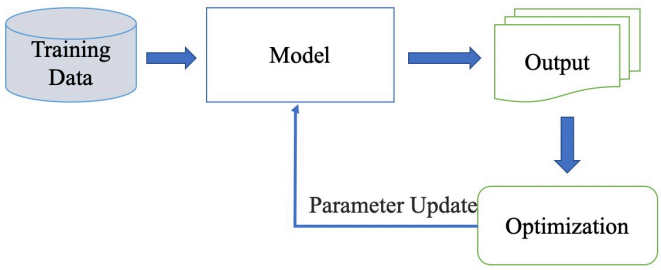


FIGURE 3.3: An example of machine learning process of MF.

Figure 3.3 showed an example of the learning process of a recommendation system. The model here can be any models used in recommendation system, for example, the matrix factorization model. As the recommendation task is reformulated as an optimization problem, we can use least squares error as the optimization method to train this recommendation system, which defines as:

$$argminL(\theta) = \sum(r_{ij} - \widehat{r}_{ij})^2 = \sum(r_{ij} - \sum_{f=1}^{k} v_{u,f} * v_{i,f})^2 \tag{3.2}$$

Usually, Stochastic Gradient Descent(SGD) [57] is commonly adopted algorithm to optimize the above function to solve such constrained optimization problem to help us quickly reach the optimal value.

3.1.2 Matrix Factorization using Bayesian Personalized Ranking

In practice, there is usually the case that a user clicks to view an item, but does not end up with a score, which is called explicit feedback. In fact, if

the user clicks to view the item, it can be assumed that the user interested in some aspect of the item. This is called implicit feedback. In fact, the explicit feedback data is quite rare in recommendation system with the implicit feedback data accounting for most of it. The least squares error mentioned above is not suitable for recommendation scenarios based on implicit feedback. Herein, we introduce an optimization criterion called Bayesian Personalized Ranking(BPR) [58] which usually used in implicit feedback-based recommendation systems.

	item 1	item 2	item 3
user 1	?	+	+
user 2	+	?	+
user 3	+	?	?
user 4	?	?	+

	item 1	item 2	item 3
user 1	0	1	1
user 2	1	0	1
user 3	1	0	0
user 4	0	0	1

FIGURE 3.4: An example of user implicit feedback.

The purpose of BPR lies in providing users with a ranked list of items. The method based on explicit feedback predicts a certain score for an item to reflect the user's preference for that item. As shown in Figure 3.4, all observed interactions between users and items are marked as positive classes, and the unobserved interactions are marked as negative classes in implicit feedback-based methods. In BPR method, supposing that the user u clicked item i when he was providing with the items i and j simultaneously, we will get a triple pair denoted as $< u, i, j >$. It pointed out that user u goes for item i more than item j. According to the above assumption, BPR criterion is denoted as:

$$\sum_{(u,i,j)\epsilon D_s} ln\sigma(\hat{x}_{uij}) - \lambda_\Theta ||\Theta||^2 \tag{3.3}$$

wherein, $(u, i, j)\epsilon D_s$ denotes that user u prefer item i more than item j, λ_Θ are the regularization parameters used in the model. $\hat{x}_{uij}$ could be determined as follows:

$$\hat{x}_{uij} = \hat{x}_{ui} - \hat{x}_{uj} \tag{3.4}$$

$\hat{x}_{ui}$ can be predicted by matrix factorization. Optimizing BPR criterion is similar to other optimizing process in Machine Learning. Stochastic Gradient Descent(SGD) willed be employed during optimizing process to find the

optimal hyper-parameters. The training process is shown in the following algorithm:

Algorithm 1 BPR optimization.

Input: initialize Θ

procedure LearnBPR(D_s,Θ)
repeat
  draw (u,i,j) from D_s
  $\Theta \leftarrow \Theta + \alpha(\frac{e^{-\hat{x}_{u,i,j}}}{1+-\hat{x}_{u,i,j}} + \lambda_\Theta * \Theta)$
until convergence
return $\hat{\Theta}$;
end procedure

In our research, we will also utilize BPR method to optimize the proposed model.

3.2 Deep Neural Network

Deep Learning [59] is a subset in Machine Learning domain, that mimicking the way that our human brain works. To act like a human's brain, deep learning algorithms develop a multi-layered structure called neural networks. First, deep learning algorithms create a set of "neurons" and connect them so that they can send messages to each other. These neurons are then asked to solve a real problem, and it will try to solve the problem over and over again. Each time strengthening the connections of the neurons that lead to success and reducing those that lead to failure. By doing so, the neural network learns to solve this problem. With neural networks, we can solve many problems such as classification or regression. In fact, the recommendation system could be viewed as a binary classification task [60] of which main task is trying to predict that if a user would like or dislike an item. Therefore, we can use deep neural network to solve this binary classification problem.

3.2.1 A Multi-Layer Perceptron

As shown in Figure 3.5, this is a typical feedforward neural network architecture called a multi-layer perceptron consists of three layers. It is a network structure composed of many neurons connected together. Neuron is the basic computation unit in a neural network, which receives input datas from

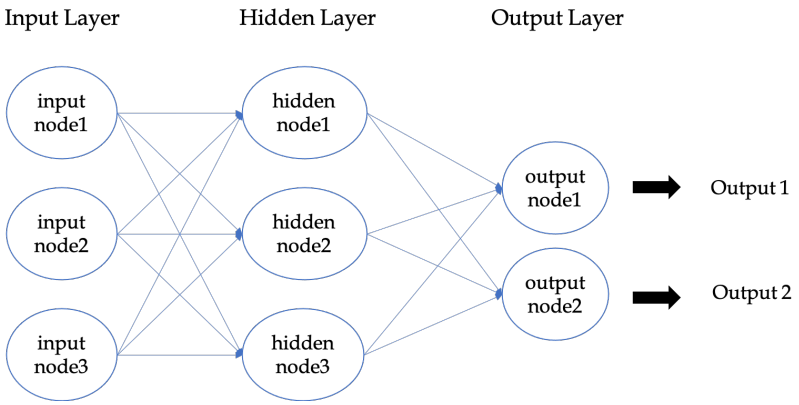


FIGURE 3.5: The network structure of a full-connected feedforward neural network

other nodes or external sources and apply a function on these data. Each neuron has a weight w and a bias b. Then, the output of a single neuron is calculated by $f(w_1 * x_1 + w_2 * x_2 + b)$. As shown Figure 3.6, an activation function f is occasionally exploited to weight the sum of its input. Without activation functions, no matter how many layers of neural network are superimposed, only the linear relations between input and output data can be learned. Therefore, the activation function plays an important role in learning the nonlinear relations between input and output data. As most data in the real world is nonlinear, it is important for neural network to improve the nonlinear representations learning ability. [61] In practice, the commonly used activation functions are sigmoid and ReLU functions.

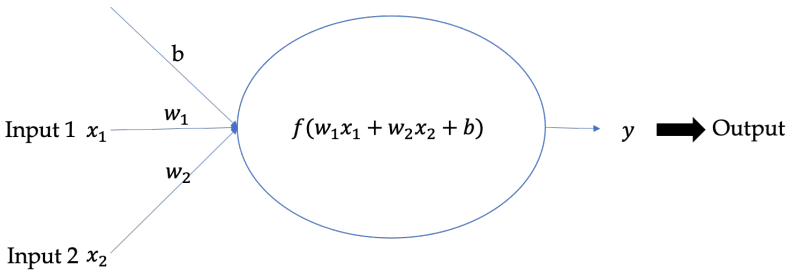


FIGURE 3.6: A single neuron

Neurons are lined up to form a layer. A typical neural network can be composed of multiple layers. For example, in Figure 3.7, this simple multilayer perceptron has three layers. The first layer in the feedforward neural

network is called an input layer, which took the response to pass the train data x to the hidden layer, and there is no computation in this layer. It should be noticed that the size of neurons in the input layer should have the same size with the dimension of the input data x.[62] The output layer will output a vector y that represents the result of learning through the whole neural network. For example, in the classification model, each neuron in the output layer represents a different class. The value of each output neuron gives the probability of whether the input data sx belongs to a possible class. To get this result, neural networks need to perform some mathematical operations, which performed by the hidden layers between input and output layer. In this example, there is only one hidden layer. In general, deep learning means that there are many hidden layers in the neural network. Given a set of features $x = (x1, x2, x3.....)$ and the target y, the neural network will the linear and nonlinear relationship between x and y.

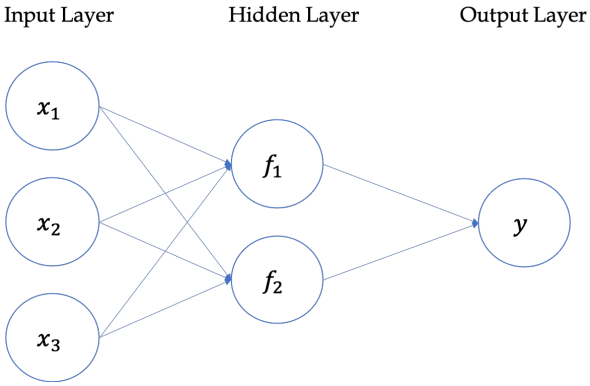


FIGURE 3.7: An example of a three-layer perceptron

3.2.2 Learning Process of a Neural Network

The training of the model mainly consists of three steps: forward learning, calculating the loss, and backward propagation. In the first step of training, we put the input data into the model. Then, in each layer, the input data will be multiplied with the weights and added with a bias. Finally, we can get the output of the model. Before the training of neural networks, loss functions need to be defined. In a deep learning task, defining the loss function is an important step in ensuring that the model works as expected. Neural networks can perform a variety of tasks, such as predicting continuous values

or classifying discrete categories. Different loss functions are defined for specific tasks. To compute the loss function, we must traverse every data x in the dataset, and predict y for each sample, and then calculate loss error by the difference between the predicted value and the real value. For example, in our recommendation task, we can use Equation 3.2 as the loss function. At the beginning of training the neural network, the weights should be randomly initialized. The training process is equivalent to minimizing the loss function by adjusting the weight. There are many algorithms for function optimization. The simplest way to optimize the loss function is utilizing stochastic gradient descent algorithm (SGD). Generally, the gradient is the maximum directional derivative of the function at a certain point, and the function has the maximum rate of change along the gradient. Since the function has the maximum rate of change along the direction of gradient at a certain point in the variable space, it is natural to reduce the value of the function along the direction of negative gradient when optimizing the objective function, so as to achieve our optimization goal. After defining the objective function, we take the derivative of the objective function and update the parameters with gradient descent. However, the problem of stochastic gradient descent is that, from the perspective of the number of iterations, there are so many iterations, and the search process in the solution space seems to be blind, resulting in too high computational complexity of the update method. Another problem with SGD is that there is so much noise that SGD does not always go in the direction of overall optimization for each iteration. To tackle this issue, we can use mini-batch gradient descent algorithm. To be specific, at each step of the algorithm, a small batch (Mini-batch) samples are randomly selected from the training set of the total sample for calculation. The batch size used to estimate the gradient is another hyper-parameter that needs to be tuned. As the powerful representation ability, neural networks can approximate any functions as long as there are enough hidden neuron layers. However, the exact number of hidden layers is impossible to calculate and can only be adjusted in practice by constant experimentation.

3.3 The Basic Framework of Recommendation System based on Deep Learning

Recent studies have show that deep learning is useful in information retrieval and personalized recommendation. Compared with the traditional recommendation methods, utilizing deep learning can better generate the feature of the user's interests and the item's characteristics. In addition, deep learning can also promote the learning capability of the non-linear relations between input features in the recommendation system. Therefore, deep learning has attracted much concentration in both scientific and industrial areas. The basic framework of the recommendation system based on deep learning is displayed in Figure 3.8.

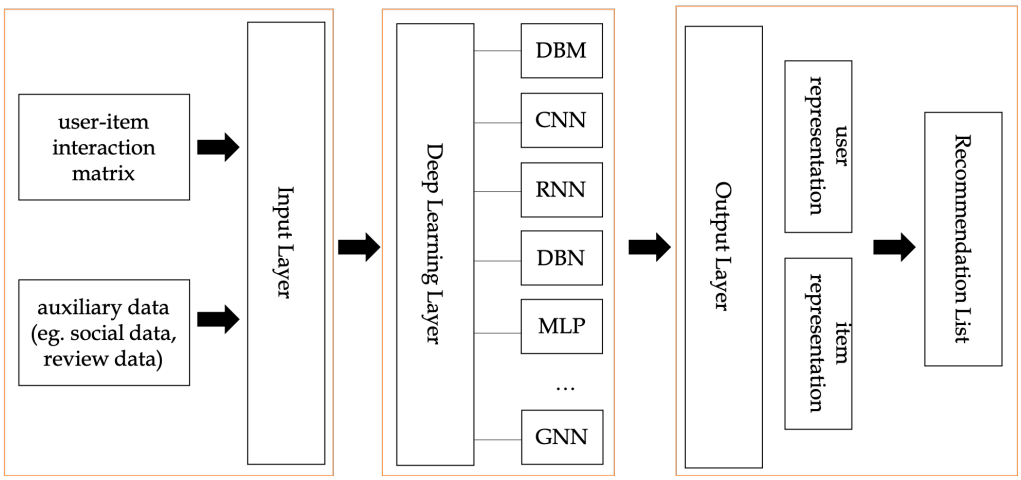


FIGURE 3.8: The basic framework of RS based on DL

As shown in Figure 3.8, the whole framework of deep learning-based recommendation system is divided into input layer and deep learning model layer and the final prediction layer. The input layer mainly includes user behavior data and other auxiliary data. Among them, user behavior data can be explicit feedback data, for example, ratings, comments, etc., or implicit feedback data, for example, clicks, favorites, browsing, etc. Other auxiliary data mainly include users' social data, contextual data and so on. On the basis of acquiring the input data from input layer, the deep learning model exploits appropriate manipulation to process the data according to the characteristics of different deep learning technologies and in combination with the actual application scenarios, so as to retrieve the relevant inherent factors of users

and items. The final output layer will generate the final recommendation list of items through similarity calculation or inner product or other operation based on the inherent features of users and items output through the hidden layer.

3.4 Embedding and Representation Learning

Theorem 1 *Representation learning: To improve the accuracy of machine learning systems, we need to translate input information into valid features, or more generally, representations. If there is an algorithm that automatically learns effective features and promotes the performance of machine learning model, then this learning will be called Representation Learning.*

If a prediction model is directly based on the underlying features, it will lead to excessive demands on the predictive model capability. If we can have a good representation that somehow reflects the high-level semantic characteristics of the data, then subsequent machine learning models will be built relatively easily. Within presentation learning, there are no clear criteria for a good representation, but it usually has the following advantages: A good representation should have a strong representation that a vector of the same size can represent more information. A good representation should make subsequent learning tasks simple, that is, it needs to contain higher level semantic information. In traditional machine learning, we often use two ways to represent features: Local Representation and Distributed Representation. Take color representation as an example. One way to represent color is to name different colors with different names. This representation is called one-hot local representation, also known as discrete representation or symbolic representation. A local representation can usually be expressed as a vector. Compared with local representation, distributed representation has much better representation capability than local representation, and the vector dimension of distributed representation is generally lower. We just need a dense vector in three dimensions to represent all the colors. And the distributed representation makes it easy to represent new color names. In addition, the similarity between different colors is easy to calculate. To learn a good high-level semantic representation, it usually needs to start from the underlying features and go through a multi-step nonlinear transformation.

The advantage of a deep structure is that multiple consecutive linear transformations that can add features are equivalent to reusability, thus exponentially increasing expressiveness. Therefore, the key to representation learning is to construct multi-level representation with certain depth.

In recent years, considerable success of the application of deep learning has been accomplished in different fields. A remarkably successful application of deep learning is embedding, which is a method of representing discrete variables as continuous vectors. We noted that the latent features in Latent Factor Model are similar to the embeddings in neural network. Therefore, it has been found that Deep Neural Network(DNN) can be utilized to help realize matrix factorization, and DNN has been successfully applied in the recommendation system. We will introduce the embedding methods and application in the recommendation system in the following section.

3.4.1 Embedding Matrix

In machine learning, data will be required specific processing before it can be fitted to a machine learning model. The most widely used method to a categorical data is one-hot encoding. However, one-hot encoding usually high-dimensional and sparse. Take the Amazon dataset as an example. There will be hundred thousands of users in the dataset. This means that, if we want to represent a user using one-hot encoding, the encoding vector will have hundred thousands dimensions and only one integer in this vector is one. In such a big dataset like Amazon, this method is not efficient in computational. Beside, We found that there is a certain similarity between different users or items in the multi-dimensional latent space. Embedding matrix is one way to represent these relationships. In deep learning-based model, we utilized an embedding layer to map these categorical datas as continuous vectors.

For example, we have five users $< u1, u2, u3, u4, u5 >$ in our Amazon datasets. First, we need to assign a unique index to each user as the user's identity information. The input data of the embedding layer will look like the following list:[0,1,2,3,4]. Then, an embedding matrix will be created. In the embedding matrix, each row represents the latent feature of the user corresponding to the index. In the previous section, we have introduced the Latent Factor Model, in which each dimension represents the factor on different aspect. Therefore, the length of the factor here can be freely specified

and is referred as the vector dimension. The commonly used vector dimension can be [128,64,32]. For easy representation, we assume that the vector has a length of 5, and its embedded matrix is shown as follows:

user1	12	45	43	26	78	52	…
user2	43	25	3	43	78	78	…
user3	34	56	23	12	78	74	…
user4	56	54	23	5	78	43	…
…	…						

FIGURE 3.9: An example of the user embedding matrix.

As shown in Figure 3.9, a user embedding matrix is a list of all users and their corresponding embeddings. Similarly, the embeddings of items can also be found through an item embedding matrix. This embedding matrix here is nearly the same as the user/item latent feature matrix that introduced in MF model. Compared with the traditional matrix factorization algorithm, in addition to embedding achieved by the embedding layer, various latent features can be added to promote the prediction accuracy. In conclusion, embedding was utilized to transform the high-dimensional vector into the relatively low-dimensional space and making sparse vectors easier for machine learning. Also, through putting the input of related semantic information tightly in embedded space, embedding could catch rich semantic information from the original data. We will explore it further in our paper.

3.4.2 Network Embedding

Network Embedding is a method with the purpose of mapping nodes to a k-dimensional latent space to learn a low-dimensional, dense, real-valued representation. The learned node representation is conducive to computation and storage. The traditional method used adjacency matrix to store graph

structure, which only records the information of nodes' neighbors of 1 degree. The dimension of the adjacency matrix is very high, which requires n*n space complexity to store. More importantly, similar structural equivalence of network nodes should be required to share similar embedding to save the network topology. The common representation of network structure does not apply to the deep learning method, because the interrelationships between network nodes cannot be divided into independent vectors. The ML method usually assumes that the sample can be partitioned into independent vectors. Therefore, network representation learning is important in the application of deep learning to network analysis.

In the conditions of development of network embedding, many successful algorithms, such as DeepWalk, node2vec, SDNE, have been occurred. Different network embedding techniques have different effects for different downstream tasks. At present, the most commonly used network embedding algorithm is node2vec. Its algorithm is shown as follows:

Algorithm 2 The node2vec algorithm

Input: $G' = (V, E, \pi)$
1: Initialize *walks* to Empty
2: **for** *iter* = 1 to r **do**
3: **for** all nodes $u \epsilon V$ **do**
4: *walk* = node2vecWalk(G', u, l);
5: Append *walk* to *walks*;
6: **end for**
7: **end for**
8: f = StochasticGrandientDescent($k, d,$ *walks*)
9: return f;

In the above algorithms, network was symbolized as $G' = (V, E, \pi)$. *walks* was used to store the random walk, it would be initialized to null at the beginning. The outer loop r means that r random walks are generated for each node. The algorithm mainly consists of two processes. First, traversing the graph structure and generating a random walk for each node and add the *walk* to the *walks* to save. In particular, node2vec proposed a parameterized random walk mode, and introduced two important parameters p and q, which are utilized to direct the walk. Then, the obtained node sequences will be fed into the skip-gram model to update representation with the following

objective function:

$$L = \sum_{u \epsilon V} \sum_{v \epsilon N_R(u))} -log(P(v|z_u) \tag{3.5}$$

z_u is the neighbors of node u in a walk. Given the representation of nodes, it will maximize the probability of next node is node v in the walk. Each of the nodes will be mapped to a d-dimensional representation. It's actually a matrix that has $|V| * d$ parameters. These parameters are what you need to learn during the training. Through network embedding, we can get a low-dimensional inherent representation for each node.

3.4.3 Attention Model

The Attention Mechanism was the generally exploited optimization model in neural networks[63]. The intuitive understanding of attention is that when faced with a large amount of information, people are more likely to pay attention to such points, such as the most prominent parts of a picture. It can help people filter out unimportant information from vast amounts of information and quickly locate more noteworthy information. The attention mechanism in deep learning refers to the attention thinking mode of human beings and is integrated into various deep learning models, thus achieving remarkable results.

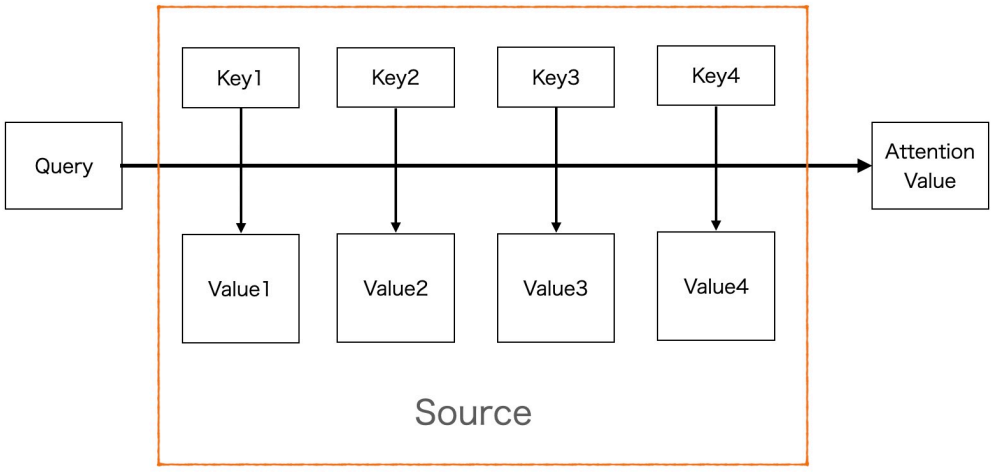


FIGURE 3.10: An illustration of attention mechanism

Figure 3.10 illustrates the theory of attention mechanism. Let's take a simple example to understand the principle. There are many books (value) in the library. In order to find them easily, we make the number (key) for the books. If we want to know more about Marvel(query), we can check out comics, movies, and even World War II books. In order to improve efficiency, not all books will be read carefully. For Marvel, comic books and movies will be read more carefully (with high weight), but World War II books will be simply browsed (with low weight). We'll get a complete picture of Marvel when we're all done. To implement the above principle, first calculate the similarity between query and key to get the weight. The weights are then normalized using $softmax()$ function to get directly available weights. Finally, weighted sum of the weights and values is calculated.

