UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Engineering and Information Technology

Graph Representation Learning-Based Recommender Systems

by

Lei Sang

Thesis for
Doctor of Philosophy

Sydney, Australia

2020

Certificate of Original Authorship

I, Lei Sang declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Electrical and Data Engineering, Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. This document has not been submitted for qualifications at any other academic institution.

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

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

Production Note:
Signature: Signature removed prior to publication.

Date: 20/09/2020

ABSTRACT

Graph Representation Learning-Based Recommender Systems

by
Lei Sang

Personalized recommendation has been applied to many online services such as E-commerce and adverting. It facilitates users to discover a small set of relevant items, which meet their personalized interests, from many choices. Nowadays, various auxiliary information on users and items become increasingly available in online platforms, such as user demographics, social relations, and item knowledge. More recent evidences suggests that incorporating such auxiliary data with collaborative filtering can better capture the underlying and complex user-item relationships, and further achieve higher recommendation quality.

In this thesis, we focus on auxiliary data with graph structure, such as social networks and knowledge graphs(KG). For example, we can improve recommendation performance by mining social relationships between users, and also by using knowledge graphs to enhance the semantics of recommended items. Network representation learning aims to represent each vertex in a network (graph) as a low-dimensional vector while still preserving its structural information. Due to the availability of massive graph data in recommender systems, it is a promising approach to combine network representation learning with recommendation. Applying the learned graph features to recommender systems will effectively enhance the learning ability of the recommender systems and improve the accuracy and user satisfaction of the recommender systems. For network representation learning and its application in recommendation systems, the major contributions of this thesis are as follows:

(1) Attention-based Adversarial Autoencoder for Multi-scale Network Embedding. Existing Network representation methods usually adopt a one-size-fits-all approach when concerning multi-scale structure information, such as first- and second-order proximity of nodes, ignoring the fact that different scales play different roles in embedding learning. We propose an Attention-based Adversarial Autoencoder Network Embedding (AAANE) framework, which promotes the collaboration of different scales and lets them vote for robust representations.

(2) Multi-modal Multi-view Bayesian Semantic Embedding for Community Question Answering:Semantic embedding has demonstrated its value in latent representation learning of data, and can be effectively adopted for many applications. However, it is difficult to propose a joint learning framework for semantic embedding in Community Question Answer (CQA), because CQA data have multi-view and sparse properties. In this thesis, we propose a generic Multi-modal Multi-view Semantic Embedding (MMSE) framework via a Bayesian model for question answering.

(3) Context-Dependent Propagating-based Video Recommendation in Multi-

modal Heterogeneous Information Networks. Conventional approaches to video recommendation primarily focus on exploiting content features or simple user-video interactions to model the users' preferences. However these methods fail to model the complex video context interdependency, which is obscure/hidden in heterogeneous auxiliary data. In this paper, we propose a Context-Dependent Propagating Recommendation network (CDPRec) to obtain accurate video embedding and capture global context cues among videos in HINs. The CDPRec can iteratively propagate the contexts of a video along links in a graph-structured HIN and explore multiple types of dependencies among the surrounding video nodes.

(4) Knowledge Graph Enhanced Neural Collaborative Filtering. Existing neural collaborative filtering (NCF) recommendation methods suffer from severe sparsity problem. Knowledge Graph (KG), which commonly consists of fruitful connected facts about items, presents an unprecedented opportunity to alleviate the sparsity problem. However, NCF only methods can hardly model the high-order connectivity in KG, and ignores complex pairwise correlations between user/item embedding dimensions. To address these issues, we propose a novel Knowledge graph enhanced Neural Collaborative Recommendation (K-NCR) framework, which effectively combines user-item interaction information and auxiliary knowledge information for recommendation.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Min Xu for her selfless devotion to supporting and cultivating me in past three and a half years. She offered me a unique opportunity to study at UTS, where I was able to learn from so many talented colleagues. From the beginning of the scholarship application to the final thesis submission, Dr. Xu constantly gave me great support. Without her professional guidance and support, this work would not have been achieved.