There are many different types of Attention, such as soft attention, hard Attention, self attention and so on. Soft attention is a common way to calculate the weighted probability of all keys, and each key has a corresponding weight. This is a rational way of looking at all the keys and weighting them. But the computation might be a little bit more complicated. The algorithm proposed in this paper also utilized soft attention. When doing attention, it was necessary to compute the score or called similarity between the query and a certain key. The easiest approach to calculate the similarity was using dot product or cosine similarity. Otherwise, it can be calculated by concatenating the q and the k. Another common approach is to use a multilayer perceptron, which is what we're going to use in the proposed algorithm in this paper. It could be represented as:

$$s(q,k) = v^T tanh(W_q * q + W_k * k) \tag{3.6}$$

Where, v^T,W_q,W_k are learnable parameters.

3.5 Heterogenous Information Network

Most information network analysis assumes that the object or link only have one category. In other words, the network is homogeneous, which comprised with identical variety of objects or links. These homogeneous networks usually ignore the heterogeneity of the network structure, in other terms, it simply considered one kind of relationship between objects. However, most real networks contain multiple types of interactions that could not

be easily modeled by a homogeneous network. Therefore, it is necessary to consider to utilize a special network structure to model these multiple types of interactions. This special network was called heterogeneous information networks(HINs)[64], which comprised with different varieties of objects and links. Based on this concept, it could be defined as:

Theorem 2 *A heterogeneous information network: If the total number of object types in the network is* $|A| > 1$ *or the total number of linked types is* $|R| > 1$ *in a network, the network would be defined as a Heterogeneous Information Network, alternatively, it would be defined as a Homogeneous Information Network.*

Considering a heterogeneous information network, it was appropriate to give a description of its meta-level with the purpose of better understanding the network structure. For this reason, a conception called network schema was proposed to depict the network meta-structure.

Theorem 3 *A Network Schema is a meta-template for a heterogeneous information network with object type maps* $\gamma : V \rightarrow A$ *and link maps* $\phi : \varepsilon \rightarrow R$*, denoted as* $T_G = (\boldsymbol{A}, \boldsymbol{R})$

Theorem 4 *A meta-path is denoted as a path in the form of* $A_1 \overset{R_1}{\rightarrow} A_2 \overset{R_2}{\rightarrow} ... \overset{R_l}{\rightarrow} A_{l+1}$*, wherein R denotes a composite relation between a start object* A_1 *and the end object* A_{l+1}*.*

The meta-path is the core of the entire HIN system. Different meta-paths depict different semantic relationships between objects, and the mining of such semantic relationships is the cornerstone of various subsequent tasks.

Chapter 4

Representation Learning Framework for Social Recommendation

4.1 Problem Definition

The main purpose of this paper is to integrate users' social information into recommendation system to achieve improvement on recommendation performance. The basic assumption of a social-based recommendation system is that the user's partiality are similar to their social neighbors. However, utilizing social information to make recommendation is quite complicated in practice, and many algorithms overestimate the effectiveness of directly applying social information to the recommendation system. The shortcomings of existed social-based recommendation algorithms can be attributed to following aspects: (1)Currently, most existed work directly embedding user ID into the latent space to learn the inherent feature vector. This initialization method does not take full advantage of the semantic knowledge in this social network, so it is difficult to get the most effective embedding method.(2) The openness of social networks allows users to easily build connections with other users. This connection is different from the real-world connection. Low-quality links make the social network become noisy. Most social-based recommendation models usually use these social relations indiscriminately, thus affecting the performance of the recommendation system. (3) Users' preferences have different aspects, but most work usually assumes that users are interested in all aspects of the social neighbors. (4)The influence of social relationships is not universally applicable to any context. When faced with different items, the influence changes dynamically. (5) The user's

social relations also come up against the data sparsity issue. When history data is sparse, the benefit of utilizing social relations to augment the user's profile and promote the recommendation quality is limited. For the purpose of solving the above key shortcomings of social recommendation, based on the framework of deep learning, this paper carries out concrete modeling and analysis on the social influence of users, and solves the complex situation of the application of social relations. (6) At present, it was not straightforward for recommendation system to provide the recommendation interpretation.

Therefore, a social recommendation system based on attention neural network was proposed for the purpose of solving the above limitations. Particularly, it has the following contributions: (1) To learn augmented effective inherent factors of users and accelerate the convergence rate of the proposed model, network embedding technology was exploited. (2)Through the factor-level attention model, we can learn specific inherent factors of users to promote the accuracy of the model. (3) We leverage attention neural networks to model the social dynamic influences.(4)We propose to use heterogeneous information network (HIN) to find users with similar preferences, and use these social relations to solve the sparsity problem of users' social data. (5) We propose a two-layer attention structure to enhance the interpretability of the recommendation model.

4.2 Notations and Formulation

For the purpose of this work, matrices were represented by uppercase letters (e.g., X), vectors were represented by lowercase letters (e.g., x), Latin letters (e.g., $\mathscr{X}$) represents sets. Without any special declarations, all vectors are column vectors. Figure 4.1 illustrates our input data in the social recommendation task. In this task, suppose there were n users and m items, which were denoted as $\mathscr{U} = u_1, u_1, ..., u_n$ and $\mathscr{I} = i_1, i_1, ..., i_n$, respectively. The l-th set of users $\mathscr{F} = u_{k_{f1}}, u_{k_{f2}}, ..., u_{k_{fl}}$ represent user u_k's friends set. Among the input data, we have two kinds of observed data, which are user-item interactions and user-user interactions. In general, $R = [r_{ij}]_{nxm}$ represents the user-item interactions and G represents the social network of users. Different from other work based on explicit feedback, the following equation demonstrate how we convert explicit feedback from users into implicit feedback:

$$r_{ui} = \begin{cases} 1, & if\ the\ user\ u\ has\ rated\ the\ item\ i \\ 0, & otherwise \end{cases} \tag{4.1}$$

It should be noticed that $r_{ui} = 0$ does not of necessity imply the user's partiality. There were several possible reasons for $r_{ui} = 0$. It may be due to the fact that user u dislike that item i, also, it could be owing to the fact that the user have no idea of that item. Then, the task is to predict the users unobserved interactions with items with the input R and G. The model will provide the recommended item list to the a target user.

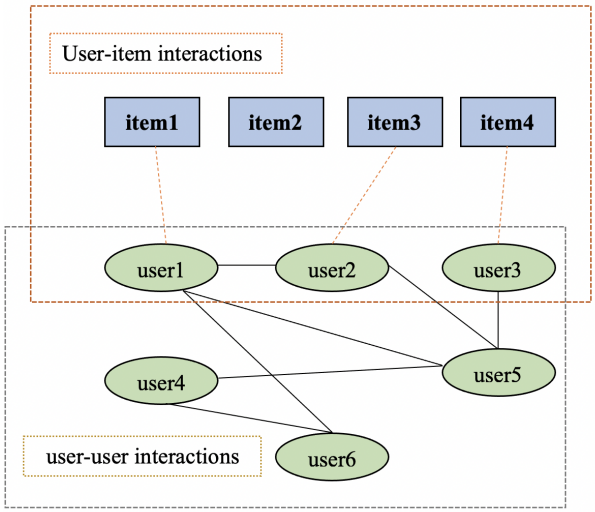


FIGURE 4.1: An example of the input data in social recommendation system.

4.3 Probabilistic Model in RLSeSys

For the purpose of this part, we presented the probabilistic model of social recommendation system based on representation learning in our paper, and described how to combine the ideas of PMF and DNN with the intention of making use of both users' ratings and social information. In general, suppose there were n users and m items, and the user-item rating matrix R. The primary objective is to find the best inherent factors to represent the users. The users' inherent factors could be symbolized by $U = [u_i]_{nxd}$, the items' inherent factors could be symbolized by $U = [v_j]_{mxd}$, where d is the embedding

size. The dot product between users and items could be employed to rebuild the rating matrix R. From the perspective of probability, the conditional distribution of observed rating matrix is given by the following equation:

$$p(R|U, V, \sigma^2) = \prod_{i}^{N} \prod_{j}^{M} N(r_{ij}|u_i^T, v_j, \sigma^2) \tag{4.2}$$

where $N(r_{ij}|v_i^T, v_j, \sigma^2)$ is the probability density function which is consistent with the Gaussian Normal Distribution. The probability density functions of user attributes U and item attributes U are gaussian distributions, which are denoted as:

$$p(U) = N(0, \sigma_U^2), p(V) = N(0, \sigma_V^2) \tag{4.3}$$

where σ_U^2 and σ_V^2 are the variance of prior noise, which can be set manually. However, our proposed model in this paper differs from the traditional PMF model is that the user's latent model U is generated from two variables. In addition to the above mentioned probability density function $p(U)$, we also learn a probability model of users' social relations through deep learning model. Finally, the latent model of users is obtained by the following formula:

$$u_i = DNN(w, u_f) + p(u) \tag{4.4}$$

where W denotes the weights used in DNN model, and u_f is the social relations of related user. It could be viewed as the user's preference is determined by the user's general preference and the user's friends preference. The primary objective of the task was to maximize the probability defined in Formula 4.2. The learning process for parameters U and V would be described in details below.

4.4 The RLSeSys Model

For the purpose of this part, a general approach in the social recommendation system was introduced, which integrating users' social information into MF-based model to make recommendations. In accordance with social theory, a user u's preference was similar or affected his/her social neighbors, who were denoted as F_u. Therefore, inspired by SocialMF [11], the user u's

latent representation of is dependent on the latent representation of his social friends $f \epsilon F_u$. This influence could be denoted as the subsequent formulation:

$$\hat{U_u} = \frac{\sum_{f \epsilon F_u} W_{u,f} * U_f}{|F_u|} \tag{4.5}$$

where, $\hat{U_u}$ denotes the latent representation of user u by summing the latent representation of user u's social friends. As our social networks in social recommendation are un-weighted graph, all the weights between user u and his friends f are equal to 1. Therefore, the above formula can be reduced to:

$$\hat{U_u} = \sum_{f \epsilon F_u} U_f \tag{4.6}$$

It should be noted that taking integrating users' social information into MF-based model does not affect the conditional distribution function of observed rating matrix. It only affects the user's latent representation. Therefore, the conditional probability function is still the same as the conditional probability in Formula 4.2. The graphical demonstration of the formula was demonstarted in the Fig.4.2.

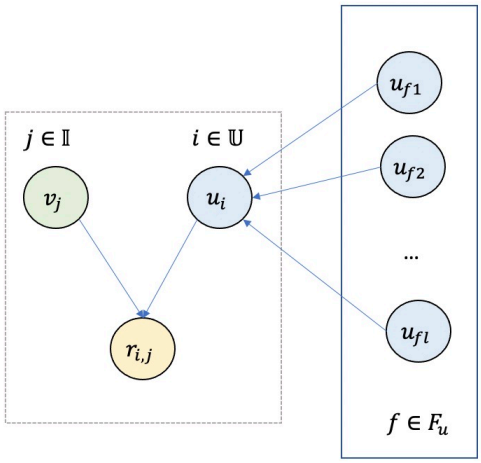


FIGURE 4.2: An illustration of integrating social information into MF-based model.

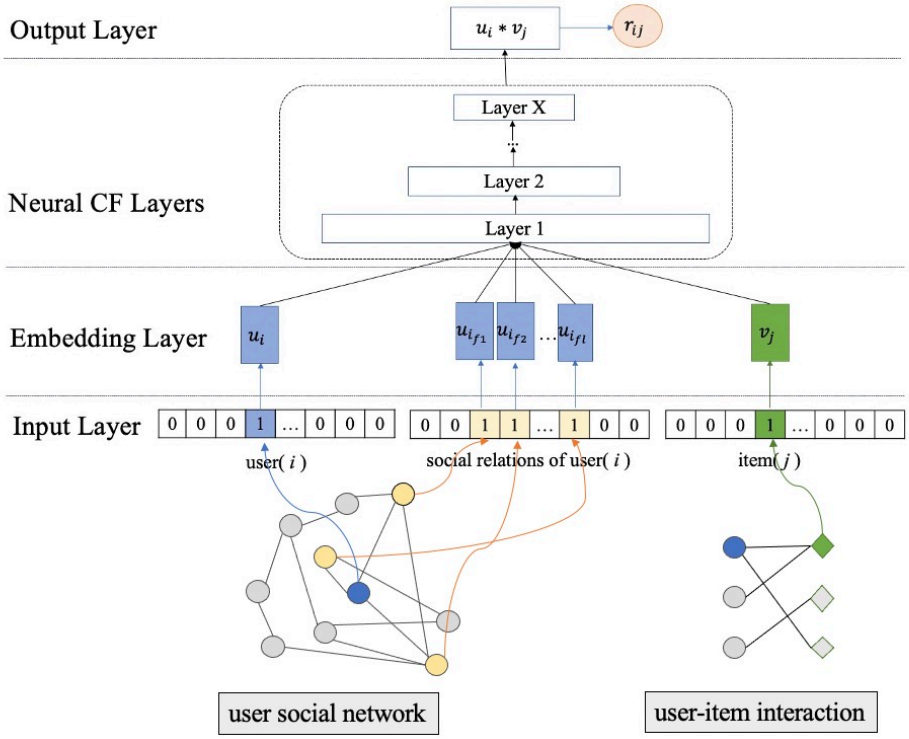


FIGURE 4.3: The overall architecture of representation learning framework in Social Recommendation.

4.5 A Whole Architecture of RLReSys

The ultimate objective was to develop a social recommendation model to depict the influence of social propagation on users' preferences. On the basis of the model-based recommendation system, we utilized deep neural network instead of the traditional model to encode the user and item to a latent embedding space and predicted the user's preference degree to a target item. A major advantage of deep neural networks over previous machine learning models is its strong advantage in feature learning. In the case of sparse training features, it can still learn new and effective features from limited feature sets. At the same time, it can learn correlations between features and discover other related features. Although the data is not preprocessed and may be noisy, it can still filter out noisy signals and learn more relevant information. This will save a lot of manual data processing. From this part, we were going to introduce the whole architecture at this study, which is shown in Figure 4.3. A model based on deep learning is proposed to learn to predict

unknown ratings between users and items. As demonstrated in Figure 4.3, the proposed model in this work mainly consists of a four-layer neural network structure, which are input layer, embedding layer, pooling layer and output layer.

Input Layer – The input in this paper was the user-item interactions and user-user interactions. The input layer took the input data as input and passed them to the next layer.

Embedding Layer – The essential idea of embedding layer is to reduce the dimension of data and use dense and low-dimensional vectors instead of sparse and high-dimensional ones to represent data. As mentioned in section 3.4, we use embedded matrix instead of huge one-hot coding vectors to represent each input user and item. For example, what the embedding layer does here is to use vector [0.22,0.02,0.48,...,0.21,0.56,0.15] to represent user with id "1", and the same for item. These embeddings could be view as the inherent factors in some latent geometric space. The latent geometric space was hard to understand subjectively, but any vector can be represented in this space. The relationship between them can be computed by vector computation. For example, the users' embedding can be viewed as the users' preference. Each dimension denotes some aspect that the user may be interested in. Through training process, the embedded vectors will also be updated, so that the interaction between users and items could explored in the low dimensional space.

Neural CF Layers – User embedding and item embedding obtained from the embedding layer would be entered into a multi-layer neural structure. The purpose is to project the latent vectors to some latent space for further operation through linear and non-linear functions. Each layer in this structure can be customized for the required function to capture the interaction between users and items. The capability of this structure is determined by the dimension and number of hidden layers.

Output Layer – The final layer of the who architecture is the output layer. It was employed to output the final predict results.

4.6 Model Optimization

Matrix decomposition is based on the prediction of the user's rating of the candidate, then the ranking based on the predicted rating, and finally the

recommendation to the user. This method is a typical Pointwise method. Whether predicting the score or predicting the implicit feedback, it essentially predicts the user's preference for an item. However, this method has a big problem, because most of the time we can only collect a few positive samples, and the rest of the data is actually a combination of real negative samples and lacking values. The missing value here refers to the unknown data except the positive example and negative exception in the training data, which can be understood as unexposed or exposed but the user may not notice the missing data, so the sample in the missing value may be the positive example or the negative example. When we use this method to construct training data, we are often unable to determine what the negative examples are, so we can only treat all the other parts except the positive examples as negative examples, which will make part of the negative examples in training data actually be missing values. Take the missing value as the negative sample, and then use the prediction error as the criterion to approximate these samples. It is ok to approximate the positive sample, but at the same time the approximate negative sample is just a missing value, which is really presented to the user, and it is not sure whether they like it or not. Moreover, such a model can only predict positive or negative examples, which is not conducive to sorting because it cannot distinguish the importance of samples within the category in depth. Of course, in this case, we can also use some other methods to avoid these problems, such as negative sample sampling, such as sorting by forecast probability, but these methods are only "delaying tactics", not perfect for solving the sorting problem. Pairwise method to re-optimize matrix decomposition.

Since this paper focuses on the implicit feedback, we define the recommended tasks from the perspective of ranking. Therefore, we chose to optimize the model parameters using a pairwise learning method that commonly used in ranking tasks, which also known as Bayesian personalized ranking algorithm. The basic assumption in pairwise learning method is that the observed interactions should rank higher than the unobserved ones. In particular, we adopted the regression-based pair-wise loss function that commonly used in the ranking-based recommendation system. The regression-based pair-wise loss function can be denoted as:

$$\boldsymbol{L}_{user} = -log(\sum_{i,j,k \epsilon \boldsymbol{R}} (r_{ij} - r_{ik})) \tag{4.7}$$

where $\boldsymbol{R}$ denotes the train data, and $i, j, k \epsilon \boldsymbol{R}$ was a triplet meaning that user i had interacted with item j and there existed no interaction between user i and item k. Here the interaction means a behavior from user i, for example, click or buy, or other behavior proving that the user took an interest in that item.

At present, the most common algorithm for optimizing the loss function is SGD. In each iteration, SGD calculates a mini-batch train data's gradient and then updates the parameters. However, in SGD, all parameters are updated based on the same learning rate, which is not always possible to reach a global optimal solution. In practice, it may be the case that some parameters are already nearly optimal and therefore only need to be fine-tuned, while others may need to be significantly tweaked. If the learning rate was too small, the variable with a large gradient will converge slowly; if the learning rate was too large, the variable that has been tending to converge may be unstable. AdaGrad is a solution to this problem. The algorithm idea is to use a different learning rate for each variable. At the beginning of model training, a larger learning rate is used to make the model gradient drop rapidly. When the training reaches a certain level, AdaGard will assign a relatively small learning speed to variables that have changed a lot, while for variables that have changed a little, the learning speed will increase relatively. RMSprop is an improvement of AdaGrad, which can alleviate the problem that the learning rate of AdaGrad declines rapidly. Therefore, AdaGrad is well suited to working with sparse data and RMSprop are suitable for handling non-stationary targets. Adam combined these two strengths. In each iteration, it calculates different adaptive learning rates for different parameters, and keeps the learning rate fixed in a value range, making the update of parameters more stable. In general, RMSprop or Adam was chosed to optimize the loss function. The optimization algorithm is shown in Alg. 3.

In general, the neural network has a good fitting ability, but when the neural network structure is too deep or the parameters are too many, the model is hard to avoid the problem of overfitting. We will mainly use regularization, dropout to prevent the model from falling into overfitting state.

Algorithm 3 Training Algorithm for the supervised deep learning model

Input: user social network $G = (V, E, \pi)$, and the user-item rating matrix R
Output: the predict rating r_{ui}

Initialize the parameters θ
for $iter$ = 1 to $iter_num$ **do**
  for train data X in mini-batch **do**
    $L(X, \theta)$ = Model(X, θ);
    $\partial L_{min} / \partial \theta = Grandient_Descent(L(X, \theta))$
    update paramaters θ
  end for
end for
return r_{ui};

Chapter 5

An Adaptive Attentive Model for Social Recommendation

5.1 Motivations and Rationales

According to sociological theory, users usually get informations from their social relations, such as friends in real world or friends in social media. Social recommendation systems were on the basis of this social effect, the phenomenon that a user's partiality for an item was influenced by his/her social neighbors. Effective use of users' social relationships had been instrumental in improving the quality of the recommendation system. This led us to explore the way of accurately catching the affect of social neighbors on users' partiality. Previous recommendation systems tried to model social influence through trust propagation, regularization constraints, deep learning and other methods. Although these methods demonstrate the effect of social relations on the recommendation system, they still have some common drawbacks.

First, in the traditional recommendation system model such as SocialMF and other deep learning based model, user embedding adopts a simple initialization method to transform user's identity ID which was typically denoted as the one-hot encoding into a random vector conforming to gaussian distribution, and these vectors are input into the interaction layer for calculation to obtain user embedding. Embedding the user'S ID directly into a low-dimensional space does not fully exploit the implicit intelligence of the social network, so it is difficult to get the most effective embedding.