I would also like to express my thanks to Dr. Shengsheng Qian for his generous guidance, encouragement and collaboration. He provided valuable suggestions to my research, including thinking and writing. I would like to pass my gratitude to my colleagues and friends in UTS: Tianrong Rao, Haimin Zhang, Lingxiang Wu, Zhongqin Wang, Zhiyuan Shi, Yukun Yang, Wanneng Wu, Ruiheng Zhang, Xiaoxu Li, Zhu Lin, Peng Zhang, Tao Shen etc. Thanks the UTS President's Scholarship and China Scholarship Council (CSC) Scholarship for funding my research.

Lastly, I want to show my special thanks to my family: my parents, my brother and my sister. Because of them, my every effort became meaningful. I especially thank my parents for their self-sacrifice and endless love. I am thankful to my beloved wife, Fengnan Ma, and my baby son, Xiayi Sang, for their love and encouragement which helps me finish this thesis.

List of Publications

Published Papers

1. **Lei Sang**, Min Xu, Shengsheng Qian, Xindong Wu. AAANE: Attention-based Adversarial Autoencoder for Multi-scale Network Embedding. PAKDD 2019.

2. **Lei Sang**, Min Xu, Shengsheng Qian, Xindong Wu. Multi-modal multi-view Bayesian semantic embedding for community question answering. Neurocomputing 334, 44-58, 2019.

3. **Lei Sang**, Min Xu, Shengsheng Qian, Xindong Wu. Context-Dependent Propagating based Video Recommendation in Multimodal Heterogeneous Networks. IEEE Transactions on Multimedia, 2020. DOI:10.1109/TMM.2020.3007330

4. **Lei Sang**, Min Xu, Shengsheng Qian, Xindong Wu. Knowledge Graph Enhanced Neural Collaborative Recommendation, Expert Systems with Applications, 2020. DOI: 10.1016/j.eswa.2020.113992

5. Lingxiang Wu, Min Xu, **Lei Sang**, Ting Yao, Tao Mei. Noise Augmented Double-stream Graph Convolutional Networks for Image Captioning, IEEE Transactions on Circuits and Systems for Video Technology. 2020

Others

1. **Lei Sang**, Min Xu, Shengsheng Qian, Xindong Wu. Knowledge Graph Enhanced Neural Collaborative Filtering with Residual Recurrent Network. submitted to Neurocomputing (Under Major Revision)

2. **Lei Sang**, Min Xu, Shengsheng Qian, Xindong Wu. Adversarial Optimisation for Heterogeneous Graph Neural Network based Recommendation, under submission to IEEE Transactions on Systems, Man, and Cybernetics: Systems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

As the amount of data from different platforms grows at an unprecedented rate, recommendation is becoming more and more crucial to help users discover personalized content from this ever-growing corpus of data [2, 3]. For example, 500 hours of videos are uploaded to YouTube every minute [4]. It is impossible for users to watch all available videos to identify their interested ones. As a result, there is an enormous demand for recommendation to provide relevant information tailored to users' interests or preferences. Recommendation problem can often be seen technically as a matching problem. Its purpose is to estimate the relevance score of a user to a new item based on historical user interactions with the item, or the characteristics of the user and the item. User-item interactions usually can be explicit feedback(such as, comments and ratings), and also implicit feedback(such as views and clicks).

There are three main methods to recommending system: the Collaborative Filtering (CF) approach, the content-based approach, and the hybrid approach. Among them, collaborative filtering is the most widely used, due to its ease of implementation and better performance in modelling user-item interactions. CF explores users' preferences for items by simply assuming that similar users have similar preferences for certain items. CF uses only the user and item IDs in matching, ignoring other auxiliary information. Matrix Factorization (MF) is the simplest and most effective collaborative filtering model, which describes a user ID or an item ID in terms of a potential low-dimensional vector and models the user-item interaction as the inner product of its potential vector. However, we note that the expressiveness of MF is greatly limited due to the use of only ID information to model user-item interactions. In particular, when a system have enormous amount of items and users, such as the millions of items on Amazon and Taobao, it is easy to lead to sparse user-item interactions. The sparsity problem greatly hinders the performance of collaborative filtering methods. In addition, collaborative methods often suffer from cold start problems, which means that they cannot recommend new items (cold start items) or new users (cold start users) because of the lack of interactions to learn their latent vectors.