Second, in practical application, people's choice of an item is usually decided by many factors. The research on the influence of the existing algorithms on social effect on people's behavior is insufficient, which is mainly

reflected in the homogeneity and heterogeneity of social interaction. (1) Social homogeneity leads to similar preferences of users. However, this similarity may only exist in certain respects. As an illustration in Figure 5.1 (A), users Alice and Bob and Mike are friends, but their preferences are not always the same. Among Bob's many preferences, Alice is only interested in movies. Similarly, in Mike's preference, Alice is only interested in sports. Therefore, when we want to make use of the users' social neighbors' preference to predict users' preference, it is necessary for the model to consider how to extract users' specific preferences in a certain aspect to enhance the quality of recommendation. (2) Although the emphasis on a certain aspect of users' friends preference can enhance users' preference to a certain extent, the influence of social effect on users also has heterogeneous characteristics. The heterogeneity of network is mainly reflected in two aspects. First of all, the social relationship between users is different, generally there is a difference between strong relationship and weak relationship. For example, connections between users and their social friends with whom they often interact are called strong links. Social friends with strong connections tend to have a greater impact on users than those with weak links. Last but not least, the affect of social friends on users' partiality tends to vary from situation to situation. We refer to the user's choice of different items as a context. Even the same social friend should have different influence on users in different contexts. For example, as shown in Figure 5.1 (b), Angel and Mike may have the same effect on Alice when the system recommends a movie to Alice, because they are both interested in movies. However, when we want to recommend travel products to Alice, it is clear that Mike's influence is higher than Angel's. Because Mike has more experience in Travel. Therefore, we found that the user influence of a social user varies with the context, and does not always remain the same.

Motivated by the previous analysis, we proposed the subsequent methods to figure out the above problems.

First, different from the previous mentioned embedding method which mapping the identity of users to a low-dimensional latent space through one-hot-encoding, we proposed to utilize network embedding technology to transform the user's identity to a dense representation which preserved rich information in social domain. Secondly, we proposed a social recommendation model based on attention mechanism. Inspired by the ensemble

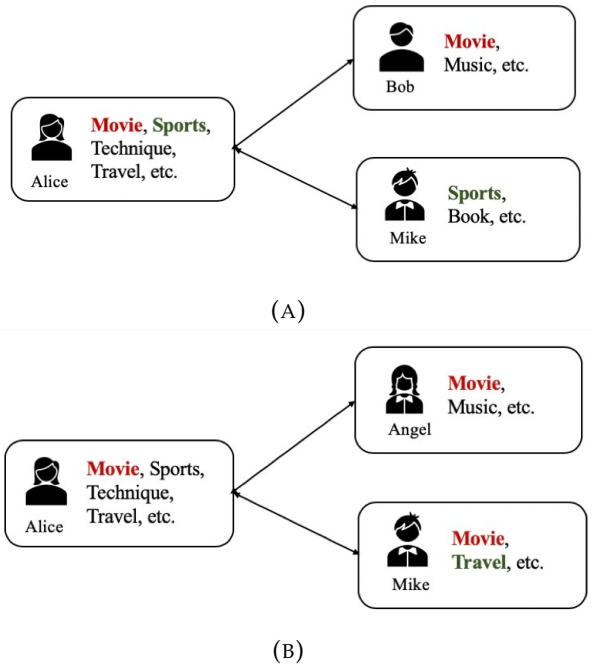


FIGURE 5.1: An illustration of social influences, (a) demonstrates that users may not have similar preferences in all aspects and (b) demonstrates different social neighbors contributes different to users' decision

methods, which argued that there were two factors influence a user's partiality. One is on the basis of the user's past taste. The other is on the basis of the influence from their social neighbors. For this reason, we distinguish two inherent factors that affect a user's decision on the given item. The first factor was called the item-based inherent factor and the second factor was called social-based inherent factor. Item-based inherent factor could be viewed as users' specific preferences, which would not be influenced by other factors. As discussed above, it could be inferred from the user's historical preferences for items. to infer the user's possible preferences based on the user's social influence. Social-based inherent factor were inferring users' possible preference based on the partiality from theirs social neighbors. It could be learned through the complex effects of the social relationship between users and their neighbors. Especially, we designed a module based on the attention mechanism. It could not only acquire the user's factor-level attention on the user's social neighbor's partiality, but also learn the different influence value of the social neighbors. Finally, these two factors were combined to form the users' representation. In next section, we are going to explain our method for

further details.

5.2 Adaptive Neural Network for Social Recommendation(ANSR)

We first introduced the framework of ANSR and then introduced the details of the key components of the proposed model, namely :1) Network embedding-based representation learning; and 2) Attention-based social effects modeling. Finally, we will explain the training algorithm in ANSR.

5.2.1 Model Framework

We will utilize the representation learning framework to solve the social recommendation problem. In a representation learning framework, each entity is represented as an embedding vector that encodes the latent feature of the entity. For example, the user's embedding vector represents the user's preference, and the item's embedding vector represents the item's property. The most widely used matrix factorization method in the recommendation system is a typical representation learning framework, and calculates the correlation between the user and item through the inner product. Therefore, based on the representation learning framework, we build our model to learn the representation of users and items and make recommendation to user. The overall architecture is show in Figure 5.2.

Input Layer

The input of this model is the rating feedback of users and the social network of users. The rating feedback of users is usually represented by a rating matrix which is denoted as $R \epsilon \mathbb{R}^{n \times m}$. If user i has rated item j, then the rating score of user i on item j is represented by $r_{ij} = 1$. The users' social network is formally denoted as $G \epsilon \mathbb{G}^{n \times n}$. Let's take one user *useri* as an example. As shown in Figure 5.2, the input will be *useri*, an item rated by user i and *useri*'s social neighbor list. Each user and item has a unique identity to represent the input data.

Embedding Layer

When the identity of the user and item is input into the model, it needs to be expressed in a way that can be understood by the computer before various calculations can be performed. Therefore, the purpose of the embedding

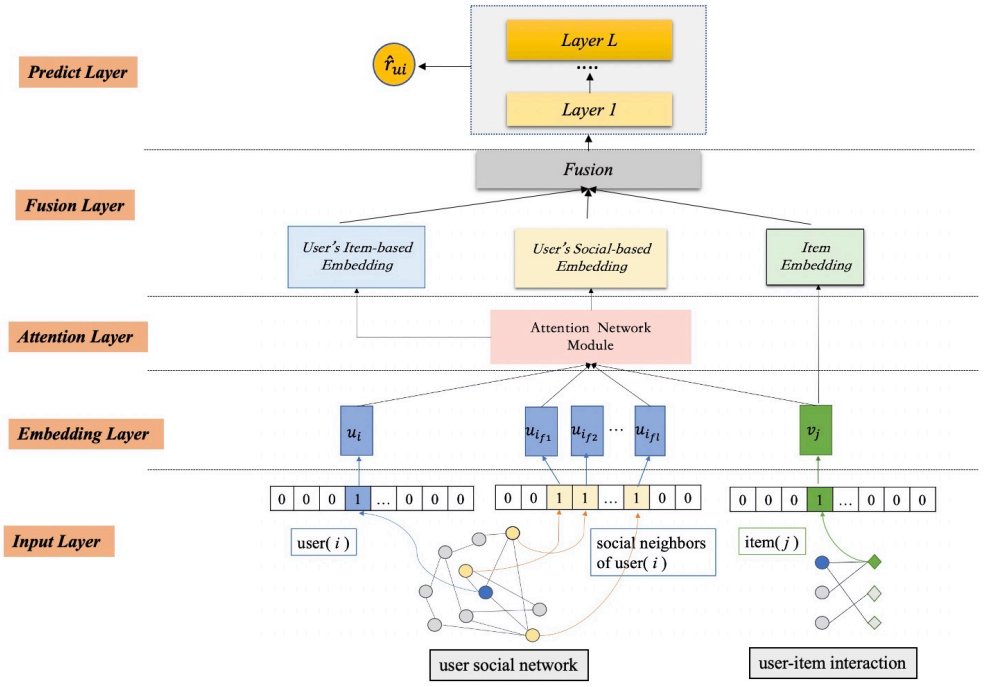


FIGURE 5.2: The overall Architecture Framework of ANSR.

layer is to encode the input object entities into a low-dimensional dense vector, which is also called a presentation in representation learning framework.

First of all, it was necessary to consider how to obtain the item's representation. The simplest way is to represent an item by one-hot-encoding. For example, suppose we have a total number of 10 items, and when these items are stored in the database, each item has its own unique identity. The $j-th$ position represents the item j in the item list, so item j can be represented by [0,0... 1... 0]. This vector is a 10-dimensional vector with only one position of 1 and the rest positions of 0, as shown in Figure 5.3. But in practice, the number of items is probably in the tens of thousands. Such an embedding will be a high dimensional sparse vector. Essentially, we define a embedding matrix in the embedding layer, in which each row of the matrix represents the embedding vector of the corresponding item, which is shown in Figure 5.4. The representation obtained from the embedding layer could be viewed as the items' latent feature which represents the items' attributes. As shown in 5.5, we took the Titanic Movie as an example, each factor in this vector represents an attribute of the movie. This embedding needs to be continuously optimized in the training of the model, and eventually a reasonable way to

present this item is realized.

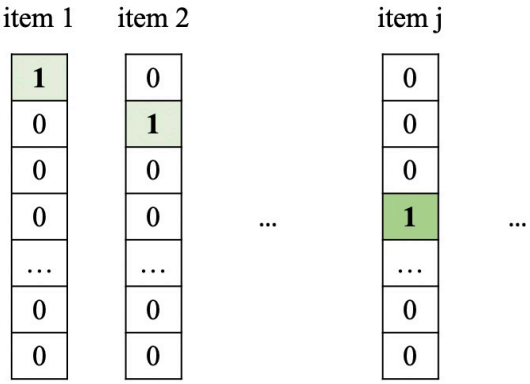


FIGURE 5.3: An example of utilizing one-hot-encoding to represent an entity.

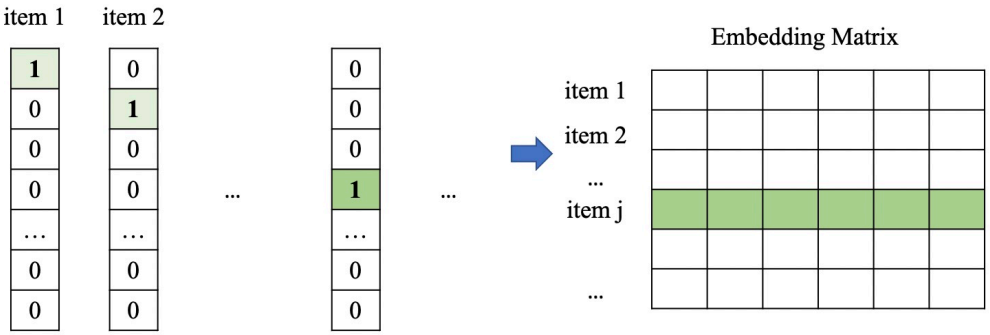


FIGURE 5.4: An example of mapping the identity of item to a low-dimensional space to obtain a dense representation of an entity.

Embedding of users can also use one-hot-encoding. However, the obtained vector does not contain the user's social network information. Therefore, different from the approach mentioned above, we have utilized network embedding to map users to a latent space, where the position of connected users is as close as possible. In this semantic space, this representation is also valid, because connected users usually have similar interests, so the vector representation between them should also be similar. When translated into the latent space, it is denoted as position proximity. The specific methods will be explained in detail in Section 5.2.2.

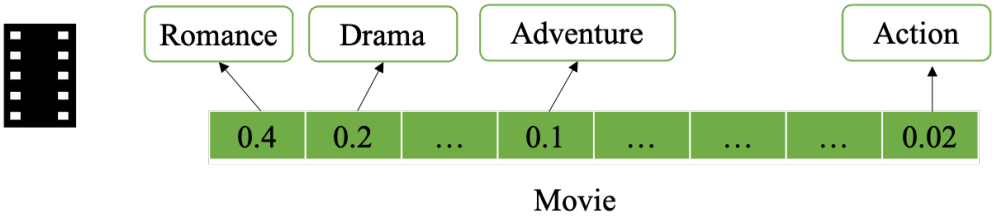


FIGURE 5.5: An example of the explanation of an item vector.

Attention Layer

In comparison with the item embedding, the way of learning the representation of user would be more complex. In a social-based recommendation system, users' preferences are often influenced in two ways. On the one hand, users' preferences can be inferred from their own rating history. On the other hand, users' partiality could be influences by the interests of their social neighbors. As a result, the user's representation is divided into two factors. The former means to infer a user's preferences from the context of the user's own preference history. The latter means to infer a user's preferences based on the preferences of a friend. We will explain how to obtain these two representations of the user in details in the next section.

Fusion Layer

From the different modules, we would get several different user representation that represents the different inherent factors of the user. We need to fuse these representations into a synthetic representation as the input of the final prediction layer, and the fusion layer was such neural network to achieve this goal. In general, the strategy of the fusion layer could exploit a number of different methods to merge different factors. These methods could be as simple as concatenating all these features, adding these features or utilizing element-wise product. By experiment and comparison, we found that simple addition works best. Accordingly, we first added these two factors to get the final user feature. According to the factorization algorithm, the interaction could be learned effectively through the element product approach. As a result, we applied the element product approach to learn how users interact with items. However, some work has found that the element product approach may lose some valid characteristics, so in addition to that, we need to concatenate the element result with user representation and item representation to get the final input of the final prediction layer.

Prediction Layer

In general, the input of the final layer was obtained in the fusion layer, and we would utilize this input to predict the result. But since we focus on implicit feedback, our goal was to predict the probability that a user u will be interested in an item i or not. It was worth noting that in the final layer, we utilized the sigmoid function to convert the output value into a probability between 0-1 to predict the probability of whether the user u will be interested in the item i. This could be achieved through several fully connected layers, which could be defined as:

$$h_g = Sigmoid(W_g * h_{g-1} + b_g)$$
$$h_{g-1} = ReLU\left(W_{g-1} * h_{g-2} + b_{g-1}\right)$$
$$...$$
$$h_g = ReLU(W_1 * h_0 + b_1)$$

As we know, deep learning models could solve complex problems by either increasing depth or increasing width, and the cost of increasing width was often much higher than the cost of depth. This was because deeper models mean better non-linear representation, more complex transformations can be learned, and more complex feature inputs can be fitted. However, as the number of hidden layers increases, the complexity of the model will increase correspondingly. Therefore, the final prediction layer actually contains only two hidden layers.

5.2.2 Network embedding-based representation learning

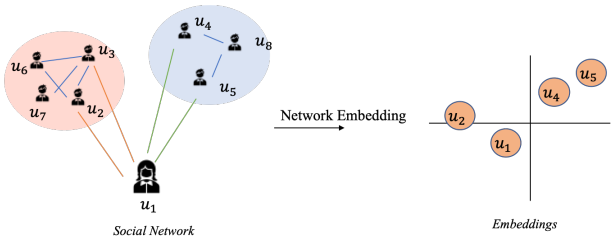


FIGURE 5.6: The illustration of Network Embedding. The connected users will have a similar embedding vector in the latent space.

In the existing research, the user's ID was directly mapped to a random latent space to represent the user's inherent factors. This approach is obviously not effective in exploiting users' social signals. Most social-based recommendation algorithms assumed that the user's representation in the social latent space should be as similar as possible to the representation in the preference latent space. Therefore, users' social network relationships may store a lot of hidden information of users, which is difficult to be read directly. However, the representation of their inherent factors may help us obtain a lot of meaningful user information. The biggest advantage of network embedding technology was that it could learn the low-dimensional vectors of nodes while preserving the network topology and node feature. Figure 5.6 gave a simple example of the process of the network embedding. Based on this advantage, we adopted the network embedding method to make full use of users' social signals. Deep Walk is the first algorithm that apply the ideas of NLP into Network Embedding. Figure 5.7 illustrate the main process of Deep Walk.

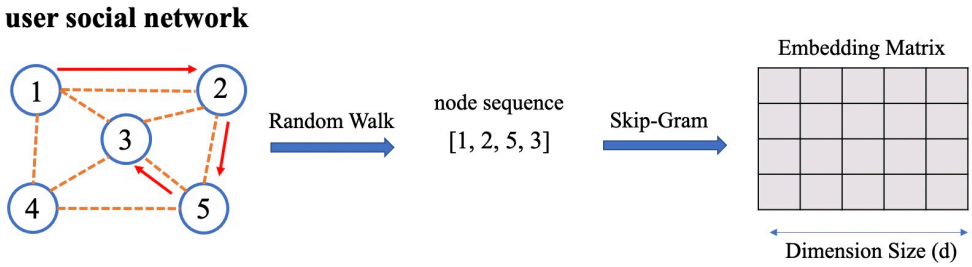


FIGURE 5.7: The illustration of the Deep Walk Methods. First, generate a social corpus with random walk. Then, learn the node representation using the Skip-Gram Model.

First, we utilized users' social network G as the input. As shown in Figure 5.7, we have five users. We will generate a certain number of node sequences for each node by random walk. The random walk method is to start at a particular node, randomly select an edge that is connected to the current node, and walk along the edge to the next node, and continue this process until all the points generate a random walk sequence. For example, starting with $user1$, we randomly select contiguous nodes to walk around and eventually generate a sequence of nodes like $[1, 2, 5, 3]$. The study found that a random walk on a network is equivalent to a sentence in a natural language domain.

And all the paths that travel through a certain strategy can be combined together as a corpus. Next, we can use the Skip-Gram model in natural language processing for network representation learning. We use the Word2Vec model directly. With the network embedding operation described above, we can output a new representation for each user. This representation contains the user's social semantic information.

There are many existing network embedding methods, such as DeepWalk, Node2Vec, SDNE and so on. We will also use different network embedding methods to learn the user representation in the following sequence to compare the impact of different network embedding methods on the model accuracy. The learning of user representation through network embedding can be regarded as a pre-training strategy, and the acquired features will be input to the next level to mine more implicit information in user representation, so as to get a better user representation. At the same time, we also found that it helps to improve the convergence speed.

5.2.3 Attention-based social effects modeling

To strengthen the prediction accuracy, applying attention mechanism into neural network model has become an important approach. For example, in the areas of research such as image recognition and machine translation, attention mechanism have been successfully employed into deep learning models to strengthen the prediction results. We could consider the attention neural network as a powerful feature extractor in nature. Simply understood, it extracts the data that most relevant to the predicted results, and the data that is not important to the predicted results will be ignored. For that reason, the attention model could be utilized to withdraw more valuable features for each user, so as to reduce the noise generated by invalid features on the model to help us learn user preference features more accurately. The purpose of attention-based social effects modeling is to make better use of the preferences of users' social friends to enrich the preference information of users, so as to advance the performance of the recommendation systems.

In the previous section, we have discussed that the user's representation can be learned from two factors. The former factors means to infer a user's preferences from the context of the user's own preference history. The latter means to infer a user's preferences based on the preferences of a friend. Users tend to have similar interest with their social neighbors and friends, so

the preference features of social friends can be used to predict users' interest. However, we found that not all preferences are consistent, and the features that other users are not interested in will become the noise of the system, thus affecting the system's recommendation results to users. Therefore, we can use the Attention model to extract the part of the features that users are interested in. Through embedding layer, we have obtained the representation of each user, and we will take these representations as the input of the attention layer to learn the preferences representation of users in specific aspects. Figure 5.8 illustrate how we learn the specific attention vector between users and users's friends.

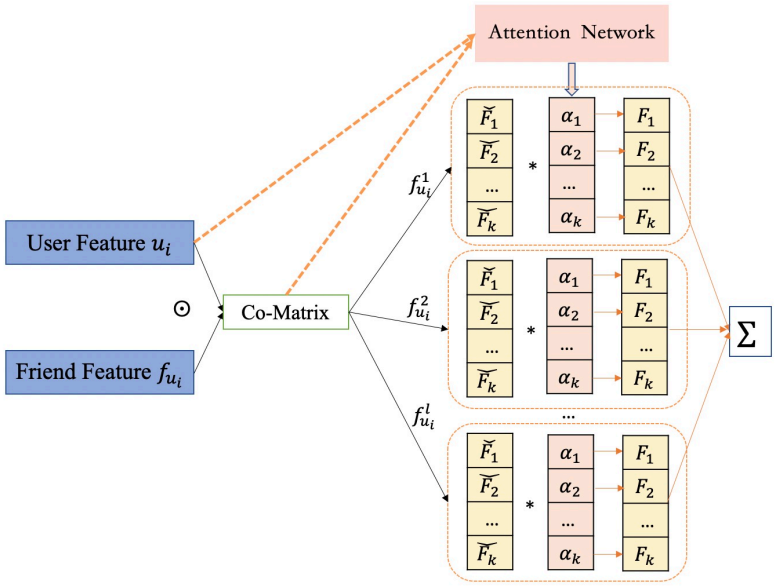


FIGURE 5.8: The illustration of the framework of aspect-aware module.

First, we need to calculate the correlation, that is, the degree of similarity, between user i's inherent factors and his social neighbor u_i's. To calculate the correlation between these two features, we could exploit different functions and calculation mechanisms. In general, the commonly employed way to achieve this function were utilizing dot product, Cosine similarity, or applying an additional neural network. Accordingly, we first use the element-wise product to calculate the similarity between user i's inherent factors and his social neighbor u_i's, which could be obtained through the following function:

$$C_i = \widetilde{u_i} \bigodot f_{u_i}^l \tag{5.1}$$

where $\widetilde{u_i}$ denoted user i's representation, $f_{u_i}^l$ denoted user i's $l-th$ friend's representation, respectively. Take the user's vector and each user's friend's vector, we can obtain a new vector by element-wise product. These new vectors can form a new matrix which is the correlation matrix between the user and the friend. In our work, the user's feature can be regarded as composed of different factors. Suppose the factors represents different factors of the friend that the user may be interest in. Then, the attention network can be viewed as a feature extractor, by assigning different weights to different factors, it will extract the factor that most relevant to the user's interest, and the factor that is not important to the predicted results will be ignored. Then, we will use $softmax()$ function to convert the correlation matrix into the probability distribution with the sum of all element weights being 1. As a result, the factor that more similar with the user's interest will assign a higher weight through the attention score learned from the $softmax()$ function. We use P_k to represent the $k-th$ factor in a user's feature. The process of calculating attention score was defined as followings:

$$o^{p_k} = ReLU(W_1 * P_{f_l}^k + W_2 * C_i^T + b) \tag{5.2}$$

where, W_1, W_2, and b are the weight matrices of the attention network. $P_{f_l}^k$ represents the $k-th$ factor of f_l. Then, we normalize this weight using the softmax function to get the available weights:

$$a^{p_k} = \frac{exp(o^{p_k})}{\sum_{k \epsilon F(f_l)} exp(o^{p_k})'} \tag{5.3}$$

The score will help us extract more related information to user's specific preference. Finally, the specific representation of the user on a friend can be obtained by the following formula:

$$user_i^{item\_based} = \Sigma_{f_l} a^{p_k} * P_{f_l}^k + u_i \tag{5.4}$$

On the other hand, the influence of users' social relations on users' preferences is reflected in the heterogeneity of social networks. Due to the different relationships between different social friends, we need to assign different weights to the users' friends. Moreover, the weight is not static, but dynamic.