Nowadays, diverse kinds of auxiliary information on users or items become increasingly available in online platforms, such as user demographics, social relations, and item knowledge, which bring new challenges as well as opportunities to the recommendation process. More recent evidence suggests that incorporating such auxiliary data with collaborative filtering can better capture the underlying and complex user-item relationships, and further achieve higher recommendation qual-

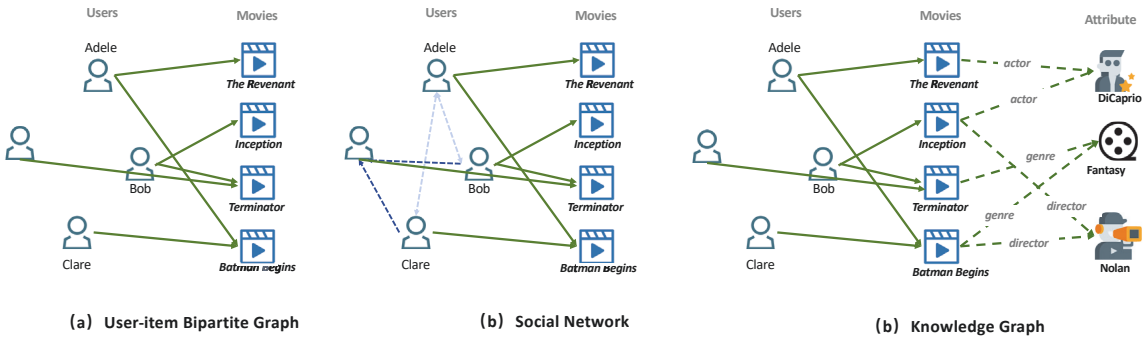


Figure 1.1 : Graph structure data used in recommendation.

ity. Furthermore, auxiliary data, such as social relations and background knowledge on items, enable us to uncover valuable evidence as well as reasons why a recommendation is made.

In this research, we focus on a special type of side information, i.e., graph-structured information. For example, there may be an online social network among users and a knowledge graph (KG) among items. And the user-item interaction can also be treated as an interaction graph. Given the rich network-structured information, it is a great challenge how to effectively utilise the high-dimensional data for recommendation. We show the three type of commonly used graph structure data in Figure 1.1.

(1) user-item bipartite graph: The user-item interaction is a bipartite graph structure, because user nodes can only connect to item nodes and vice versa. This bipartite graph can show the subtle relationship between indirectly connected users and items. According to this graph-based representation of interaction, the process of recommendation is equivalent to a process of link prediction of unseen edges based on the existing edges that connect users and items.

(2) Social Network: As members of a society, people like to seek recommendations from friends or people they know in social network. In fact, social scientists and some empirical studies have proved that social network users are correlated, that is, their preferences are similar to or influenced by those of their social network friends. Therefore, by combining social networks among users, social recommendation systems are created.

(3)Knowledge Graph: In addition to social networks between users and users, items can also form knowledge graph. A knowledge graph is a heterogeneous graph in which nodes are entities and edges represent relationships between entities. Recommended items and their properties can be mapped to the knowledge graph to obtain the interrelationships between recommended items. In addition, users can also be integrated into the knowledge graph, which allows the relationship between users and items and user preferences to be captured more accurately.

Compared with other features, graph structure are more complex and the raw dimension of the features is higher. How to learn effective features from graph

structures and apply these graph features to the recommendation algorithm is the key issue to improve the recommendation efficiency.

In practice, we cannot directly use the raw representation of graph data to assist the recommendation system. Thus, we need to learn the representation of graph structure data. The most traditional way to represent network data is to use the adjacency matrix. However, an adjacency matrix represents each node with the same dimension as the number of nodes. This representation can be very noisy and high dimensional data is difficult to handle, So we need to find a low-dimensional representation of the data so that the node representation dimension is much lower than the number of nodes. In recent years, graph representation learning (also known as network embedding) has attracted increasing attention, because it attempts to represent each node in a network as a low-dimensional representation vector, while still keeping as much information about its original network structure as possible. This low-dimensional vector representation is more easily applied to a variety of machine learning methods. In addition, the study of graph feature learning has been extended to various types of networks, such as weighted graphs, directed graphs, and heterogeneous graphs.