This weight should be different when the user interacts with different items. Intuitively, if a friend has more experience on an item, he should have more influence on the user when the user facing choice of that item. To understand this, consider an example where a user decides to watch an action movie. If a friend of the user has seen many different action movies, that friend should have more influence when the user considering whether or not to watch an action movie. We need to learn a parameter that represents the user's influence on different item choices. Different with the aspect-aware module which focuses on assigning different weight to the factors inside a user's feature, this module will assign different weight to different friend's feature. The process of calculating the influence value will be similar with the aspect-aware module. The framework of the module was displayed in Figure 5.9. First, we need to calculate the correlation between these three features, the calculation process was shown as following equation:

$$H_{f_l} = w^T ReLU(W_v * v_j + W_f * f^l_{u_i} + W_u * u_i + b) \tag{5.5}$$

$$\alpha(f) = softmax(H_f) = \frac{exp(H_{f_l})}{\sum_{f_l \epsilon \mathbb{F}_{u_i}} exp H'_{f_l}} \tag{5.6}$$

where, W_v,W_f,W_u,b is the weight matrices and bias of the attention layer. f_l is the l-th friend of user i. $\mathbb{F}_{u_i}$ is the friends set of user i. $\alpha(f)$ is the influence score of $l-th$ friend on user i. Based on this attention score, we could select more representative social friends to represent the user's social information. Then, user's latent factor in social domain was denoted as:

$$user_i^{social\_based} = \Sigma\alpha(f) * f^l_{u_i} \tag{5.7}$$

5.2.4 Optimization and Training

In the recommendation system, it is not common for the user to give an explicit rating to the item, what more common is that the implicit feedback which without obvious rating behaviors, such as clicking an item, buying an item, watching a movie, etc. Compared with explicit feedback, implicit feedback is more practical because most websites today are not rating based sites, but interaction-based sites, so implicit feedback can be used in a wider range

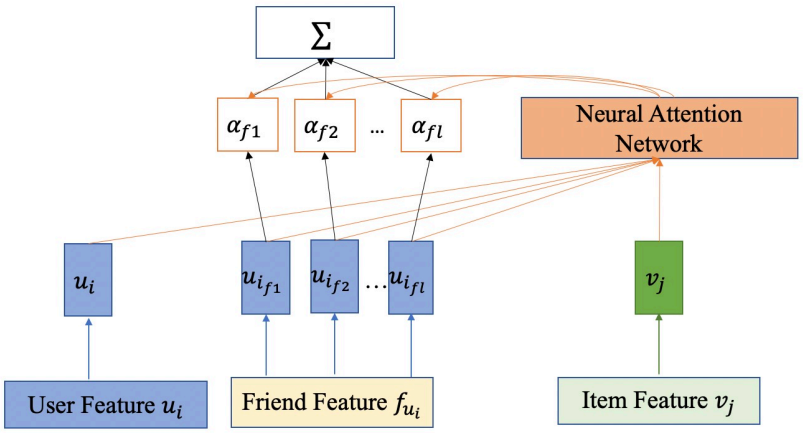


FIGURE 5.9: The illustration of the influence calculating module.

of scenarios. Consequently, we will pay more attention to the implicit feedbacks. Here we're going to focus on the pair-wise loss function for implicit feedback. BPR is a pair-wise ranking algorithm which is commonly used in ranking task. BPR is actually an optimization of the loss function in the factorization model. It is based on a partial ordered model that sorts the user's interactions with different items. In general, the loss function was displayed as the following formula:

$$L_{BPR} = \sum_{(u,i,j)\epsilon D_s} -ln\sigma(\check{y}_{ui}(\Theta) - \check{y}_{uj}(\Theta)) + \lambda_\Theta ||\Theta||^2$$

The meaning of this formula was that the user's positive interaction should be ranked higher than the negative one. Where D_s represents the training data set, which is composed of different pairs of user-item. As we mentioned in our explanation above, the loss function requires a sample of positive and negative items corresponding to the user. Item i represents the positive sample of user u, which can be obtained directly from the user's historical interaction data. Item j represents the negative sample of user u, which the user has not interacted with. Therefore, during training, we need to randomly sample a negative item. Additionally, in order to prevent overfitting of the model, we need to regularize the parameters of the model. λ_Θ represents the regularization parameter, which can be specified in advance. With the objective loss function, we need to utilize the loss optimization algorithm to minimize the

above objective loss function. Deep network parameter learning mainly uses gradient descent method to minimize loss function. The method used for optimization can be generally divided into three forms: batch gradient descent(BGD), stochastic gradient descent(SGD) and mini-batch stochastic gradient descent. BGD will obtain a global optimal solution, but each iteration requires all data in the training set. By doing so, the calculation resources will be occupied and calculation efficiency will be slow. In SGD, only one sample is used for each parameter update, which is randomly selected. However, a problem of SGD is that it is sensitive to noise data. In addition, thousands of iterations may be required before the optimal solution is obtained, which also affects the efficiency of calculation. Therefore, here mini-batch gradient descent method was utilized for model training.

Algorithm 4 Optimization Algorithm

Input: R: Observed Rating Matrix; G_u:Users' Social Network;
Output: unobserved rating r^*

```
Initialize parameters : D, λ, dp, lr, num_neg;
for i = 1 to epoch do
    Sample mini-batch size user-item pairs;
    For each positive pair (u, p_i) and negative pair (u, n_i);
    pos_prediction = model(Θ_u,Θ_i);
    neg_prediction = model(Θ_u,Θ_i);
    Loss(pos_prediction,neg_prediction)
    Update Θ_u and Θ_i via backward()
end for
```

Algorithm 4 demonstrated the training algorithm. In the beginning, we need to initialize the parameters of users, items and the weight/bias in each neuron. In addition to the user embedding that can be learned by embedding the graph, all other parameters can be initialized by the Gaussian distribution. After the model parameters are initialized, we can input the training data into the model for learning. In the above discussion, we have mentioned the influence of different batch selection on the model. The neural network training process is carried out simultaneously on all m samples (called batch) through vectorization calculation. If m is large, the training will be slow, because each iteration will require summation and matrix operations on all sample bases. Therefore, here we use the commonly used Mini-Batch gradient descent algorithm to optimize the model. M training batches are divided into several subsets, called mini-batches, and then neural network training

is carried out on a single subset at a time. In 5, the algorithm of small batch sequence generation is presented. Using this training data, we will calculate the loss function described above and compute the gradient to update the parameters.

Algorithm 5 Mini-batch user-item pairs Generation Algorithm

Input: R: Observed Rating Matrix; G_u:Users' Social Network;
Output: a mini-batch size train data:u_i, p_i, g_i, F_{u_i}

```
for user in mini − batch size train user set do
    u_i ← append(user);
    p_i ← append(positive item in user i's rating data);
    for t in negative sample size do
        n_i ← append(sample a negative item of user i);
    end for
    F_{u_i} ← append(user i's friend list);
    Reshape;
end for
```

5.3 Experiments Settings

5.3.1 Datasets

The data set used in the experiment is commonly used in the field of social recommendation. They were collected and processed by other scholars and published publicly. Statistics of different data set are recorded in Table 5.1. The detailed description of the data set is as follows:

Delicious: Delicious(http://www.delicious.com) was an online social bookmarking platform that enable custormers to store and find bookmarks on the site and assign various semantic tags to them. At the same time, users can also create social relations and share bookmarks with their social friends. This data set is collated and published by HetRrc [65]. Since we only consider social based personalized bookmark recommendation, we only select the bookmark records and user social data for experiment.

Ciao: Ciao is a data set widely used in social recommendation systems. It was crawled from a DVD website by Guo et al. [8] Users can rate movies on the website and follow the users they trusted.

Epinions: Epinions is also an online review site. Users can rate many different types of items. In particular, Epinions provide a trust system that

TABLE 5.1: Data set statistics

Data set	Delicious	Epinions	Ciao	Douban
users	1,521	18,163	17589	2,848
items	1,202	37,325	16,121	39,586
ratings	8,397	374,658	62,452	894,887
social	10,401	287,260	40,133	35,770

allows users to add trusted users to their own trust networks. Therefore, the dataset includes not only the user's rating data, but also the user's trust relationship network. This dataset could be found from LIBREC's website.

Douban: Douban is a Chinese social review site. The site enables custormers to rate movies, books and music they have watched. The site also supports a variety of online social services, such as joining different interest groups and following other users with the similar interests. The data set consists of three categories, namely, the user's movie rating data, and music and book rating data. Due to the large amount of data and different categories in the data set, we mainly exploit the movie rating data to predict the movies that users might be interested in. The data set is publicly available on a data site and can be downloaded via the link written in the paper.

5.3.2 Baselines

For the purpose of examining the effectiveness of ANSR, we will carry out experiments on the above mentioned data set to compare the performance of ANSR with baseline methods. To better understand the baseline methods, the baseline methods would be classified into the following categories: traditional recommendation method:BPR, social-aware recommendation method: SBPR, deep learning-based method:NCF,DMF,APR, and deep learning-based social recommendation method:CUNE-BPR, which are shown as follows:

BPR: BPR(Bayesian Personalized Ranking)[58] algorithm was a commonly used pairwise ranking model in the recommendation system, which was an optimization of MF. Instead of viewing the recommendation task as a regression problem, BPR is more concerned with which items have higher priority in the user's preference. Since we focused on implicit feedback, it is very competitive for ranking model.

SBPR: ScoialBPR [20] is an optimization model of BPR. On the basis of

the original BPR, SocialBPR incorporated the user's social data into the original BPR model to improve the recommendation performance. The concept of SocialBPR is that users tilting towards to rank the items that their social neighbors prefer higher compared to the negative items that they did not rated. It is competitive in social recommendation domain.

NCF: NCF [39] is a collaborative filtering method based on deep learning, which is also the basic model of various other deep learning algorithms. It proposed to use multilayer perceptrons to model the linear and nonlinear interactions between users and items. Our recommendation algorithm will also based on this model, but new input features and algorithms are added to optimize the NCF algorithm.

DMF: DMF [66] is an improvement of traditional MF algorithm using neural network. The difference between DMF and NCF is that NCF randomly initializes the user and item embedding representation from a gaussian distribution, but DMF takes the user's feedback as input and maps it to the a latent space to learn the user's representation by imitating the idea of semantic model. It is a competitive model in deep learning-based recommendation domain.

CUNE-BPR: CUNE-BPR [67] is an optimization of the BPR model based on network embedding. This model solves the cold start problem effectively by utilizing the feedback to extract reliable friends with similar interests of users through network embedding.

APR: APR [68] is a new framework based on deep learning, which makes use of GAN's idea to enhance the recommendation system through adversarial training. At the same time, the robustness and generalization ability of the recommendation model can be improved.

5.3.3 Evaluation protocols

In the experiment, we would adopt a commonly utilized method called 'leave-one-out protocol' as the test method. In particular, we randomly selected an item from his implicit feedback and put it into the test data set for each user. This method is often used in recommended ranking tasks. The remaining data were randomly divided into 80% training data and 20% validation data. The purpose of validation data set is to optimize the model parameters. Due to the huge amount of data in training sets such as Douban, it would take a lot of time if all items were tested and sorted. In order to shorten the test

time, we randomly selected 300 non-interactive items for each user as negative data set, and put the test item which was picked up according to the 'leave-one-out protocol' into these negative sampling data for ranking.

HR@K and NDCG@K were commonly utilized metrics to examine the validness of a ranking model. HR@K represents the success rate of hit items. At the same time, we hope that the position of test item appearing in the list of predict items as high as possible. This is important for ranking tasks, because users would become dissatisfied with the recommendation system if the items they are interested in are ranked very low. Therefore, NDCG (NDCG@K) could tell us whether the recommend list provide a good ranking. The definition of these two metrics were shown as follows:

$$HR@K = (Numberof Hits@K)/(|GT|) \tag{5.8}$$

$$DCG@K = \Sigma_{i=1}^{K}(\frac{2^{rel_i}-1}{log_2(i+1)}); NDCG@K = \frac{DCG@K}{IDCG@K} \tag{5.9}$$

In this experiment, we used the leave-one-out strategy to evaluate the model. The number of items is determined by 300 negative items of the random sample and a positive item randomly selected. The numerator represents the number of positive items in the test set that the predicted result can accurately hit. Generally, we will call it a 'hit'. K represents the number of items in the predicted list for each user. The formula of NDCG looks complicated, but its idea is very simple. In the test, the model returns a recommendation item list, and we want to calculate how good the list is. Each item in the list has an associated score, usually a non-negative number, known as 'gain'. $rel_i = 1$ if the $i-th$ item hits the test set exactly. For other items with no interaction with users, we usually set the gain to 0. And then we add all the scores, which is called 'cumulative gain'. We want the item in the test to be at the top of the recommended list, so before adding up the scores, we divide each item by an increasing number, usually the logarithm of that position, which is called the depreciation value, denoted as $log_2(i+1)$. NDCG is the normalized DCG. Finally, NDCG@K is the normalized DCG@K.

5.3.4 Parameter settings

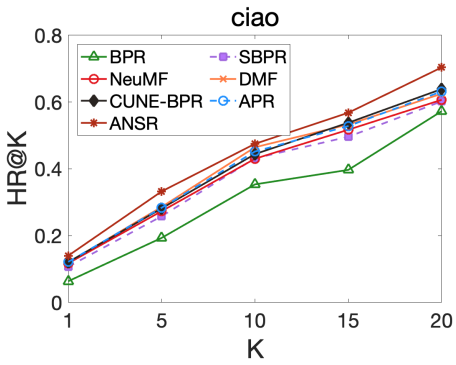
We implement our proposed model ANSR on Pytorch, which is implemented by the Python language. Parts of the parameters were learned through pretrain strategy. And part of the parameters used in the initial model are randomly initialized. In the experiment, RMSProp, which is an adaptive learning rate optimization algorithm, was used to optimize the model.Batch size is an important parameter in deep learning. Within a reasonable range, the increase of batch size can make the parallelization efficiency of matrix multiplication higher and improve the utilization of memory. At the same time, the number of iterations needed to complete an epoch can also be reduced, thus improving the running speed of the model. The large batch size can also reduce the gradient oscillation to some extent. However, batch size is not the bigger the better. We should consider the memory capacity. Therefore, we need to carefully tune the batch size. Usually, it could be tuned among [64,128,256,512]. In addition to using regularization, we will also utilize dropout method to prevent model overfitting. Specifically, during forward propagation, we let the neurons in the network stop working with a certain probability. This has the advantage of making the model more generalizable. The reason is that models don't rely too much on local features. According to practice, the dropout ratio will be tuned between 0.4 and 0.6. Another useful method to avoid overfitting is called regularization. By regularization, additional constraints are imposed on the model, such as constraining the range of parameter values, to prevent the model from becoming too complex. Therefore, we utilize L2 regularization and the regularization parameter will be tuned among [0.01, 0.001, 0.0001].

5.4 Performance Evaluation

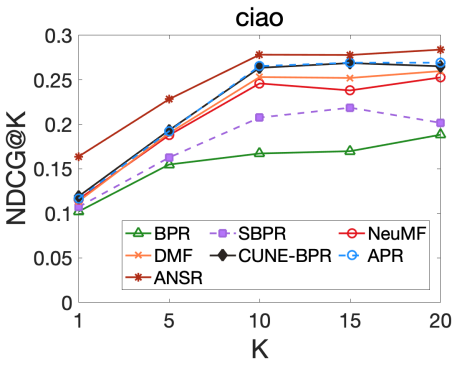
5.4.1 Overall Performance Comparison

In the first experiment, we will evaluate our proposed model(ANSR) on the above mentioned data sets. For the purpose of confirming the effectiveness of ANSR, we additionally compare the evaluation results to the results on other baselines. Figure 5.10 illustrates the performance comparison results. Here, k means the number of items present in the prediction list. Through

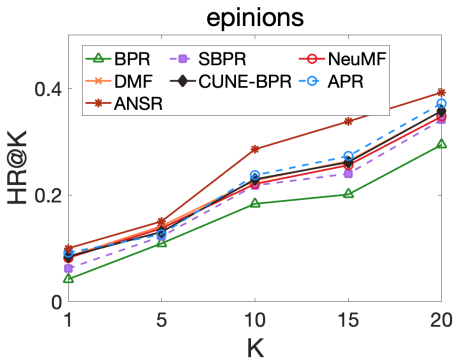
the observation of the experiment results, the following conclusions could be attained.



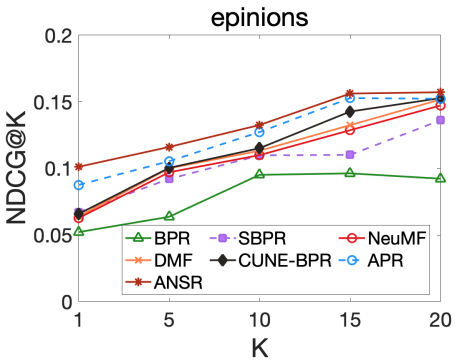
(A) HR@K



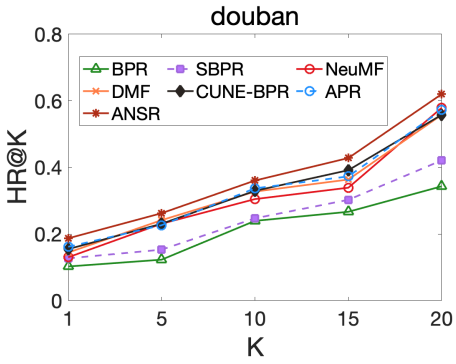
(B) NDCG@K



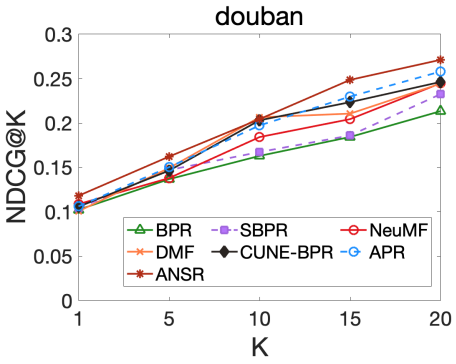
(C) HR@K



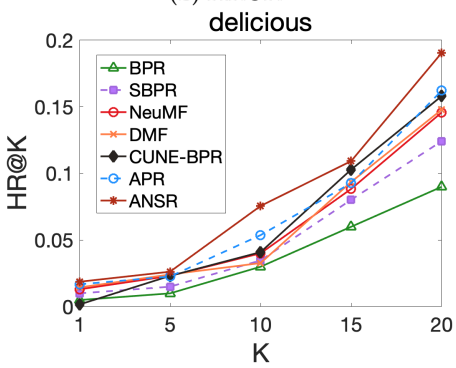
(D) NDCG@k



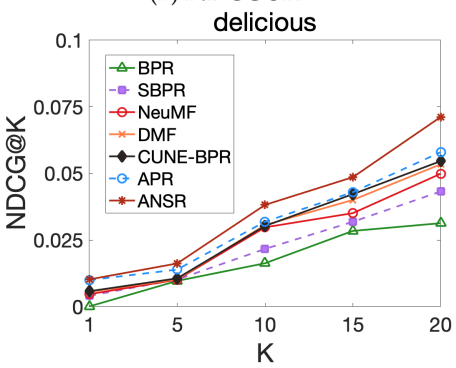
(E) HR@K



(F) NDCG@K



(G) HR@K



(H) NDCG@K

FIGURE 5.10: The overall comparison results.

1) The top red line represents the performance results of the proposed ANSR. It is easy to note that our proposed ANSR was superior to other methods in different K values. This result also fully confirms the effectiveness of the ANSR model, which can effectively enhance the performance of the recommendation system.

2) In comparison to the traditional factor decomposition-based algorithm BPR, deep learning-based model ANSR, NCF and other models significantly enhance the functioning of recommendation systems. The reason was that the algorithms based on traditional factorization are difficult to deal with large-scale data. In addition, the model based on deep learning can fit any function, but the complexity of the model determines that it is difficult for general factorization model to get the same effect. At the same time, the activation function in the model makes the deep learning model have stronger nonlinear learning ability and can learn the more complex nonlinear relationship between users and items. Furthermore, we could observe that APR performs even that CUNE-BPR which was based social information. This was due to the high learning ability of adversarial learning, which also proves the superiority of deep learning-based recommendation algorithm.

3) Compared with the model based on users' feedback, integrating users' social information into recommendation model could enhance the functioning of recommendation systems. For example, SBPR performs better than BPR, CUNE-BPR performs better than NeuMF, DMF. Especially, SBPR can perform almost similar with NeuMF on some small data sets such as Ciao. This was due to the fact that leveraging social recommendation could enrich the intelligence of users by using their social data. As a result, it could mitigate the influence of data sparsity. This also proved that exploiting social information could significantly enhance the performance of the recommendation systems.

4) Since all models aim to find representations of users and items and utilize the same objective function, it could be concluded that the ANSR model is better at learning user representations. ANSR can learn better user representation because the model not only considers how to rationally use social network to learn user embedding and promote the recommendation ability. In addition, the attention neural network was also effectively used to capture the dependence among latent factors, which helps us extract more effective information, thus leading to better prediction results.