Various kinds of auxiliary data that are increasingly appearing on web platforms, which can be used to improve the data sparsity problem of recommendation. In this research, we focus on two types of graph structure of ancillary data, social networks and knowledge graphs. For example, by considering the social relationships between users to improve recommendation performance. There is also the use of knowledge graphs to enhance the semantics of recommended items. We can use graph feature learning methods to process the relevant features in the recommender system, and then apply the learned features to the recommender system effectively Since such a large number of graph structures are naturally present in the recommendation problem, combining graph learning with the recommender system will enhance the learning ability of the recommender system and improve its accuracy and user satisfaction.

1.2 Motivation & Contribution

In this research, we investigate graph-based recommender systems. The major contributions of this thesis are as follows:

First, for network representation learning, existing methods usually adopt a "one-size-fits-all" approach when concerning multi-scale structure information, such as first- and second-order proximity of nodes, ignoring the fact that different scales play different roles in the embedding learning. In this work, we propose an Attention-based Adversarial Autoencoder Network Embedding(AAANE) framework, which promotes the collaboration of different scales and lets them vote for robust representations. The proposed AAANE consists of two components: 1) Attention-based autoencoder effectively capture the highly non-linear network structure, which can deemphasize irrelevant scales during training. 2) An adversarial regularization guides the autoencoder learn robust representations by matching the posterior distribution of the latent embeddings to given prior distribution. This is the first attempt

to introduce attention mechanisms to multi-scale network embedding. Experimental results on realworld networks show that our learned attention parameters are different for every network and the proposed approach outperforms existing state-ofthe-art approaches for network embedding.

Second, we study social network-enhanced recommender systems. We apply homogeneous and heterogeneous social network embedding to community Question Answer (CQA) recommendation and video recommendation, respectively.

(1) *Bayesian semantic embedding for community question recommendation*: Semantic embedding has demonstrated its value in latent representation learning of data, and can be effectively adopted for many applications. However, it is difficult to propose a joint learning framework for semantic embedding in Community Question Answer(CQA), because CQA data have multi-view and sparse properties. In this work, we propose a generic Multi-modal Multi-view Semantic Embedding (MMSE) framework via a Bayesian model for question answering. Compared with existing semantic learning methods, the proposed model mainly has two advantages: 1)To deal with the multi-view property, we utilize the Gaussian topic model to learn semantic embedding from both local view and global view. 2) To deal with the sparse property of question answer pairs in CQA, social structure information is incorporated. Social information can enhance the quality of general text content semantic embedding from other answers by using the shared topic distribution to model the relationship between this two modalities(user relationship and text content). We evaluate our model for question answering and expert recommendation task, and the experimental results on two real-world datasets show the effectiveness of our MMSE model for semantic embedding learning.

(2) *Context-Dependent Propagating based Video Recommendation in Multimodal Heterogeneous Information Networks*: With the emergence of online social networks (OSNs), video recommendation has come to play a crucial role in mitigating the semantic gap between users and videos. Conventional approaches to video recommendation focus primarily on exploiting content features or simple user-video interactions to model users' preferences. While these methods have achieved promising results, they fail to model complex video context interdependency, which is obscure/hidden in heterogeneous auxiliary data from OSNs. In this work, we study the problem of video recommendation in Heterogeneous Information Networks (HINs) due to its excellence in characterizing heterogeneous and complex context information. We propose a Context-Dependent Propagating Recommendation network (CDPRec) in order to obtain accurate video embedding and capture global context cues among videos in HINs. CDPRec can iteratively propagate a video's contexts along links in a graph-structured HIN and explore multiple types of dependencies among the surrounding video nodes. Each video is then represented as the composition of the multimodal content feature and global dependency structure information using an attention network. The learned video embedding together with sequential based recommendation are jointly optimized for the final rating prediction. Experimental results on real-world YouTube video recommendation scenarios demonstrate the effectiveness of the proposed methods compared with strong baselines.

Third, we also study knowledge-graph-aware recommender systems. Existing recommendation approaches — especially the collaborative filtering (CF) — suffer from the cold-start and sparsity problem. Knowledge graph (KG), which commonly consist of fruitful connected facts about items, present an unprecedented opportunity to alleviate these problems. In this work, we design a neural architecture that transfers useful auxiliary information from the knowledge domain to a target recommendation domain. However, building recommender systems using knowledge graph faces three challenges: 1) how to learn item representation from complex heterogeneous structured KG, 2) how to model user profile from weighted item attributes and 3) how to capture complex non-linear interactions between users and items. To address the three aforementioned challenges simultaneously, we propose a novel Knowledge Enhanced Neural Collaborative Recommendation (K-NCF) framework with three parts:1) an item aggregation embedding part, 2) an attention-based user modelling part and 3) a convolutional neural collaborative recommendation part. For items, we present Graph Convolutional Network (GCN) based aggregation modal that learns the representation of an item entity by aggregating information from its multi-hop neighbours in KG to mine users' potential preferences. For users, the heterogeneous attention strengths are leveraged to strengthen the embedding learning of users. The user and latent vector are then fed into a newly designed two-dimensional interaction map with convolutional hidden layers to explicitly model the complex high-order pairwise correlations between the dimensions of the user and item embedding.

Chapter 2

Related Works

2.1 Introduction

In recent years, network feature learning has become a popular direction in machine learning, which attempts to learn a low-dimensional vector representation for each node in the network structure. This low-dimension representation can preserve the structural information between the original nodes. Recommendation systems contain lots of network structure data, which is not used effectively in traditional approaches. Therefore, by combining network feature learning methods with recommendation systems, the relevant features of the network data can be better exploited, thus enhancing the learning capability of the recommendation system. In this section, we briefly review the related work including those in the areas of network representation learning, graph convolutional network, graph-enhanced recommendation and neural collaborative filtering.

2.2 Network Representation Learning

Network Representation Learning (NRL) or Network embedding (NE) methods have shown outstanding performance on many tasks including node classification [1], link prediction [5] and community detection [6]. These methods aim to encode network nodes into a low-dimensional vector space while preserving network topology structure information. Figure 2.1 visualizes an example embedding of the famous Zachary Karate Club social network [1], where two dimensional node embeddings capture the community structure implicitly in the social network. DeepWalk [1] uses skip-gram over a random walk based node sequence to learn network representation. LINE [7] models both local and global network structures by using first-order and second-order proximities. Recently, there have been several attempts to apply the notion of "graph convolutions" into network embedding, which has resulted in a new version of graph convolutions based on spectral graph theory. Typical examples include GCN [8], GraphSAGE [9], and the state-of-the-art model GAT [10].

Besides network structures, some methods have also aimed to incorporate side information into network embedding. There are two kinds of side information available to be incorporated in network embedding: node content and types of nodes and edges. MMDW [11] extends the DeepWalk method by taking the label information of nodes into account. TADW [12] incorporates text features associated with nodes into network embedding under the framework of matrix factorization

The above methods are designed for embedding a homogeneous graph with a single type of nodes. Several works have tried to incorporate the structural information