5.4.2 Model Analysis

Attention Module Analysis

During last performance comparison experiment, we had verified the performance of ANSR by using the overall performance comparison experiments. However, we do not know why the algorithm can effectively improve the performance of the model. In the second experiment, ablation study was carried out to analyze the contribution of our proposed approaches. The ablation study is to analyze the influence of different modules on the experimental results so as to obtain the effects of different modules. In order to analyze the contribution of different modules to the experimental results, we will transform the proposed models into different variation models and compare the results. In Chapter 4, we have introduced a general framework for deep learning-based social recommendation models, which were based on NeuMF. Our proposed ANSR algorithm was also based on the NeuMF and added new modules, so we chose the NeuMF as the baseline for other variant models to prove the effectiveness of the added modules. Among them, the first variant model called SNeuMF was based on the assemble methods which exploiting social data to enhance the user representation. The second variant model called SNE which utilized the network embedded module to capture the user's social information. In the fourth model which denoted as SN-CUNE, we tried to utilize the algorithm provided by CUNE-BPR to optimize the influence of social relations on the representation of user preferences. Here we used cosine function to calculate the similarity of different users, and assign a value to the influence of different social neighbors on users' preferences. To clearly express different variant model, Figure 5.11 was used to demonstrate and symbolize the different variant model, while SNE and SNU were used to symbolize the variant model that only utilize the network embedding module and attention-based module, respectively.

Table 5.2 shows the performance comparison on several variant models utilizing four data sets. From the experimental results demonstrated in this table, several subsequent could be drawn. The SNeuMF results were relatively poor. The reason was that assumes that all the user's friends have the same effect on the user. SNE performed a litter better than SNeuMF. This was because we utilized the network embedding to enhance the user's representation. However, we have known from social regularization methods

variant model	principle	user repersentation
NeuMF	basic DL-based model without any social information	u_i
SNeuMF	utilizing social information based on assemble methods	$u_i + \sum u_{fl}, f \in \mathbb{F} = \{u_{f1}, u_{f2}, \dots, u_{fn}\}$
SNE	utilizing social information by Network Embedding	$u^{i^*} + \sum u_{fl}{}^*,\ u^{i^*}$: NE (u_i)
SN-CUNE	utilizing social information by NE and similarity function in CUNE-BPR	$u^{i^*} + \sum \alpha * u_{fl}{}^*,\ \alpha$: Sim$(u^{i^*}, u_{fl}{}^*)$
ANSR	proposed model utilizing social information by both NE and Attention	$u^{i^*} + \sum \alpha * u_{fl}{}^*,\ \alpha, u_{fl}{}^*$: $Attention()$

FIGURE 5.11: The illustration of the variant models.

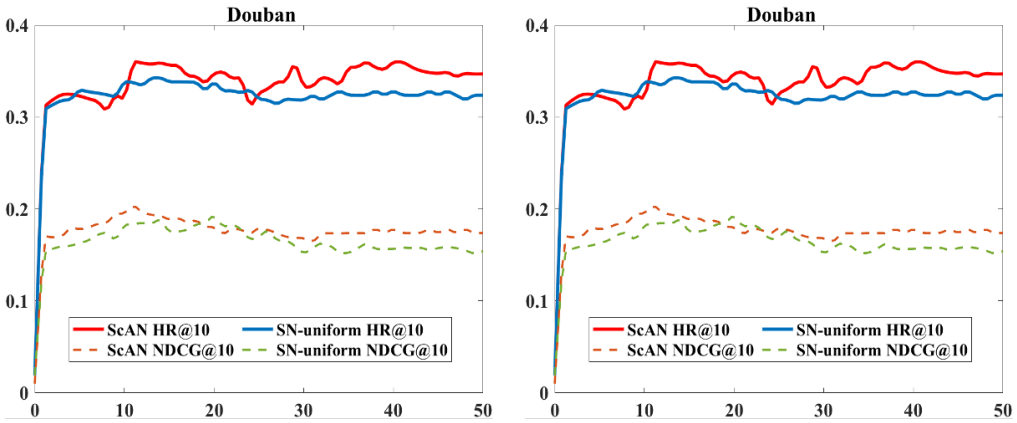
that the heterogeneity of a network leads to the difference in the contribution of different friends to user preference. Therefore, the SNeuMF results did relatively poorly. At the same time, we could observe that the SN-CUNE perform better than the SNeuMF results. Obviously, in the real world, different social relationships differ in the degree to which they contribute to users' preferences. For example, we usually use the strength of a user's social relationship as an example. A close social friend may be highly consistent with the user's preferences, or have a greater influence. Therefore, when we utilized the principle in CUNE-BPR and employed these social regularization terms to constrain the social neighbor's representation, SN-CUNE could help improve the accuracy of the variant model. Nevertheless, instead of using the above algorithm, our proposed ANSR algorithm based on the attention mechanism can further improve the performance of the model. This was because the weight based on attention can further optimize the social regularization coefficient by changing with the context, compared to calculating the fixed influence of friends on users using the cosine formula. At the same time, our model could effectively extract more useful feature information from redundant information and solve the problem of data noise. As a result, from the experimental results of these variant models, we can conclude that the superposition of the two modules shows the best effect. Through the network embedding module, we can learn more social semantic information about users. At the same time, through this social semantic

TABLE 5.2: The comparison results on variant models.

	Ciao		Epinions	
	HR@10	NDCG@10	HR@10	NDCG@10
NeuMF	0.4302	0.2546	0.2209	0.1100
SNeuMF	0.4438	0.2587	0.2402	0.1224
SNE	0.4495	0.2602	0.2415	0.1247
SN-CUNE	0.4520	0.2592	0.2542	0.1284
ANSR	**0.4751**	**0.2614**	**0.2860**	**0.1326**
	Delicious		Douban	
	HR@10	NDCG@10	HR@10	NDCG@10
NeuMF	0.0592	0.0298	0.3250	0.1889
SNeuMF	0.0642	0.0312	0.3291	0.1902
SNE	0.0677	0.0320	0.3435	0.1926
SN-CUNE	0.0681	0.0349	0.3503	0.1928
ANSR	**0.0756**	**0.0382**	**0.3607**	**0.2044**

information, the attention layer can learn more effective representations for the user. These two modules promote each other and cannot be separated from each other.

In order to make the results more intuitive to be observed, we use the graphical representation method to draw a comparison diagram of the experimental results. Based on the above results, we further analyze the impact of the attention module on model performance. Where, ScAN represents the use of the attention model in the variant model, while SN-uniform represents the use of the uniform weight assignment strategy in the variant model. Figure 5.12 shows the comparison of ScAN and SN-uniform performance in each training iteration under optimal parameter Settings. Compared with the uniform weight distribution model, the performance of ScAN on both data sets is the best. When K=10, our method improves 8.54% and 10.87% on HR and NDCG, and 5.22% and 5.53% on douban, respectively, compared with the baseline method. Since ScAN and Sn-uniform are both models based on multilayer perceptron, the convergence speed is fast. However, with the increase of the number of iterations, the performance began to fluctuate or even decline, which indicates that excessive iterative calculation may lead to overfitting of the model, thus affecting the performance of the model.



(A) HR@10 (B) NDCG@10

FIGURE 5.12: Performance of HR@10 and NDCG@10 varying iterations on Epinions and Douban

Network Embedding Module Analysis

In the previous discussion, we noted that the embedding layer used random initialization method, for example, users' embedding were obtained from parameters compatible with Gaussian distribution, is unable to fully receive social semantic signals from users. Through network embedding technology, each user could be represented by a node embedding vector that include social semantic signals. Intuitively, if two users are connected with each other, they may have some similar interest. The consequence of such embedding is that the users with similar preferences will have close representations in the latent space. By doing so, we can learn a better embedding for each user to encode the social semantic signals. In order to understand the contribution of network embedding module, we conducted next experiment to compared the result of utilizing Gaussian initialization and the variant models utilizing different network embedding methods. We will use _[method] to represent models that use different network embedding methods. Table 5.3 shows the comparison of the results of variant models deformations on the four data sets. From the results, we can draw a conclusion that the performance of utilizing embedding method initialized by Gaussian distribution is the weakest in the embedding layer, while the adoption of network embedding method as embedding method has generated effective improvement as a result. We tried to analyze why SNDE worked best. This may be due to the advantages of the SNDE algorithm. In SNDE [69], the node embedding can not only learn the similarity between two connected nodes, but also learn the

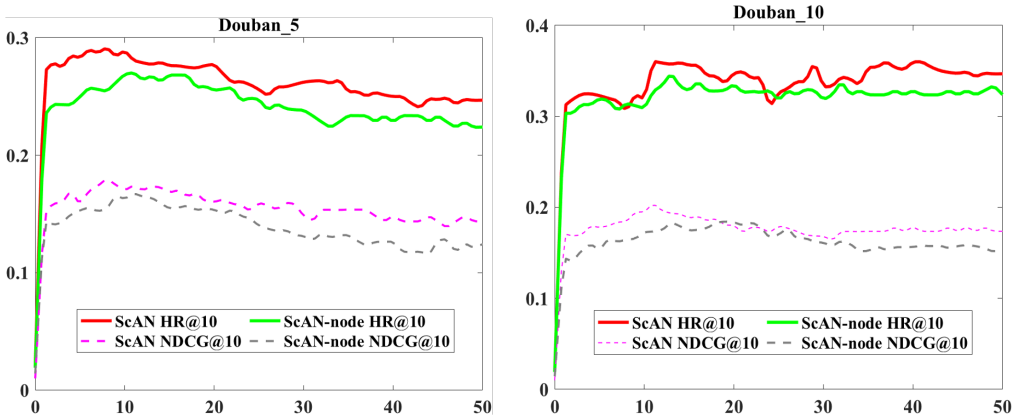
TABLE 5.3: The comparison results varying different network embedding methods.

	ciao		epinions	
	HR@10	NDCG@10	HR@10	NDCG@10
_GD	0.4326	0.2598	0.2402	0.1198
_hope	0.4598	0.2603	0.2751	0.1287
_n2v	0.4602	0.2607	0.2785	0.1292
_line	0.4632	0.2602	0.2815	0.1307
_snde	**0.4751**	**0.2614**	**0.2860**	**0.1326**
	delicious		douban	
	HR@10	NDCG@10	HR@10	NDCG@10
_GD	0.0591	0.0229	0.3202	0.1673
_hope	0.0708	0.0340	0.3535	0.1947
_n2v	0.0713	0.0347	0.3541	0.1972
_line	0.0723	0.0362	0.3581	0.2017
_snde	**0.0756**	**0.0382**	**0.3607**	**0.2044**

secondary similarity between two common neighbor nodes that are not directly connected. This has practical implications for social recommendation systems.

In addition to the above points, we also found through the experimental results that the adoption of network embedding can accelerate the convergence speed of the neural network. To prove this finding, we carried out the following experiment. In order to enrich the experimental data, we divided *Douban* data set into two data sets with different data sparsity. We deleted user data with less than 5 rating records to form the data set named $Douban_5$, at the same time, we deleted user data with less than 10 rating records to form the data set named $Douban_10$. ScAN represents the use of network embedding model, while Scan-Node represents the model using a random initialization method. These two models were experimented on these two different data sets. Figure 5.13 shows the performance changes in the HR@10 and NDCG@10 results as the number of iterations increases. We found that the ScAN results were relatively superior and could accelerate the convergence rate of the neural network. In particular, when $k = 10$, ScAN performs better than $ScAN_node$ and received 7.88% on HR and 7.45% on NDCG respectively on $Douban_5$. While, ScAN performs better than $ScAN_node$ and received 3.80% on HR and 4.77% on NDCG respectively on $Douban_10$. In addition to the above results, we also found that the model training converges

faster on the $Douban_5$ dataset. The reason may be that the social recommendation system is a solution for the the data sparsity problem. When training data set is sparse, this advantage is amplified, so the convergence speed of the network is accelerated. However, when the data set is relatively dense, the performance enhancement reflected by social information is not as obvious as that reflected by sparse data.



(A) HR@10 (B) NDCG@10

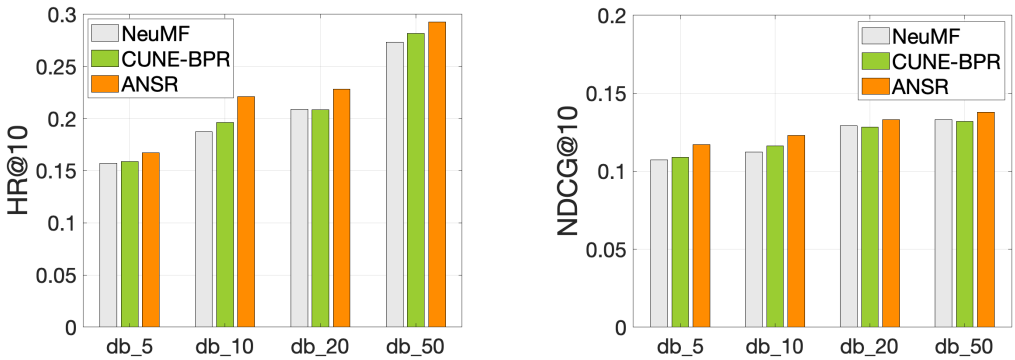
FIGURE 5.13: Performance of HR@10 and NDCG@10 varying iterations on $Douban_5$ and $Douban_10$

5.5 Discussion

5.5.1 Data Sparsity Problem Study

The core goal of social recommendation system is to alleviate the data sparsity problem of traditional collaborative filtering methods. In the experiment of this section, we will analyze the effect of the proposed algorithm on alleviating data sparsity. In order to enrich the experiment data, we still divide the original data set into four data sets according to the degree of data sparsity. The sparsity of the data is measured by the number of the user's rating records. In this experiment, we divided the data set into four data sets (0-5, 5-10, 10-20, 20-50) according to the number of rating records of users. Next, we will carry out experiments on these four experiment data sets to check if the model have the ability to recover the data sparsity problem. We chose NeuMF and CUNE-BPR as the baseline algorithms in the experiment. Figure 5.14 shows the comparison results of different algorithms across three data sets. From the histogram of results, we can easily see that our model always

performs relatively well on data sets with different degrees of sparsity. The reason for this is that NeuMF is a collaborative filtering method based on deep learning that uses only the user's rating data, so data sparsity can affect the results of the model to some extent. ANSR, which is a social recommendation algorithm based on deep learning, performs better than NeuMF by learning more features from the user's social domain, enabling the model to learn better user representation and improve recommendation performance. Compared with CUNE-BPR, our algorithm can further improve the accuracy of user representation. In particular, we find that ANSR performs better when the data set is relatively sparse. However, with the increase of data sparsity, the superiority of the model began to decrease. This may be related to the rating data of users. It can help the model learn rich user representations since the rating data is diversity.



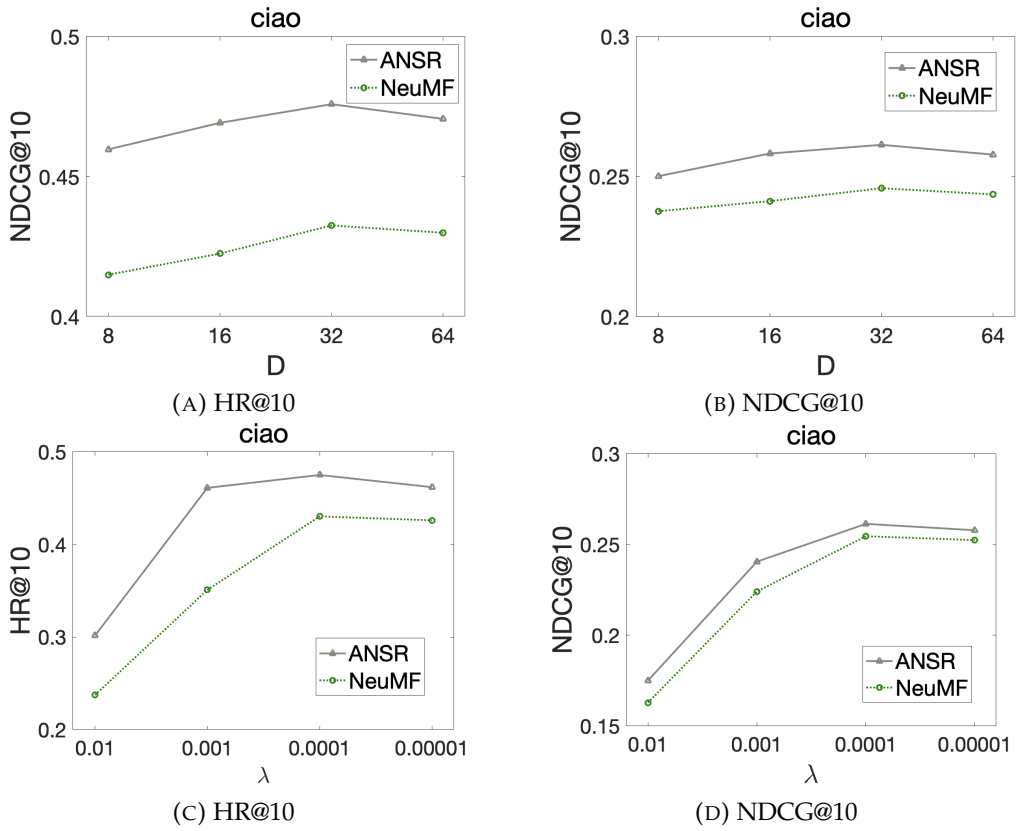
(A) HR@10 (B) NDCG@10

FIGURE 5.14: (a), (b) are HR@10 and NDCG@10 of NeuMF, CUNE-BPR, ANSR on the Douban dataset w.r.t. different data sparsity levels.

5.5.2 Discussion

In the following experiments, we will analyze other factors that affect the performance of the model, especially the hyper-parameters. Regularization and dropout are both commonly used techniques in neural networks, whose main purpose is to control the complexity of the model and reduce the influence of overfitting. The essence of overfitting is that the model is too complex, which weakens the generalization ability. Model complexity is important for generalization of the model. Regularization is a general term for suppressing overfitting methods, which reduces the complexity of model by dynamically adjusting the value of model parameters. This is because when

the value of some parameters is small enough, the corresponding attributes of the parameters have little effect on the result, which is essentially the effect of non-related attributes. To obtain a stable and high generalization model, it is necessary to utilize these techniques to prevent model from overfitting. Therefore, we conduct experiments and compare the results using different hyper-parameters to help us choose an appropriate hyper-parameter to control the complexity of the model. Figure 5.15 shows the comparison of model results when hyper-parameters take different values. From the results, we can observe that as the regularization parameter λ changes, the model's performance changes accordingly. When the regularization parameter λ is too large, the model performs relatively poorly. This is because when the regularization parameter λ is too large, the more stringent the constraints, the model loses its ability to fit. The regularization parameter λ can't be too small too. If it is too small, the regularization parameter λ loses its ability to restrict the parameters, and the network will tend to rely on certain characteristics, which reduces the generalization ability of the model. Therefore, we should choose a more reasonable regularization coefficient in a limited range. Meanwhile, embedding size has a corresponding impact on the model performance. We find that the model performance is relatively poor when there is small embedding dimension.on the contrary,the model performance is also affected when the embedding dimension is too large. This is because embedding size also affects the generalization ability of the model. When the size is too small, the model cannot learn effective features, thus resulting in the lack of fitting ability of the model. When the size is too large, the complexity of the model will greatly increase, which is easy to lead to overfitting. Therefore, it is very important to select the appropriate embedding dimension for the fitting ability of the model.



(A) HR@10 (B) NDCG@10

(C) HR@10 (D) NDCG@10

FIGURE 5.15: (a) and (b) are HR@10 and NDCG@10 of ANSR on the Ciao dataset w.r.t different embedding dimension size D, respectively. (c) and (d) are HR@10 and NDCG@10 of ANSR on the Ciao dataset w.r.t different regularization parameter λ, respectively.

5.5.3 Threats to the Experiment

In the above experiment, we have discussed the influence of the hyper-parameters of the model on the experimental results. In this section, we will continue to discuss other factors that may influence the results of the experiment. Through different processing of the data set, we have found that different evaluation protocols will lead to great deviation of the experimental results. For example, we have discussed how to generate test data. We adopted the leave-one-out protocol to generate the test data. The meaning of leave-one-out protocol was that we randomly selected an item from his implicit feedback and put it into the test data set for each user. Then, we will random select 300 items as the test set. The advantage of doing this was to save a lot

TABLE 5.4: Leave-one-out vs Hold-out

		leave-one-out protocol	hold-out protocol
Douban	HR@10	0.3607	0.0598
	NDCG@10	0.2044	0.1203

of testing time, so that we can get the experimental results as soon as possible. Of course, there are other ways to generate test data. For example, we could directly divide the data set into a training set, a validation set, and a test set. At this point, the test will iterate over all item data, resulting in slower test time. Table 5.4 displayed the comparison results when different protocols were selected.

The deviation of the experimental results is also of practical significance. Because in real life, online platforms can have millions of items. Recommendation results from all items are unrealistic. Typically, recommendation results are generated from a set of candidates. This candidate set is a set smaller than the total item set that is generated by a crude algorithm in advance according to the category to be recommended. This is similar to the small test set of 300 items generated when we used the leave-one-out protocl. Therefore, the results of the negative sample could better reflect the actual results.

Chapter 6

Adaptive Social Influence Learning for Recommendation via Heterogeneous information Networks

6.1 Motivations and Rationales

In the previous chapter, we introduced that the social-aware recommendation system can recover the data sparse problem when use the users' social information. The prediction result of the preference is effected not only by the users' past rating behaviors, but also influenced by the users' social neighbors' preference. However, the data of explicit user-user social links also has the problem of data sparsity, which is much higher than the user-item rating data. In most real world systems, such as *Epinions*, the social relations data is also sparse, with a trust density of just 0.029%. The problem of data sparsity is not only in the user's rating data, but also in the social aspect, users rarely establish explicit social relations with other users. Figure 6.1 is an Example of the social friends' size of users grouping by the number of the observed relation data in the dataset we used in our paper. From this figure, we can observe that Nearly half of the users only have zero to five social friends. Only 100 users have more than 50 social friends. Therefore, using existing social data to help predict users' preferences is very limited. However so far, few studies have focused on the sparsity issue of users' social relation data. Therefore, we need to consider how to overcome this shortcoming and help users expand more relations with the other users who have similar interests to help predict users' preferences, and thus reduce the impact of the

sparsity problem on the performance of the social-aware recommendation system. Besides, users' social relations do not always play a positive role in predicting users' preferences. Although we would like to find more potential user relations to infer users' preferences, we need develop a specific model to verify whether these user relations have a positive effect on the predicted results.