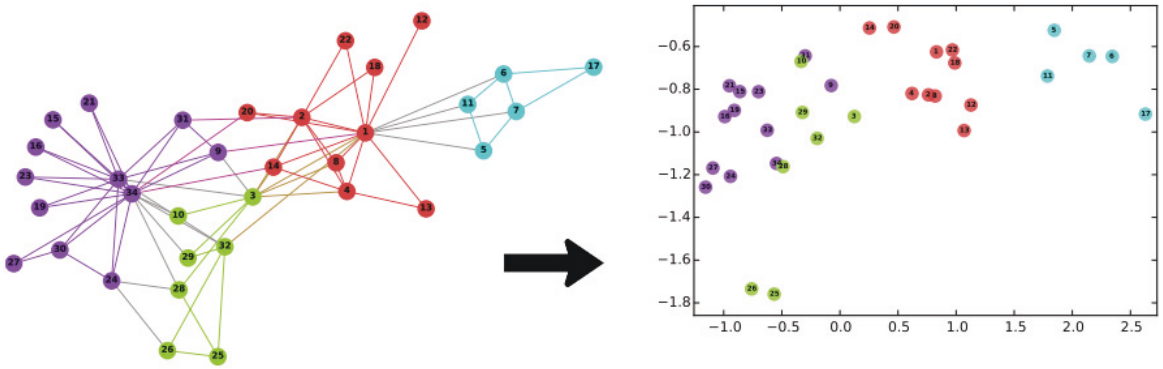


Figure 2.1 : Graph structure of the Zachary Karate Club social network (left) and Two-dimensional visualization of node embeddings generated from this graph using the DeepWalk method (right). Reprinted from [1].

of a Heterogeneous Interaction Network(HIN) into embeddings. Dong et al. [13] use meta-path based random walks over HIN to construct the heterogeneous neighborhood of a node, then feed these walk paths into a skip-gram model to generate node embeddings. Fu et al. [14] propose HIN2Vec, which adopts a logistic classification method to learn the node embeddings as well as meta-paths in HIN. Although these methods can embed various heterogeneous networks, their representations ignore the deep interdependencies among nodes and the rich content features of nodes, and may thus not be optimal for recommendation task.

Knowledge-graph is a special kind of heterogeneous network, and knowledge graph representation aims to learn a low-dimensional vector for each entity and relation in the knowledge graph, while preserving the original graph structure. KGE is used to embed entities or relations in a KG into continuous vector spaces, so that KG can be easier to process while still preserving the inherent structural information [15]. In general, existing KGE methods can be classified into two categories: (1) Translation-based embedding methods try to model the relations between entities as translations from head entity to tail entity by exploiting distance-based loss functions, such as TransE [16], TransH [17], and TransR [18]; (2) Semantic matching methods try to model plausibility of entity pairs by exploiting similarity-based loss function between low-dimension latent semantics vectors of entities and relations, such as ANALOGY [19], RESCAL [20] and HolE [21]. To enrich the KGE semantic information, some recent works are proposed to learn KGE by making full use of auxiliary data, such as textual attributes [22] and entity types [23]. However, the above KGE methods can hardly mine high-order potential interactions among entities, which is crucial to exploit users' potential interests to a long-range distance.

Recently, Graph Neural Networks [24] was introduced to apply neural networks in the graph structure data. GCN [8] extend CNN to aggregate information from long-term dependency graph structure context, which has received increasing research attention. There are two streams of GCN construction: spectral GCN [8, 25]

and spatial GCN [26, 27, 28]. Spectral GCN is based on the spectrum of graph Laplacian. Features of the graph are firstly transformed using Fourier transform and the convolutions are performed in the spectral domain. Spatial GCN approaches define convolutions directly on general graphs, operating on spatially close neighbours. Our method falls into this category, which offers us a chance to perform embedding propagation (or message passing) along the graph structure to faraway neighbour context. At its core is the neighbourhood aggregation, which iteratively aggregates the representations of each node with that of its adjacent nodes as its new representations. Inspired by the idea, researchers attempt to leverage GNNs to model high-order connectivity in recommender systems [29, 30]. For example, PinSage [29] employs the graph convolutional networks [9] to the pin-board bipartite graph in Pinterest and then update the representation of each pin. Wu et al. [30] use GCNs on user/item intrinsic structure graphs to learn user/item representations. The difference between these works and ours is that they are all designed for homogeneous bipartite graphs or user/item similarity graphs where GCNs can be used directly, while here we investigate GCNs for heterogeneous KGs.

2.3 Graph-enhanced recommendation

Based on how these works utilize the graph information, prior efforts on Graph-enhanced recommendation are roughly categorized into into two categories: embedding-based methods and path-based methods.