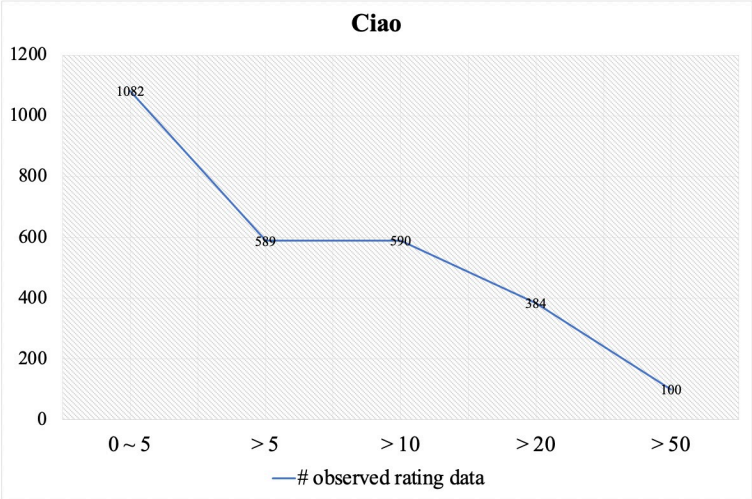


FIGURE 6.1: Example of the social friends' size of users grouping by the number of the observed rating data.

Last but not least, the current end-to-end recommendation system based on deep learning is difficult to interpret the recommendation results. Recommendation explanation is to provide users with recommendations results and at the same time give the reasons for recommendation. Deep learning is basically viewed as a black box, and it's hard to explain. We can only understand the input and output of the deep model, but it is difficult to understand how the black box works. This leads to the difficulty in predicting and debugging the results of deep learning models, which ultimately affects the understanding of the models and the improvement of the results. Therefore, How to make users understand the recommendation generating mechanism is also valuable for researchers and recommendation algorithm developers to better understand the operating principle of the algorithm, which can help us to answer questions such as why these recommendations are given to me,

how to improve the recommendation effect, and what data or features can affect the recommendation results.

First, for the purpose of solving the mentioned shortcomings, it is necessary to find a way to enrich the user's social datas. Accordingly, heterogeneous information network (HIN) was utilized to help users find new social neighbors who will share similar interest, which can also be regarded as users' implicit social relations, and utilized these implicit social relations to solve the mentioned problem. In social theoretical studies, an important explanation for the relatively sparse social network data of users is that some users rarely explicitly establish social connections with other users. As a result, the explicit social relations we can observe in social networks represent only a limited portion of a person's real social network. Therefore, we want to find those users who have no explicit relations with the user, but share similar preferences with the user, and thus are likely to be friends of the user. We believe that users with similar preferences and the same friends as target users can be used as auxiliary social data of target users to make up the missing part of social relations. For example, in Figure 6.2, u_1 and u_4 are "strangers" in the social network, there is no clear social relationship between them, but they are both interested in the same movie, so it can be assumed that u_1 and u_4 may be potential friends, and the rating information of u_4 can be used to help predict the preference of u_{41}. Motived by [70, 71], it is naturally to combine the user-user social network and the user-item rating network into a whole network, which could be denoted as a heterogeneous information network. Even if the user is not directly connected in the original social network, an indirect connection can be created in the HIN through the user-item link. Therefore, establishing a HIN to help users find indirect social relations will help the social recommendation system alleviate the problem of social data sparsity. Another problem is that the constructed heterogeneous information network will be a complex network structure, therefore how to reasonably find users' potential social friends with similar preferences also needs to be considered.

In the above discussion, we have discussed how the obtaining auxiliary user social links can help solve the problem of data sparsity. However, we need a reasonable way to use these auxiliary social links. Obviously, these newly acquired auxiliary social links do not necessarily play a positive role in predicting users' preferences. It has been discussed in the social regularization method that different social neighbors have different effects on users.

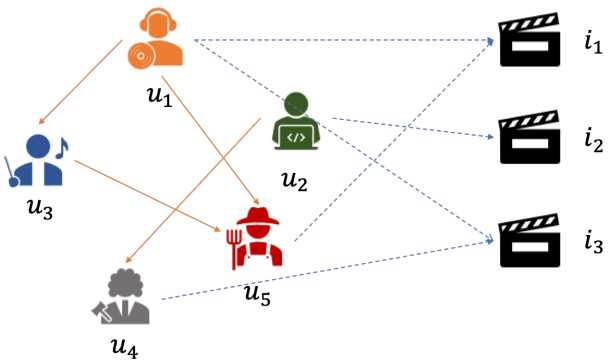


FIGURE 6.2: Example of an HIN in a recommendation system. The dotted blue line represents that a user has rated an item, and the solid yellow line represents the user trust his/her friends.

A simple example is the diversity of users' social relationships. In general, the closer the relationship between users, the stronger the influence between users should be. If we just utilize these auxiliary social relations equally, they are not necessarily effective for the model and may even become the input noise of the model, thus reducing the performance of the model. Therefore, the usage of these auxiliary social relations requires a reasonable evaluation of their quality. To solve this problem, we put forward a model based on attention neural network, which will automatically assign different weights to different social relations, so as to control the influence of social relations on users. In general, when a social relationship contributes more to the prediction of user preferences, it should be given higher weight, while when a social relationship is not important to the prediction or even makes noise, we need to punish the social relationship by assigning it as little weight as possible. Through the above process, we can use these social auxiliary links more properly. Another important reason is the dynamic nature of social impact. Even the same social relationship may not produce the same contribution to the same user. For example, as shown in Figure 6.3, when we want to recommend a science fiction movie to Alice, Mike's influence may be higher than Angel's. Because Mike has more experience in science fiction movie. However, when we want to recommend a romantic movie to Alice, Angel may have a higher influence. Because the romantic movies played a large part in angel's historical feedback data. Therefore, it should be noticed that the user influence varies with the context and does not always remain the same. It

should be considered how to learn dynamic influence value during difference context. Another important reason is that the model based on attention network can improve the interpretability of the recommendation system. We can judge the influence of different factors on the recommendation results according to the different weights.

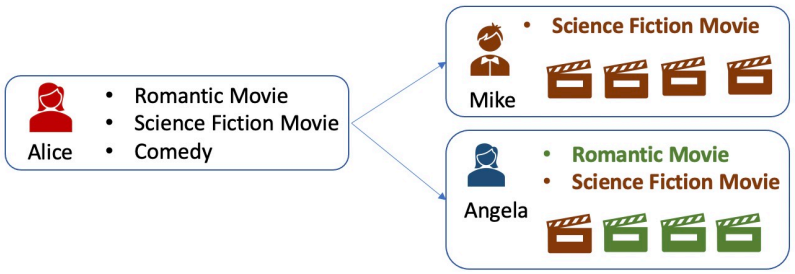


FIGURE 6.3: Example of the social influence.

6.2 Methodology

6.2.1 Overview of HASRec

The purpose of the proposed model is to infer user preferences and provide a top-k recommendation item list to user. To achieve this purpose and solve the problems mentioned in the previous section, our proposed model will consist of two main parts, the first part is the missing link identification module and the second part is the attention-based recommendation module. The missing link identification module was proposed to figure out social data sparsity issue. After generating new user social neighbors, we used the user's new social network as input of the attention-based recommendation module. This module aim to utilize the users' social relations in a more reasonable way to learn a better representations for the user. In the end, the model will generate a recommendation list for users. In the subsequent part, these two parts will be introduced in details.

6.2.2 Missing Links Identify Module

The primary purpose of the proposed missing links identify module it to discover the unobserved relations between users to enrich the users' social data.

To achieve this goal, our missing links identify module is mainly divided into three steps: 1) Establish a heterogeneous information network; 2) Generate users' social semantic corpus by random walk. 3) Learn the latent representation of users via word2vec model. Through this three steps, we can find users' top-K similar friends by calculating the similarity between users. The framework of this module was shown in 6.4.

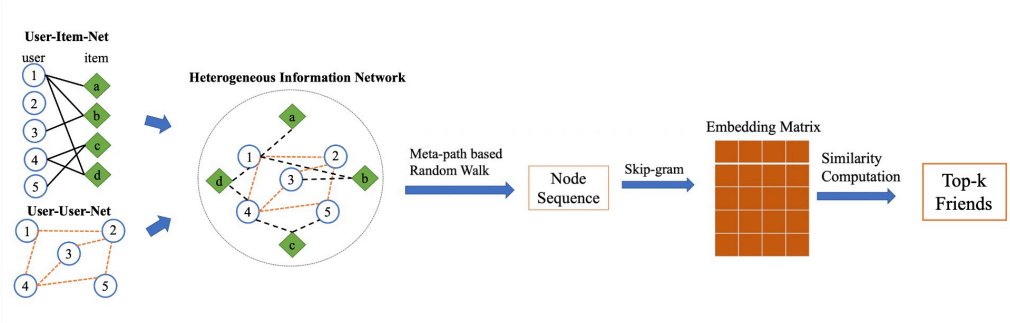


FIGURE 6.4: The framework of Missing Links Identify Module

- **Heterogeneous Information Network Construction**: In the first step, we need to connect the user-item rating network and the user-user social network together to form a complete heterogeneous information network. The process of generating HIN was demonstrated in Figure 6.5. User-item rating network (User-Item-NET) using rating relations to connect two sets of nodes from different sets, namely users set and items set. The connection between user-item represents a user-item rating feedback. For example, the edge between *user* 1 and *item a* indicates that the *use*r 1 has rated *item a*. Since both network structures contain the same node type 'user', we can connect the two networks together to form a new network structure through the user nodes. In this network structure, there are two types of edges between nodes, namely, social relations and rating relations. After the new network is built, we can use indirect connections between users to find other users that were not connected in the original network structure. Moreover, as the HIN is rich in semantic information, for example, two users connected by the same item can be considered to have similar preferences, which can help us to obtain more reliable users with similar semantic information. Next, we will find users with similar semantics by random walk.

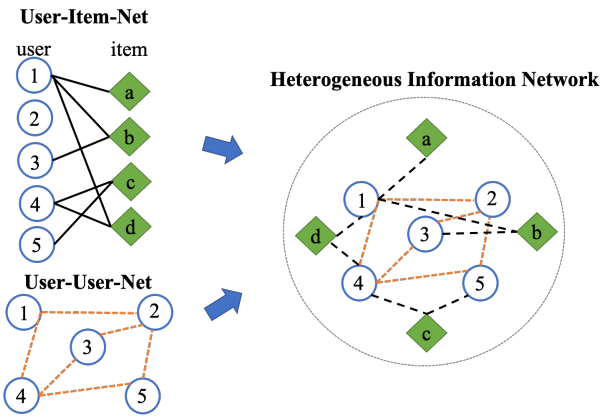


FIGURE 6.5: the Construction of a Heterogeneous Information Network from a user-item rating network and a user-user social network.

- **Generate users' social semantic corpus by random walk**: Heterogeneous networks contain more semantic information than homogeneous networks, and if we just use methods like DeepWalk and Node2Vec, we can only simply number the different nodes and then conduct a random walk. However, this approach ignores the type information of nodes and treats user and item as nodes of the same type, which loses a lot of semantic information. To solve this problem, we use the walk strategy based on the meta-path to generate node sequence. A meta-path can be understood as a predefined sequence of node types. In general, a meta-path is denoted as a path in the following form:

$$A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} ... \xrightarrow{R_l} A_{l+1} \tag{6.1}$$

wherein R denotes a composite relation between a start object A_1 and the end object A_{l+1}. For example, the UIU represents $[user - item - user]$. By the definition of meta-path, a series of node sequences with semantic and structural dependencies between different types of nodes can be obtained in heterogeneous information networks. The meta-paths utilized in this paper are presented in Figure 6.6.

For example, the meta-path $P_1 : user \xrightarrow{click} item \xleftarrow{click} user$, which is denoted by UIU, represents that two users have rated the same item which illustrate that they have similar interest, and a meta-path P_2 :

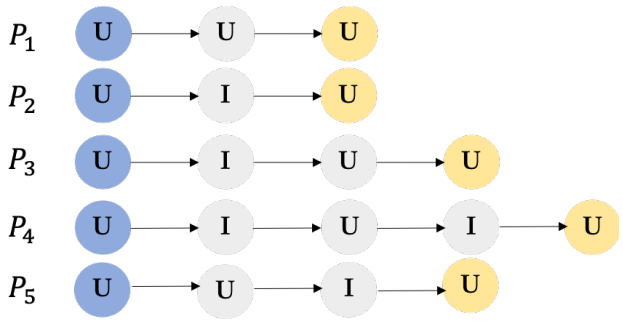


FIGURE 6.6: The meta path designed in our work. U denotes the user and I denotes the item.

$user \xrightarrow{click} item \xleftarrow{click} user \xleftarrow{trust} user$, which is denoted as $UUIU$, represents that a user may have similar preference with another user who have similar taste with his social neighbor. Given a meta-path $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} ... \xrightarrow{R_l} A_{l+1}$, we use the meta-path to guide the next node in a random walk, the transition probability is defined as:

$$p(v^{l+1}|v^l, \rho) = \begin{cases} \frac{1}{|N_{l+1}(v^l)|}, & (v^{l+1}, v^l)\epsilon\rho \\ 0, & otherwise \end{cases} \tag{6.2}$$

where v^l denote the current node in the walk, and v^{l+1} is the next node in the walk. $N_{l+1}(v^l)$ denote the node type of the neighbor node of v^l. $\frac{1}{|N_{l+1}(v^l)|}$ means that if $N_{l+1} = I$, the next node will randomly selected with equal probability. Through random walk based on the meta paths defined above, we will get paths rich in different semantic information. These generated node sequence will be put into a set which is called social corpora. It is found that the probability distribution of nodes in the random walk follows is similar to the word frequency distribution in the context corpus. Therefore, inspired by word2vec, we can use the skip-gram model to learn the semantic characteristics of each node. Next, we will learn the representation of each node by treating these node sequences as sentences in the language model.

- **Embedding Learning via skip-gram**: Inspired by word2vec, we utilized the skip-gram model to learn the node embedding. Word2vec is a model of learning semantic knowledge in an unsupervised way from a large-scale text corpus. It represents the semantic information of words

by learning word vectors. In other words, an embedded space is constructed to make words with similar semantics close to each other in the space. Skip-gram model is a fully connected neural network with only one hidden layer. Its training process can be regarded as a Fake Task. In other words, the purpose of training model is not to use the trained model for downstream tasks, but to obtain the weight matrix of hidden layer, because the weight matrix contains the weight information of all words. The optimization function of skip-gram model is to maximize the average log probability of a given word appearing in a corpus. According to skip-gram's idea, we will maximize the co-occurrence probability of nodes in each random walk. In general, the objective function is defined as:

$$max \sum_{v \epsilon \boldsymbol{V}} \sum_{v^k \epsilon \boldsymbol{C}(v^l)} logp(v^k|v^l) \tag{6.3}$$

where v^l is the current node, $C(v^l)$ is the context of node v^l, which the widow size is fixed. $logp(v^k|v^l$ is usually calculated by a softmax function, which is defined as:

$$logp(v^k|v^l) = \frac{e^{x_{v_k} x_{v_l}}}{\sum_{v \epsilon C(v)} e^{x_v x_{v_l}}} \tag{6.4}$$

where x_{v_k} represent the embedding vector of node v_k. We only employ the same type node to optimize the objective function. After defining the objective function, we can use gradient descent to optimize the objective function.

Through the above steps, we first generate the corresponding semantic corpus for each node through a meta-path based random walk, then use the stochastic gradient descent method to optimize the skip-gram model's objective function. Through training, a vector representation of each node can be obtained. We can calculate the cosine similarity of each pair of users through the embedding vector, and then find most K similar users for each user, namely the new top-K social neighbor.

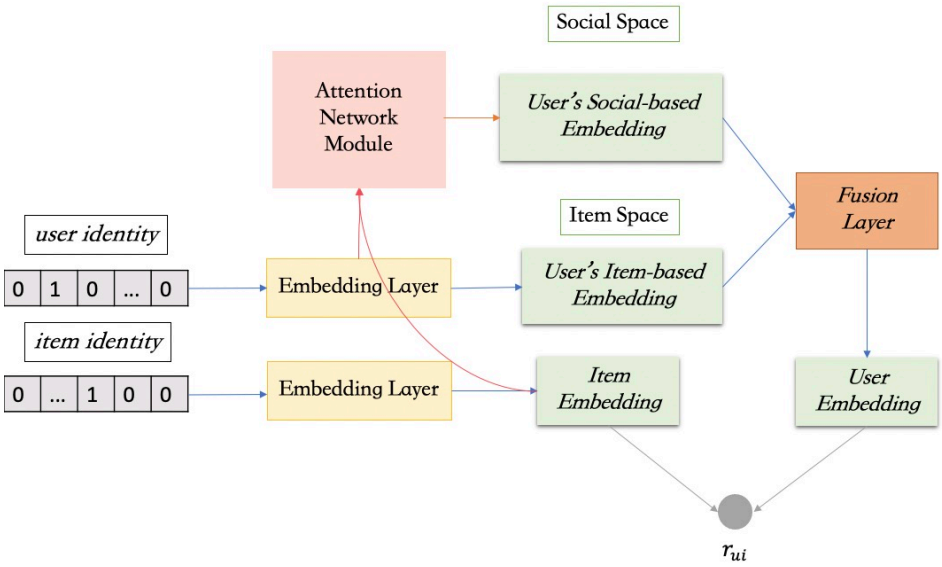


FIGURE 6.7: The architecture framework of Attention-based Recommendation Module.

6.2.3 Attention-based Recommendation Module

In the previous section, we utilized the meta-path based random walk to find neighbor users with the similar preferences, and thus to enrich the users' social information in order to recover the data sparsity problem which existing in the training data. In this section, we will use the generated user social semantic neighbors to help predict user preference. Similar to other social-based recommendation systems, we also assume that users' preferences come from two aspects: the users' history rating data and history rating data from the users' neighbors. To increase the interpretability of the system, we divided the factors of users' latent preferences into two aspects, item-based and social-based representation. The first is used to achieve the user's preferences based on the user's rating history. The latter representation extrapolates the effect of a user's social relations on users' preferences. The overall architecture of attention-based recommendation module is shown in Figure 6.7. The following section will give the details of the Attention-based Recommendation Module.

Embedding Layer The input data of attention-based recommendation module were the user's interaction data which has a property of item and the other is social interaction data. More particularly, each user and item are given a unique identity and then stored in a file. One input instance for user

will be the user-item pair in addition with the additional social neighbors for this user, which were denoted as a friends list. These index number will be the initial input for the embedding layer. Obviously, the input data needs to be encoded before it can be understood and calculated by a computer. The embedding layer can be viewed as a simple matrix look-up table that transforms positive index into a dense vector of fixed size. Through embedding layer, each user and item will be represented by a dense and low-dimensional vector. Then, these embeddings will be put into next layer for further processing. Besides the user's feature, the user's friends feature will also be put into next layer to incorporate social information the enhance the model.

Attention Layer

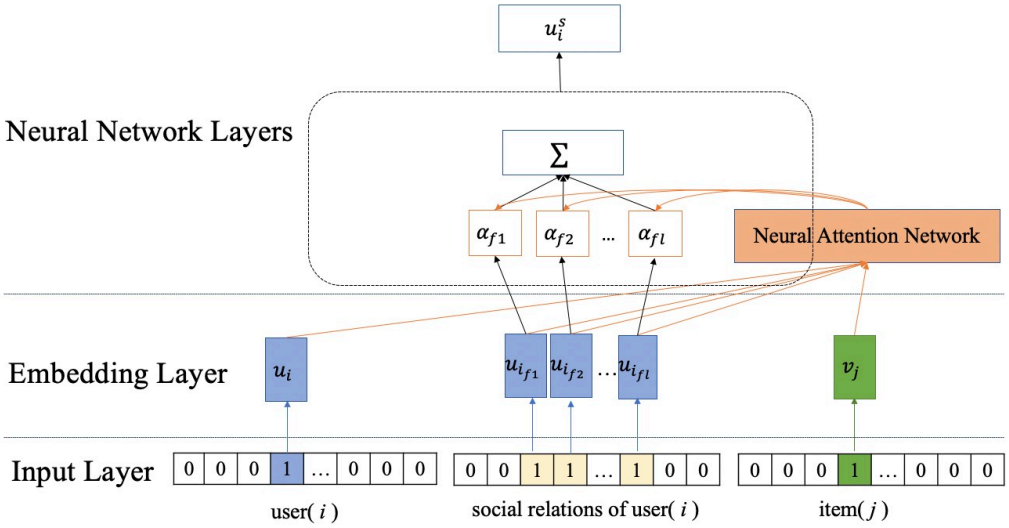


FIGURE 6.8: The framework of the attention layer.

In the attention layer, we will focus on the task of incorporate the user's social information into this model. The framework of this layer was demonstrated in Figure 6.8. Of course, we can take the simplest approach, such as the one used in SocialMF, which averages all the features of a user's social neighbors to get a social-based representation of a user. The problem with this is that not all social neighbors are properly utilized in the model. The irrationality mainly reflected in two problems. Firstly, different friends will have different influence on user. For example, close friends and family, these connections could be viewed as strong relations because they often interact with each other. Compare with this, Classmate and colleagues could be viewed as weak relations for that they may not often interact with each other.

Obviously, Social friends with strong connections tend to have a higher influence on users than those with weak relations. Assign uniform weight to all user's friends will not work well. Therefore, it is necessary to consider how to assign different influence value to user's different friends to model this heterogeneous social effect. Another problem of the influence in social effect is that the influence of social relations is not universally applicable to any context. We refer to the user's choice of different items as a context. When user faced with different items, the social influence changes dynamically. Therefore, it should be noticed that the user influence varies with the context and does not always remain the same. It need to be considered how to learn dynamic influence value during difference context.