2.3.1 Path-based methods

Path-based methods [31, 32, 33] can also be called Heterogeneous Information Network(HIN)-based recommendation. In this method, meta-path is introduced to refine the similarities between users and items, which explore the various patterns of connections among items in social network or KG to provide additional guidance for recommendations. *PathSim* [34] is commonly used to measure the connectivity similarity between entities in the graph, which is defined as

$$s_{x,y} = \frac{2 \times |\{p_{x \rightsquigarrow y} : p_{x \rightsquigarrow y} \in \mathcal{P}\}|}{|\{p_{x \rightsquigarrow x} : p_{x \rightsquigarrow x} \in \mathcal{P}\}| + |\{p_{y \rightsquigarrow y} : p_{y \rightsquigarrow y} \in \mathcal{P}\}|} \tag{2.1}$$

where $p_{m \rightsquigarrow n}$ is a path between the entity m and n. One type of path-based method leverages semantic similarities of entities in different meta-paths as the graph regularization to refine the representation of users and items in the HIN. There are three kinds of commonly used path similarity:(1) User-User Similarity; (2)Item-Item Similarity; (3)User-Item Similarity. For example, HeteRec[35] leverages the meta-path similarities to enrich the user-item interaction matrix, so that more comprehensive representations of users and items can be extracted. FMG[33] extract meta-graph based latent features to represent the connectivity between users and items along different types of relation graphs. SemRec[36] uses a weighted HIN and weighted meta-path to integrate attribute values in the link, which considers the interaction of user's favored and hated past items.

Recently, some frameworks have been proposed to learn context-based path embedding that explicitly exploit long-term dependencies relational context in HIN,

which characterise a three-way interaction of the form: (user, HIN, item) tailored for the recommendation task. MCRec[37] learns the explicit representations of meta-paths to depict the interaction context of user-item pairs. RippleNet [38] recently construct ripple set (*i.e* high-order neighbouring items derived from KG) for each user to enrich her representations. While explicitly modelling high-order connectivity, it is highly challenging in real-world recommendation scenarios because most of these methods require extensive domain knowledge to define meta-paths or labor-intensive feature engineering to obtain qualified paths. Moreover, the scale of paths can easily reach millions or even larger when a large number of KG entities are involved, making it prohibitive to efficiently transfer knowledge.

2.3.2 Embedding-based methods

Another line of research leverages network representation learning (NRL)(or network embedding) techniques to regularize the representations of items [39, 40]. As a result, items with similar connected entities have similar representations, which facilitate the collaborative learning of user interests. Recently, NRL is becoming a new research direction in the field of machine learning, such as DeepWalk[1], Node2vec[41], TransE[16] and TransR[18], NRL aims to represent each vertex in a network (graph) as a low-dimensional vector while still preserving its structural information. Due to the existence of massive networks in RS, it is a promising approach to combine NRL with RS. Using NRL to process the network data in RS can improve the capacity of recommendation and boost the precision as well as user satisfaction of recommended results, thereby enhancing the overall economic efficiency. Signed Heterogeneous Information Network Embedding (SHINE) [42] designs deep autoencoders to embed sentiment networks, social networks and profile (knowledge) networks for celebrity recommendations. To exploit the knowledge graph information, knowledge graph embeddings (KGE) algorithms are applied to encode the knowledge graph into lower-order embeddings, such as TransE, TransH. KGE can provide more comprehensive representation of items, and then integrate the item-assistance information into the recommendation framework. The item-assisted information is then integrated into the recommendation framework. For example, Deep Knowledge-aware Network (DKN)[43] treats entity embeddings and word embeddings as different channels, then designs a CNN framework to combine them together for news recommendation. Collaborative Knowledge base Embedding (CKE) [39] combines a CF module with knowledge embedding, text embedding, and image embedding of items in a unified Bayesian framework.

In recent years, Graph Convolutional Network (GCN) techniques have gained considerable interests which can naturally integrate node information and topological structure. GCN can stimulate the propagation of user preferences over the set of knowledge entities by iteratively and automatically extending potential interests along with links in the knowledge graph. GCN is proposed to collectively propagate and aggregate information from the graph structure, which can thus model input and output consisting of the nodes' feature and their interdependency. Traditional KGE methods only consider the local structure of an entity, which is far from modelling users' varying interest in the knowledge. Different from traditional KGE, GCN can

explore entities' potential high-order structural proximity on KG entities with deep learning-based models. For every entity node in the KG, GCN encodes relevant information about its neighbourhood as a real-valued feature vector in an unsupervised manner. For example, RippleNet[38] and KGCN [44] apply a graph neural network to compute personalised item embeddings, which can capture item relationship by mining their associated attributes on the KG for recommendation. Recently, KGNN-LS [45] provides extra regularisation over the edge weights for recommendation by adding label smoothness to KGCN method. Embedding-based methods show high flexibility in utilizing graph-structure data to assist recommender systems, but the adopted NRL algorithms in these methods are usually more suitable for in-graph applications such as link prediction than for recommendation, thus the learned entity embeddings are less intuitive and effective to characterize inter-item relations.