Based on the above discussion, we propose to use the attention network to make reasonable use of users' different social relations. We utilized the same idea provided by an attention mechanism to assign different and dynamic weights to different social relations. You may wander 'Why this weight is dynamic'? The reason is that the contribution of a friend is dependent on his relation type and the target item's property. when the user interacts with different items, the context will change. Therefore, the attention value will also change due to the context change. Generally, under the attention neural network, the weight is called an attention score. First, we need to calculate the correlation between the user feature and friend feature and item feature. The following function demonstrate how to represent this correlation vector:

$$a^*_{if} = \sigma(W_1 * U_{if} + W_2 * U_i + W_3 * V_j + b) \tag{6.5}$$

Where, W_1,W_2 and b were the weights and bias utilized in this attention layer. U_i represents the user feature, V_j represents the item feature and U_{if} represents the friend features. Then, we utilize $softmax()$ function to convert the correlation vector into the probability distribution with the sum of all element weights being 1. As a result, the social neighbors who are more useful to the prediction result will assign a higher weight through the attention score learned from the $softmax()$ function. The process was defined as followings:

$$a_{if} = \frac{exp(a^*_{if})}{\sum_{f \epsilon N_i} exp(a^*_{if})} \tag{6.6}$$

where, a_{if} can be view as the influence weight of each friend with respect

to different candidate items. Finally, we can get the updated the user's social-based preference by the following function:

$$user_i^s = \sum a_{if} * u_{if} \tag{6.7}$$

By doing so, these weighted friends features can describe a user's feature more accurately than directly using the friend feature.

Fusion Layer Next, we use a fusion layer to fuse the user's different factors obtained from the previous layer into a synthetic one. We can simply generate a new vector of the user by concatenating these two latent factors to a wide one. However, it is difficult to determine which factor contributes more to the recommendation results. Therefore, we apply an attention network to assign non-uniform weights to different factors. If the social factor influences the user more on item selection, the attention network will learn a higher weight on social-based representation and penalize the item-based representation. Let $user_i^s$ denotes the social-based representation, $user_i$ denotes item-based representation, respectively. The attention score is defined as:

$$s_u^* = \sigma(W * user_i^s + b), I_u^* = \sigma(W * user_i + b) \tag{6.8}$$

where W, b are the weights and bias of the attention network, respectively. Then, the attention score of the social factor is obtained by normalizing with a softmax function:

$$\alpha_s = \frac{exp(s_u^*)}{exp(s_u^*) + exp(I_u^*)} \tag{6.9}$$

The user's final representation is defined as:

$$u_i = \alpha_s * user_i^s + (1 - \alpha_s) * user_i \tag{6.10}$$

Predict Layer After obtaining the latent factors of users and items, we predict the probability of user u clicking the item i. The prediction probability is defined as follows:

$$r_{i,j}^* = u_i * i_j \tag{6.11}$$

6.3 Optimization and Training

Unlike other methods based on user rating matrix, the task of our recommendation system is to rank items based on implicit feedback from the user

(such as clicks, favorites, etc.) to produce a personalized recommendation list. The implicit feedback of users is mainly used in BPR (Bayesian Personalized Ranking) to sort the item through the maximum posterior probability obtained by bayesian analysis, so as to generate recommendations. Therefore, we choose BPR as the objective function to optimize the model. According to the derivation, it can be defined as:

$$L_{BPR} = \sum_{(u,i,j)\epsilon D_s} -ln\sigma(\check{y}_{ui}(\Theta) - \check{y}_{uj}(\Theta)) + \lambda_\Theta ||\Theta||^2 \tag{6.12}$$

Where, D_s is the training set that we extracted data from the original data set and built, and its construction method is as follows. Based on the user history feedback data in the original data set, we will mark all users and the corresponding items. If item i is in the positive feedbacks of user u, and there is no item j, we will randomly sample item j from all the negative feedbacks as the training data, so that we will get a training triple $< u, i, j >$. The algorithm to generate the training set is as follows:

Algorithm 6 Algorithm of generating training instances

Input: rating matrix R
Output: training instances $< u, i, j >$

```
for (u, i) in R do
    i_idx.append(i)
    u_idx.append(u)
    neg_item = choice(item_list)
    while neg_item in pos_item do
        neg_item = choice(item_list)
    end while
    j_idx.append(neg_item)
end for
return u_idx,i_idx,u_idx
```

Additionally, to increase the robustness of the model, we fine-tuned the learning algorithm to improve the performance of the model on unknown data. We update the objective function by adding a regularization term $\lambda_\Theta ||\Theta||^2$. We need to optimize the value of the regularization term λ_Θ to get a well-fitting model. A good optimization algorithm can improve the running time of the algorithm. Therefore, we choose Adam optimization algorithm [72] instead of the classical stochastic gradient descent method to update the network weight more effectively. We also leverage mini-batch training strategy

to count gradient and minimize loss function. For each training epoch, a batch size of user-item pairs together with the social friends of the user are input into the HASRec model to compute the loss. To alleviate overfitting and improve the generalization ability, we apply the dropout strategy to our model. Only a part of the parameters are updated during training.

6.4 Experimental Settings

6.4.1 Datasets

The data set used in this experiment have been introduced in Chapter 5. Here, we chose three of these data sets to conduct experiment: Delicious, Ciao, and Douban. They are commonly used in the field of social recommendation. Since we have covered each data set in detail in the previous chapter, there is no further description of the data set in this chapter. It is noteworthy that the above described datasets only contain explicit feedback data, but we want to evaluate the performance against implicit feedback data. Therefore, we need to do some processing on the data. We mark the rating data of each user as 1 to indicate that the user has clicked or bought the item. However, the consequence of such processing is that each data set contains only positive samples, but no negative samples. Therefore, in the actual experiment, it is necessary to randomly generate negative samples to train the model.

6.4.2 Baselines

So as to validate the efficacy of the methods presented in this chapter, we selected the baseline algorithms commonly used in the evaluation of recommendation system for comparison. Here, we also added the proposed ANSR method described in the previous chapter for comparison. The above mentioned methods can be divided into the following groups: factor decomposition-based algorithms, social-based algorithms, and DL-based algorithms. Some of these algorithms have already been described in detail in Chapter 5, and we will not repeat them here. The specific description is as follows:

- BPR [58]: BPR is a commonly used ranking algorithms in recommendation systems.

- SBPR [20]: A social-aware recommendation algorithm which extend BPR. The basic concept of SBPR is that users tilt towards to prefer the items that their social neighbors clicked or bought.
- NeuMF [66]: A CF method based on a multi-layer perceptron. It has been proven that NeuMF is superior to the traditional recommendation algorithms.
- CUNE-BPR [27]: A network embedding(NE)-based model, which calculate the similarities between users through network embedding. It extended SBPR by assigning different regularization terms to different friends.
- SERec [73]: A a social-aware recommendation algorithm that integrate social exposure into CF to improve social-based recommendation performance.
- ANSR [61]: This is the proposed method in Chapter 5. ANSR is an attention-based deep learning model for social recommendation.

6.4.3 Evaluation protocols

In general, the data set is divided into training set and test set. There are two drawbacks to this simple division. First of all, the selection of the final model and parameters will greatly depend on how you divide the training set and the test set. Secondly, the method only uses part of the data to train the model. The test set is independent of the training and does not participate in the training at all, and is used for the evaluation of the final model. In the process of training, the problem of overfitting often occurs, that is, the model can match the training data well, but can not predict the data outside the training set well. Therefore, we use cross validation to train the model. We integrate the original data into 5 groups (5-fold), make a validation set for each subset data respectively, and take the remaining 4 sets of subset data as the training set. After the model was trained, we need to evaluate how well this model. How to evaluate a recommendation algorithm is an important link in the research of recommendation system. Accuracy metric is the most important metric in recommender system. The most commonly used accuracy metrics is the recall rates. It test how many of the user's favorite items appear in the recommended list. It could be defined like below:

$$HR@K = (Numberof Hits@K)/(|GT|) \tag{6.13}$$

In addition to predict accuracy, we also need a metric to measure the accuracy of the ranking of results. Normalized Discounted Cumulative Gain(NDCG) was used as the evaluation index of the sequencing results to evaluate the accuracy of the sequencing. It could be defined as follows:

$$NDCG@K = r_k \Sigma_{i=1}^{K} (\frac{2^{r_i} - 1}{log_2(i+1)}) \tag{6.14}$$

where, 'gain' represents the correlation score for each item in the list, which is denoted as r_i; 'cumulative gain' means the cumulative gain of K items; Considering the factor of sorting order, the item at the top of the list will gain more and the item at the bottom will lose. If the correlation score r_i has only two values of (0,1), that is, if the item in the sorting list returned by the algorithm appears in the real interaction list, the numerator is incremented by 1, otherwise it is skipped. Finally, we need to normalize the 'Discounted Cumulative Gain' indicator. r_k represents the normalized coefficient.

6.4.4 Hyper-parameter Settings

We implement our proposed model HASRec on Tensorflow, which is implemented by the Python language. The parameters used in the initial model are randomly initialized. In the experiment, the dataset was randomly split into two subsets, among this, 30 percent of the data was used as test data, and the remaining 70 percent of the data was used for cross-validation. After we prepared the data and model, we needed to choose the appropriate hyper-parameters to train the model. Once these hyper-parameters were not chosen well, the performance of the neural network is likely to be worse than that of the perceptron. Therefore, it is necessary to avoid unreasonable influence of hyper-parameters on the model. Learning rate determines the step size of weight iteration, so it is a very sensitive parameter. There must be an optimal value for the initial learning rate. If the learning rate is too large, it will not converge; if the learning rate is too small, it will converge very slowly or fail to learn. With the increase of learning rate, the model may transition from under-fitting to overfitting. As a rule, the learning rate could be tuned among [0.1,0.01,0.001]. The learning rate is rarely unchanged in the training process of the model, the way to change the learning rate is called

the self-adaptive learning rate change method. Therefore, we often chose RMSProp as the optimization method. Although the model performance is not as sensitive to batch size as the learning rate, batch size also become a very key parameter when the model performance is further improved. We could feed the whole data set or a sample to the neural network, or feed a part of the samples at a time to complete the iteration. In engineering practice, from the point of view of convergence speed, small batch sample sets are optimal, which is also known as mini-batch. Therefore, the batch size is the result of tuned to [64,128,256,512]. For the purpose of preventing the model from overfitting, we use regularization and dropout methods. Specifically, we utilized L2 regularization method to impose additional constraints on the model to restrict the range of parameter values, so as to prevent the model from becoming too complex. Therefore, the regularization parameter will be adjusted between [0.01,0.001,0.0001]. In addition, during forward propagation, we make the neurons in the network stop working with a certain probability. By convention, the drop-out rate will be adjusted between 0.4 and 0.6.

6.5 Performance Evaluation

6.5.1 Performance Comparison of Recommendation Systems

For the purpose of verifying the effectiveness of our proposed algorithm, we carried out experiments on the above mentioned data sets and compared the experiment results of HASRec to the results of the mentioned baseline methods. Table 6.1 demonstrated the experiment results of HASRec and the baseline methods on three different datasets. Several conclusions could be obtained through the observation of the results.

1) we found that based on depth of learning is always better than the effect of the methods such as NeuMF based on traditional factorization method such as BPR since compared to the deep learning based method, the method based on factorization can only learn a linear relationship between the user and item, NeuMF endows the model with more learning and representation capabilities through hierarchical network results to learn the complex nonlinear relationship between the user and the item.

2) Another finding is that social-aware recommendation systems such as SBPR, CUNE-BPR always outperform methods such as BPR and NeuMF

that use only information about the user's interaction with item. This phenomenon is easily explained, because the method of using the user history interaction with the item were easy prone to data sparsity problem, and social-aware recommending method has the advantage that it can incorporate social information into the model, leverage the characteristics of users' social neighbors to enrich their own characteristics, so as to learn better user portrait to promote the performance of the system.

3) As social-aware recommendation system, HASRec's performance is better than CUNE-BPR, SERec, and ANSR methods. This is because, compared with other algorithms, HASRec method helps users generate connections with rich semantic information through the random walk base on meta-path, through which users can find their potential social neighbors, thus alleviating the problem of sparse social information.

TABLE 6.1: Recall@K and NDCG@K comparisons for different top-k values on different datasets

Delicious	Recall@10	NDCG@10	Recall@20	NDCG@20	Recall@50	NDCG@50
BPR	0.1139	0.0758	0.1613	0.0836	0.2284	0.1033
SBPR	0.1477	0.0881	0.1733	0.1068	0.2701	0.1304
NeuMF	0.1454	0.0948	0.1658	0.1029	0.2615	0.1255
CUNE-BPR	0.1544	0.0968	0.1903	0.1037	0.2731	0.1327
SERec	0.1511	0.0997	0.2057	0.1152	0.2705	0.1342
ANSR	0.1729	0.1120	0.2287	0.1310	0.3120	0.1465
HASRec	**0.1995**	**0.1292**	**0.2526**	**0.1408**	**0.3308**	**0.1593**
Ciao	Recall@10	NDCG@10	Recall@20	NDCG@20	Recall@50	NDCG@50
BPR	2.7525%	1.3454%	4.6078%	2.3151%	7.1967%	3.3265%
SBPR	3.8335%	2.2744%	5.5479%	2.4441%	9.5698%	3.8875%
NeuMF	3.7585%	2.3541%	5.1323%	2.4026%	9.3597%	3.6882%
CUNE-BPR	4.2172%	2.3938%	5.7372%	2.4928%	10.0283%	3.8729%
SERec	4.2581%	2.4094%	5.8028%	2.4827%	10.2373%	3.9287%
ANSR	4.4172%	2.4827%	5.9479%	2.5413%	10.5698%	4.0875%
HASRec	**4.7709%**	**2.8333%**	**6.1630%**	**2.8997%**	**11.2452%**	**4.1160%**
Douban	Recall@10	NDCG@10	Recall@20	NDCG@20	Recall@50	NDCG@50
BPR	0.0381	0.0824	0.0698	0.1056	0.1366	0.1151
SBPR	0.0482	0.0973	0.0802	0.1108	0.1402	0.1192
NeuMF	0.0452	0.0927	0.0729	0.1083	0.1392	0.1184
CUNE-BPR	0.0493	0.0975	0.0928	0.1092	0.1483	0.1203
SERec	0.0502	0.0989	0.0972	0.1107	0.1458	0.1239
ANSR	0.0528	0.0992	0.1025	0.1124	0.1565	0.1283
HASRec	**0.0598**	**0.1023**	**0.1092**	**0.1173**	**0.1672**	**0.1302**

6.5.2 Model Analysis

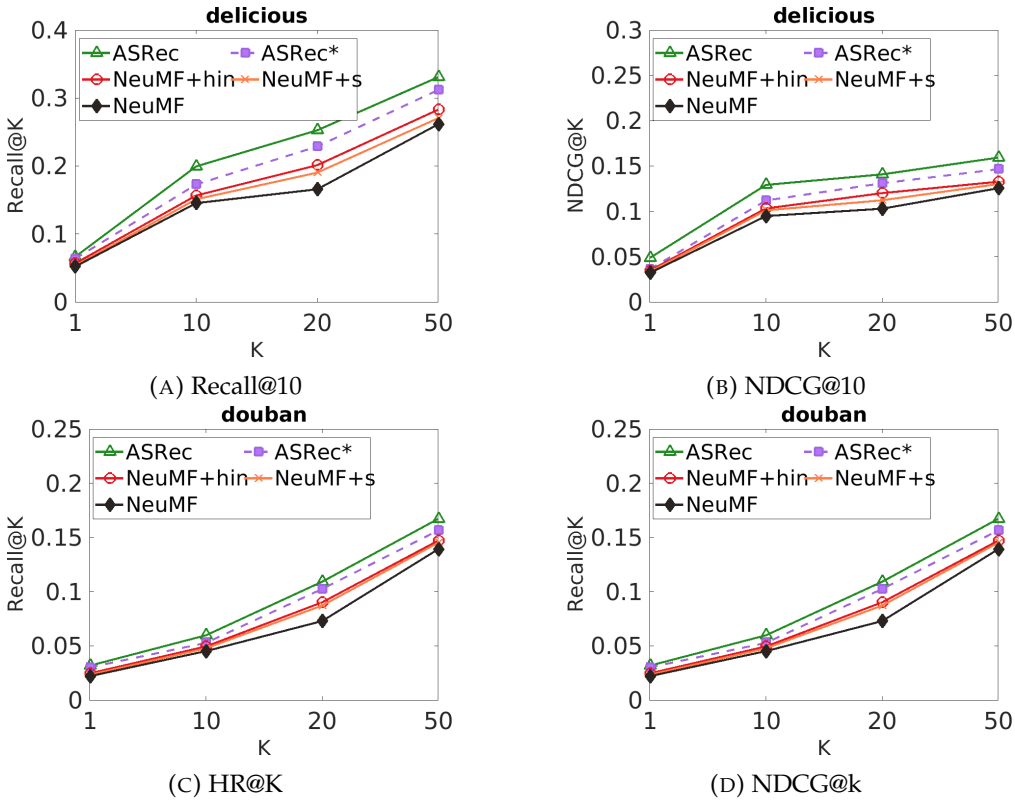
Next, we will conduct experiments to analysis how the proposed modules contribute to the final results. In the ablation studies, we will transform the proposed models into different variation models and compare the results. To clearly express different variant model, Figure 6.9 was used to demonstrate the different variant model. Firstly, we choose NeuMF method as the benchmark algorithm to compare the performance of variant models. This is because the various deformation models are based on the NeuMF model. Next, we performed ablation studies by superimposing different modules on the NeuMF model to compare the effectiveness of different modules. In order to reasonably stack each module, we took the idea from ensemble method, and divided the factors affecting user characteristics into the user's own potential preference characteristics and social preference characteristics. $NeuMF + S$ stands for incorporation of social information into the NeuMF model. The user's characteristics can be represented as the user's own potential preference characteristics and the weighted average of the user's social neighbors' feature. $NeuMF + hin$ utilized social information in the same way as $NeuMF + s$, the main difference being that it integrates the missing link identifier module into $NeuMF + s$ to address social data sparsity. $ASRec*$ is a variant model of $HASRec$ with the missing link identifier module removed and the attention module retained to evaluate the adaptive effects of different social links.

variant model	principle	user repersentation
NeuMF	basic DL-based model without any social information	u_i
NeuMF+s	utilizing social information based on assemble methods	$u_i + \sum u_{fl}, f \in \mathbb{F} = \{u_{f1}, u_{f2}, \ldots, u_{fn}\}$
NeuMF+hin	integrating the missing link identifier module to enrich social data	$u_i + \sum u_{fl}{}^*, u_{fl}{}^*$: HIN()
ASRec*	HASRec with the missing link identifier module removed	$u_i + \sum \alpha * \sum u_{fl}$, α: $Attention()$
HASRec	proposed model utilizing social information by both missing link identifier module and Attention	$u_i + \sum \alpha * u_{fl}{}^*$, $\alpha, u_{fl}{}^*$: $Attention(), HIN()$

FIGURE 6.9: The variant models in ablation study.

Figure 6.10 shows the performance of the different HASRec variant models. As it can be seen from Fig 6.10, after incorporating different modules into

the deep learning-based model, all components have made positive contributions to the improvement of the model's performance and achieved significantly improvement. As NeuMF only utilizes users' history interaction with items, which suffers the data sparsity problem, it performs worse than other variant models. NeuMF+s and NeuMF+hin perform better than NeuMF. This is because by incorporating a user's social profile into the model, we can learn a user's features from auxiliary information to solve the data sparsity problem. Compared with NeuMF+s, NeuMF+hin performs better, especially on the Douban dataset. Due to the different degrees of social data sparsity in the datasets, identifying missing links has a different impacts on improving the performance of recommendation models. The more sparse the dataset is, the more improvement the model can achieve. HASRec and ASRec* further improve the performance on two datasets because they include the attention-based module to evaluate the quality of social links. Unlike the methods that view all social relations as the same influence to the users, the variant models assign non-uniform weights to the social links. If the social links are not helpful to a user's recommendation results, the module will punish these links and increase the weights of more useful social links. Compared with ASRec*, the HASRec model add missing links identifier module enable it to augment the user's social data to further improve the performance of recommendation models.



(A) Recall@10 (B) NDCG@10

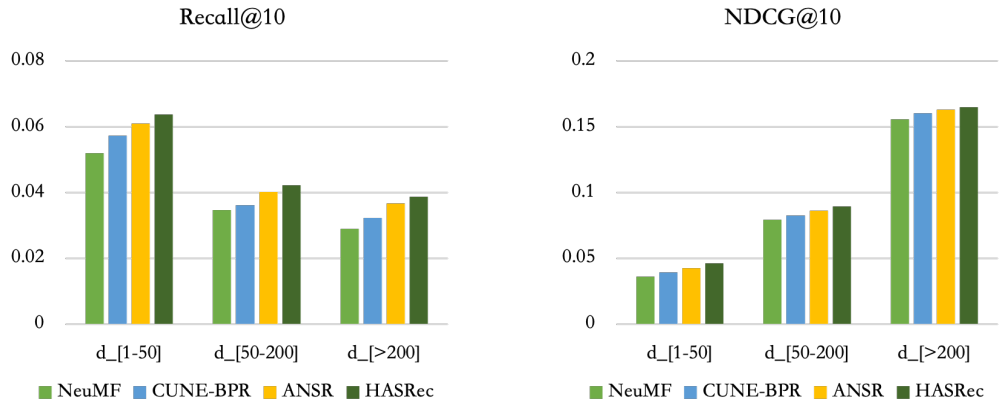
(C) HR@K (D) NDCG@k

FIGURE 6.10: (a) and (b) are Recall@10 and NDCG@10 of the variant models of HASRec on the delicious dataset, respectively. (c) and (d) are Recall@10 and NDCG@10 of the variant models of HASRec on the douban, respectively.

6.5.3 Case Study

For the purpose of studying whether the algorithm we proposed can effectively alleviate the problem of data sparsity, we will conduct comparative experiments on data sets with different data sparsity degrees. Due to the limited data, we divide the same data set into data sets with different data sparsity. Specifically, we divided the *Douban* dataset into three groups according to the number of scoring records. Where, D_50 means that the user has less than 50 rating records, D_50_100 means that the user's rating records are between 50-200, and D _100 means that the user has more than 200 rating records. Next, we choose several representative algorithms to compare with HASRec to study the performance of different models at different data sparsity levels. Figure 6.11 shows the performance comparison results of different algorithms on three data sets which have different data sparsity. It can

be found that when the dataset was sparse, HASRec improved significantly compared to the other baselines. HASRec's performance gains are less pronounced when the data density is higher. This is because, when the data set is very sparse, we need to make full use of social data as auxiliary information to enrich the user portrait, so as to learn richer user characteristics. When the data becomes dense, and the data set itself contains a lot of information, we don't necessarily need additional information. Of course, in practice, it is difficult to get such dense data distribution, so HASRec is used to solve the problem of data sparsity.



(A) HR@10 (B) NDCG@10

FIGURE 6.11: (a) and (b) are Recall@10 and NDCG@10 of NeuMF, CUNE-BPR, ANSR and HASRec on the douban dataset w.r.t. different data sparsity levels.

We have mentioned in the discussion that attention neural network can effectively improve the explanatory ability of models. Next, we will demonstrate why HASRec can provide explanatory recommendation results through a case study. For the purpose of reasonably exploiting users' social data, the factors affecting user characteristics were divide into the user's own potential preference characteristics and social preference characteristics. HASRec can learn two attention scores for these two factors through the Fusion Layer. For example, we randomly selected $item(\#387)$ and $item(\#929)$ from the top10 list of $user(\#1408)$. Through the attention network, different factors are assigned different weights, known as attention scores. In this way we can see which factor has the greatest impact on the user. For example, $item(\#387)$ is 0.38 and 0.62, respectively; This means that social factors have a greater influence on the recommendation results. By doing so, models provide some explanatory

power to a certain extent. In addition to knowing which factors influence recommendations, we can also understand the distribution of influence of different friends on users. Figure 6.12 shows the attention distribution of the top 10 friends of *user*(#1408) interact with *item*(#387) and *item*(#929). From Figure 6.12, we can observe that different friends have different attention weights and contribute differently. A number of friends have a higher attention weight, which means they influence the target user more than other friends. This is easy to understand, because the diversity of the network leads to different social relationships contributing differently to users, and therefore, these weight of attention will change accordingly.

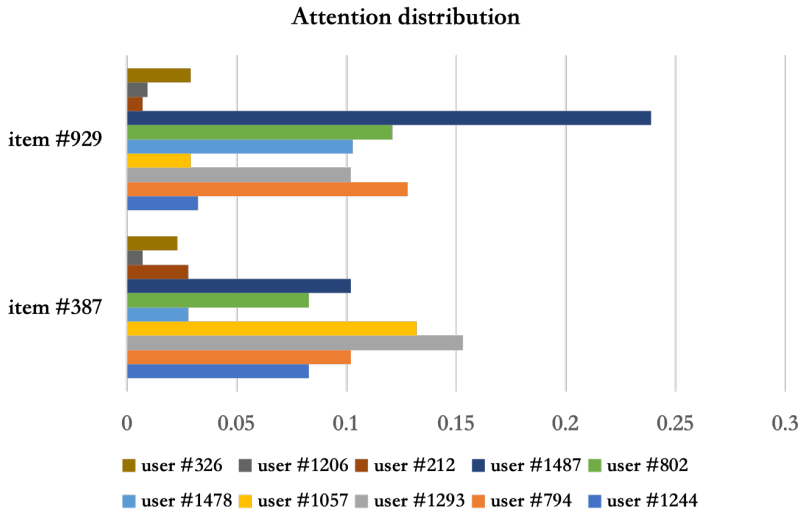


FIGURE 6.12: Case Study: The attention distribution of sampled friends of *user*(1408).

Chapter 7

Conclusions and Future Work

In this paper, we discussed how to incorporate users' social information into recommendation models and generate a more quality recommendation results. As we have discussed in this paper, traditional recommendation algorithms mainly fall ill with data sparsity issue. Several studies have demonstrated that leveraging social information to make recommendation could over the above shortcomings. So, our research focused on the social recommendation. Despite the success of the existed works, we have found several limitations. For example, the embedding method used in the existed work are not sufficient to yield satisfactory embeddings for users. The user's different social relations will play a different role to recommendation results. The user's social data are still facing the data sparsity problem. And there are still other limitations need to solve.

For the purpose of solving the above limitations, we proposed to design a representation learning framework to train a better model for different user and improve the recommendation quality. To be more specific, we propose a social semantic encoding module in the embedding layer to solve the encoding problem. An aspect-aware Module and influence calculating module is proposed in the neural network layer to solve the problem of noise data and social influence. Last, an Identify missing link module is proposed to solve the social data sparsity problem. There are two main contribution in this paper. The first contribution is that we design a model called 'An Adaptive Attentive Model for Social Recommendation'. In this model, we improve the method used in embedding layer to learn a better embedding for user which contain the social semantic signals. Besides, we design an aspect-aware module and an influence calculating module could solve noise data problem and assign adaptive influence on different friends. Another contribution is that

we proposed another work called 'Enhance Social Recommendation via Heterogeneous information Networks'. In this work, we design an identify missing link module to find the missing link of users and solve users' social data sparsity problem. Also, the fusion layer provide a way to improve the interpretability of the recommendation results.

These methods still have some limitations. First, this paper focuses on the user modeling. However, the use-item rating network is not well studied. In fact, interactions in the user-item graph should also be jointly captured to learn item latent features. In the future work, we will further incorporate the item modeling into the existed framework to learn a better representation for the item. We would also like to incorporate more side information, like reviews and knowledge graphs, to the recommendation model to learn more informative features of users and items. Moreover, we found that the temporal signals of users are not well studied. For example, a person's interests shift over time. A person's interests after work may be quite different from those of his school days. In this way, the accuracy of a recommendation system based on historical feedback will be influenced by past preferences. Therefore, establishing a recommendation system based on timing information need to be considered. It is also worth noting that users' social information is not easily accessible. Some online platform doesn't have a clear social network. We may need to continue to explore the user's history to build similar communities. These studies may brings new opportunities for social recommender system. Besides, the evaluation of the generated friends is not well studied. So far, we utilize the manual design to generate social data. How is the quality of the generated social data? Inspired by the GAN, which can automatically complete the process of data generation, and constantly optimize to judge the quality of the generated results.

Nowadays, the development of deep learning technology provides a new research direction for social recommendation. Other studies have attempted to integrate other deep learning methods into collaborative filter-based models. For example, multi-task learning provides a new research direction for the development of recommendation system. Most algorithms currently focus on a single model, which can cause the model to ignore potential information in related tasks that might improve the target task. By sharing parameters between different tasks to a certain extent, the original task will have a better generalization ability. For example, [74] proposed a method based on multi-task learning that adaptively assign a personalized scheme

to transfer the shared knowledge between item domain and social domain. Adversarial Learning is similar to a regularization method, which can improve the quality of embedding and avoid overfitting to achieve more excellent performance. [75] adopted a bidirectional mapping method to transfer users' information between social domain and item domain using adversarial learning. These latest researches provide important reference significance for our future work.

At present, the recommender system is mainly used in PC and mobile, which accounts for the vast majority of recommender system products. In the future, with the development of intelligence, intelligent devices will appear in more scenes, and the application in these scenes may also need to use recommendation technology to distribute information. Recommendation system into daily life to make life more intelligent. For example, it is also feasible to make intelligent recommendations on smart refrigerators. The smart refrigerator can record the consumption of food in household life and understand the family's eating habits. Based on the mining of family consumption habits, it is a very feasible strategy to carry out accurate personalized food recommendation, which is directly linked to e-commerce. It is of great commercial value and is worth looking forward to and exploring. With the powerful generative ability of GAN, everyone can become an artist in the future. If someone has the ability to create a story but can't draw, GAN can help generate the corresponding cartoon based on their idea. Not only that, everyone can have the ability to compose and write poetry, making life full of creativity. Another future for AI is building smart cities. With the development of the city, the intelligent technology of the city is constantly evolving to build an intelligent system based on serving the residents in the future. For example, intelligent travel, based on historical information such as traffic flow, it can intelligently control the traffic light time to make travel more convenient. In a word, AI is the future.

Reference

[1] J Ben Schafer et al. “Collaborative filtering recommender systems”. In: *The adaptive web*. Springer, 2007, pp. 291–324.

[2] Yehuda Koren, Robert Bell, and Chris Volinsky. “Matrix factorization techniques for recommender systems”. In: *Computer* 42.8 (2009), pp. 30–37.

[3] Hao Ma, Irwin King, and Michael R Lyu. “Learning to recommend with explicit and implicit social relations”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 2.3 (2011), pp. 1–19.

[4] Gediminas Adomavicius and Alexander Tuzhilin. “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions”. In: *IEEE transactions on knowledge and data engineering* 17.6 (2005), pp. 734–749.

[5] Miller McPherson, Lynn Smith-Lovin, and James M Cook. “Birds of a feather: Homophily in social networks”. In: *Annual review of sociology* 27.1 (2001), pp. 415–444.

[6] Fuzheng Zhang et al. “Collaborative knowledge base embedding for recommender systems”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 353–362.

[7] Junliang Yu et al. “Hybrid attacks on model-based social recommender systems”. In: *Physica A: Statistical Mechanics and its Applications* 483 (2017), pp. 171–181.

[8] Xin Wang et al. “Social Recommendation with Strong and Weak Ties”. In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. CIKM ’16. Indianapolis, Indiana, USA: Association for Computing Machinery, 2016, pp. 5–14. ISBN: 9781450340731. DOI: 10.1145/2983323.2983701. URL: https://doi.org/10.1145/2983323.2983701.

[9] Jiliang Tang, Huiji Gao, and Huan Liu. "mTrust: discerning multi-faceted trust in a connected world". In: *Proceedings of the fifth ACM international conference on Web search and data mining*. 2012, pp. 93–102.

[10] Jiliang Tang, Xia Hu, and Huan Liu. "Social recommendation: a review". In: *Social Network Analysis and Mining* 3.4 (2013), pp. 1113–1133.

[11] Mohsen Jamali and Martin Ester. "A matrix factorization technique with trust propagation for recommendation in social networks". In: *Proceedings of the fourth ACM conference on Recommender systems*. 2010, pp. 135–142.

[12] Yehuda Koren. "Factorization meets the neighborhood: a multifaceted collaborative filtering model". In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2008, pp. 426–434.

[13] Yifan Hu, Yehuda Koren, and Chris Volinsky. "Collaborative filtering for implicit feedback datasets". In: *2008 Eighth IEEE International Conference on Data Mining*. Ieee. 2008, pp. 263–272.

[14] Mao Ye, Xingjie Liu, and Wang-Chien Lee. "Exploring social influence for recommendation: a generative model approach". In: *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. 2012, pp. 671–680.

[15] Mohsen Jamali and Martin Ester. "Trustwalker: a random walk model for combining trust-based and item-based recommendation". In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, pp. 397–406.

[16] Ruslan Salakhutdinov and Andriy Mnih. "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 880–887.

[17] Hao Ma, Irwin King, and Michael R Lyu. "Learning to recommend with social trust ensemble". In: *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. 2009, pp. 203–210.

[18] Hao Ma et al. "Sorec: social recommendation using probabilistic matrix factorization". In: *Proceedings of the 17th ACM conference on Information and knowledge management*. 2008, pp. 931–940.

[19] Shuang-Hong Yang et al. "Like like alike: joint friendship and interest propagation in social networks". In: *Proceedings of the 20th international conference on World wide web*. 2011, pp. 537–546.

[20] Tong Zhao, Julian McAuley, and Irwin King. "Leveraging social connections to improve personalized ranking for collaborative filtering". In: *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*. 2014, pp. 261–270.

[21] Hao Wang, Xingjian Shi, and Dit-Yan Yeung. "Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks". In: *Advances in Neural Information Processing Systems* 29 (2016), pp. 415–423.

[22] Bo Yang et al. "Social collaborative filtering by trust". In: *IEEE transactions on pattern analysis and machine intelligence* 39.8 (2016), pp. 1633–1647.

[23] Hao Ma et al. "Recommender systems with social regularization". In: *Proceedings of the fourth ACM international conference on Web search and data mining*. 2011, pp. 287–296.

[24] Xin Wang et al. "Social recommendation with strong and weak ties". In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 2016, pp. 5–14.

[25] Soude Fazeli et al. "Implicit vs. Explicit Trust in Social Matrix Factorization". In: *Proceedings of the 8th ACM Conference on Recommender Systems*. RecSys '14. Foster City, Silicon Valley, California, USA: Association for Computing Machinery, 2014, pp. 317–320. ISBN: 9781450326681. DOI: 10.1145/2645710.2645766. URL: https://doi.org/10.1145/2645710.2645766.

[26] Mao Ye et al. "Exploiting geographical influence for collaborative point-of-interest recommendation". In: *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 2011, pp. 325–334.

[27] Chuxu Zhang et al. "Collaborative User Network Embedding for Social Recommender Systems". In: *SDM* (2017), pp. 381–389.

[28] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. "Restricted Boltzmann machines for collaborative filtering". In: (2007), pp. 791–798.

[29] Jian Wei et al. "Collaborative filtering and deep learning based hybrid recommendation for cold start problem". In: *2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*. IEEE. 2016, pp. 874–877.

[30] Mingyang Sun et al. "Using Bayesian deep learning to capture uncertainty for residential net load forecasting". In: *IEEE Transactions on Power Systems* 35.1 (2019), pp. 188–201.

[31] Bartłomiej Twardowski. "Modelling contextual information in session-aware recommender systems with neural networks". In: *Proceedings of the 10th ACM Conference on Recommender Systems*. 2016, pp. 273–276.

[32] Yong Kiam Tan, Xinxing Xu, and Yong Liu. "Improved recurrent neural networks for session-based recommendations". In: *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. 2016, pp. 17–22.

[33] Ying Shan et al. "Deep crossing: Web-scale modeling without manually crafted combinatorial features". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 255–262.

[34] Sungyong Seo et al. "Interpretable convolutional neural networks with dual local and global attention for review rating prediction". In: *Proceedings of the Eleventh ACM Conference on Recommender Systems*. 2017, pp. 297–305.

[35] Hanh TH Nguyen et al. "Personalized deep learning for tag recommendation". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2017, pp. 186–197.

[36] Steffen Rendle and Lars Schmidt-Thieme. "Online-updating regularized kernel matrix factorization models for large-scale recommender systems". In: *Proceedings of the 2008 ACM conference on Recommender systems*. 2008, pp. 251–258.

[37] Arkadiusz Paterek. "Improving regularized singular value decomposition for collaborative filtering". In: *Proceedings of KDD cup and workshop*. Vol. 2007. 2007, pp. 5–8.

[38] Lei Zheng, Vahid Noroozi, and Philip S Yu. "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*. 2017, pp. 425–434.

[39] Xiangnan He et al. "Neural collaborative filtering". In: *Proceedings of the 26th international conference on world wide web*. 2017, pp. 173–182.

[40] Yao Wu et al. "Collaborative denoising auto-encoders for top-n recommender systems". In: *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. 2016, pp. 153–162.

[41] Suhang Wang et al. "What Your Images Reveal: Exploiting Visual Contents for Point-of-Interest Recommendation". In: *Proceedings of the 26th International Conference on World Wide Web*. WWW '17. Perth, Australia: International World Wide Web Conferences Steering Committee, 2017, pp. 391–400. ISBN: 9781450349130. DOI: 10.1145/3038912.3052638. URL: https://doi.org/10.1145/3038912.3052638.

[42] Donghyun Kim et al. "Convolutional matrix factorization for document context-aware recommendation". In: *Proceedings of the 10th ACM conference on recommender systems*. 2016, pp. 233–240.

[43] Balázs Hidasi et al. "Session-based recommendations with recurrent neural networks". In: *arXiv preprint arXiv:1511.06939* (2015).

[44] Robin Devooght and Hugues Bersini. "Collaborative filtering with recurrent neural networks". In: *arXiv preprint arXiv:1608.07400* (2016).

[45] Jan K Chorowski et al. "Attention-based models for speech recognition". In: *Advances in neural information processing systems*. 2015, pp. 577–585.

[46] Jingyuan Chen et al. "Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention". In: *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. 2017, pp. 335–344.

[47] Antonia Creswell et al. "Generative adversarial networks: An overview". In: *IEEE Signal Processing Magazine* 35.1 (2018), pp. 53–65.

[48] Weizhi Nie et al. "HGAN: Holistic Generative Adversarial Networks for Two-dimensional Image-based Three-dimensional Object Retrieval". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 15.4 (2019), pp. 1–24.

[49] Jun Wang et al. "Irgan: A minimax game for unifying generative and discriminative information retrieval models". In: *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. 2017, pp. 515–524.

[50] Dong-Kyu Chae et al. "Cfgan: A generic collaborative filtering framework based on generative adversarial networks". In: *Proceedings of the 27th ACM international conference on information and knowledge management*. 2018, pp. 137–146.

[51] Shuicheng Yan et al. "Graph embedding and extensions: A general framework for dimensionality reduction". In: *IEEE transactions on pattern analysis and machine intelligence* 29.1 (2006), pp. 40–51.

[52] Xin Li and Hsinchun Chen. "Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach". In: *Decision Support Systems* 54.2 (2013), pp. 880–890.

[53] Yehuda Koren and Robert Bell. "Advances in collaborative filtering". In: *Recommender systems handbook*. Springer, 2015, pp. 77–118.

[54] Li Yu. "Using ontology to enhance collaborative recommendation based on community". In: *2008 The Ninth International Conference on Web-Age Information Management*. IEEE. 2008, pp. 45–49.

[55] Artus Krohn-Grimberghe et al. "Multi-relational matrix factorization using bayesian personalized ranking for social network data". In: *Proceedings of the fifth ACM international conference on Web search and data mining*. 2012, pp. 173–182.

[56] Rodolphe Jenatton et al. "A latent factor model for highly multi-relational data". In: *Advances in neural information processing systems* 25 (2012), pp. 3167–3175.

[57] Léon Bottou. "Stochastic gradient descent tricks". In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 421–436.

[58] Steffen Rendle et al. "BPR: Bayesian personalized ranking from implicit feedback". In: *arXiv preprint arXiv:1205.2618* (2012).

[59] Ian Goodfellow et al. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016.

[60] Xiaofeng Meng et al. *Database systems for advanced applications*. Springer, 2010.

[61] Munan Li, Kenji Tei, and Yoshiaki Fukazawa. "An Efficient Adaptive Attention Neural Network for Social Recommendation". In: *IEEE Access* 8 (2020), pp. 63595–63606.

[62] Marcel Van Gerven and Sander Bohte. "Artificial neural networks as models of neural information processing". In: *Frontiers in Computational Neuroscience* 11 (2017), p. 114.

[63] Xiangnan He et al. "Trirank: Review-aware explainable recommendation by modeling aspects". In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 2015, pp. 1661–1670.

[64] Wenbo Wang et al. "Using Visualization to Improve Clustering Analysis on Heterogeneous Information Network". In: *2018 22nd International Conference Information Visualisation (IV)*. IEEE. 2018, pp. 220–227.

[65] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. "2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011)". In: *Proceedings of the 5th ACM conference on Recommender systems*. RecSys 2011. Chicago, IL, USA: ACM, 2011.

[66] Hong-Jian Xue et al. "Deep Matrix Factorization Models for Recommender Systems." In: *IJCAI*. Vol. 17. Melbourne, Australia. 2017, pp. 3203–3209.

[67] L. Chuanzhen et al. "Social Recommender Systems using Collaborative User Network Embedding with Bias". In: *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*. 2019, pp. 230–234. DOI: 10.1109/ICCC47050.2019.9064484.

[68] Xiangnan He et al. "Adversarial Personalized Ranking for Recommendation". In: SIGIR '18. Ann Arbor, MI, USA: Association for Computing Machinery, 2018, pp. 355–364. ISBN: 9781450356572. DOI: 10.1145/3209978.3209981. URL: https://doi.org/10.1145/3209978.3209981.

[69] Daixin Wang, Peng Cui, and Wenwu Zhu. "Structural Deep Network Embedding". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 1225–1234. ISBN: 9781450342322. DOI: 10.1145/2939672.2939753. URL: https://doi.org/10.1145/2939672.2939753.

[70] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. "metapath2vec: Scalable representation learning for heterogeneous networks". In: *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 2017, pp. 135–144.

[71] Xiao Yu et al. "Collaborative filtering with entity similarity regularization in heterogeneous information networks". In: *IJCAI HINA* 27 (2013).

[72] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[73] Menghan Wang et al. "Collaborative filtering with social exposure: A modular approach to social recommendation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[74] Chong Chen et al. "An efficient adaptive transfer neural network for social-aware recommendation". In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2019, pp. 225–234.

[75] Wenqi Fan et al. "Deep adversarial social recommendation". In: *arXiv preprint arXiv:1905.13160* (2019).

Research Contributions

- Journal

1. Li, Munan, Kenji Tei, and Yoshiaki Fukazawa. "An Efficient Adaptive Attention Neural Network for Social Recommendation." *IEEE Access* 8 (2020): 63595-63606. DOI:10.1109/ACCESS.2020.2984340

- Conference

1. Li, Munan, Kenji Tei, and Yoshiaki Fukazawa. “An efficient co-attention neural network for social recommendation.” In *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)* (pp. 34-42). IEEE.
2. Li, Munan, Kenji Tei, and Yoshiaki Fukazawa. “Heterogeneous Information Network based Adaptive Social Influence Learning for recommendation and explanation.” (accepted by The 2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology)

◊ Other Papers

1. Wu Weijiang, Li Munan and Li Guohe. “Parallel Processing of the Louvain Algorithm.” Computer & Digital Engineering. DOI: 10.3969/j.issn.1672-9722.2016.08.002