2.4 Neural Collaborative Filtering

There are two types of methods for implementing an effective neural collaborative filtering model [46, 47, 48]. One is based on representation learning and the other one is based on matching function learning.

Since early work Funk-SVD [49] win the Netflix Prize competition, matrix factorisation based methods have become the mainstream of recommendation research over the past ten years. [50, 51, 52, 53]. These works try to assist matrix factorisation by incorporating auxiliary information, such as time, social relationship, text description, and location. Recently, there are also some studies proposed to use deep learning methods for collaborative filtering based on representation learning. AutoRec [54] is the first model that try to use autoencoder to learn user and item representation. DMF [55] was proposed matrix factorisation model with a two pathway neural network architecture to factorise rating matrix and learn user/item representations into low dimensional vectors. Overall, representation learning-based methods learn representation in different ways and can flexibly incorporate with auxiliary data such as images, text descriptions, demographic information and so on. However, they still resort to the dot product or cosine similarity when predicting matching scores.

Neural collaborative filtering(NCF) [46] is a recently proposed framework that unifies MF and MLP in one model, which replaces the dot product used in vanilla MF with a neural network to learn the matching function between users and items. NNCF [56] is a variant of NCF that takes user neighbors and item neighbors as inputs. To better capture pairwise correlations user and item dimension, ConvNCF [48] replace concatenation used in NeuMF as an outer product operation. Other than NCF, there are also many other works that introduce auxiliary information to learn the matching function. For example, Wide&Deep [57] learn the matching function by adapting LR and MLP with the help of categorical features of users and items.

Chapter 3

AAANE: Attention-based Adversarial Autoencoder for Multi-scale Network Embedding

Network embedding represents nodes in a continuous vector space and preserves structure information from a network. Existing methods usually adopt a "one-size-fits-all" approach when concerning multi-scale structure information, such as first- and second-order proximity of nodes, ignoring the fact that different scales play different roles in embedding learning. In this chapter, we propose an Attention-based Adversarial Autoencoder Network Embedding (AAANE) framework, which promotes the collaboration of different scales and lets them vote for robust representations. The proposed AAANE consists of two components: 1) an attention-based autoencoder that effectively capture the highly non-linear network structure, which can de-emphasize irrelevant scales during training, and 2) an adversarial regularization guides the autoencoder in learning robust representations by matching the posterior distribution of the latent embeddings to a given prior distribution. Experimental results on real-world networks show that the proposed approach outperforms strong baselines.

3.1 Introduction

Network embedding (NE) methods have shown outstanding performance on many tasks including node classification [1], community detection [6, 58] and link prediction [5]. These methods aim to learn latent, low-dimensional representations for network nodes while preserving network topology structure information. Networks' structures are inherently hierarchical [59]. As shown in Figure 3.1, each individual is a member of several communities and can be modeled by his/her neighborhoods' structure information with different scales around him/her, which range from short scales structure (e.g. families, friends), to long-distance scales structure (e.g. society, nation states). Every single scale is usually sparse and biased, and thus the node embedding learned by existing approaches may not be so robust. To obtain a comprehensive representation of a network node, multi-scale structure information should be considered collaboratively.

Recently, a number of methods have been proposed for learning data representations from multiple scales. For example, DeepWalk [1] models multi-scale indirectly from a random walk. Line [7] proposes primarily a breadth-first strategy, sampling nodes and optimizing the likelihood independently over short scales structure information such as 1^{st}-order and 2^{st}-order neighbors. GraRep [60] generalizes LINE to incorporate information from network neighborhoods beyond 2-order, which can embed long distance scales structure information to the node representation. More

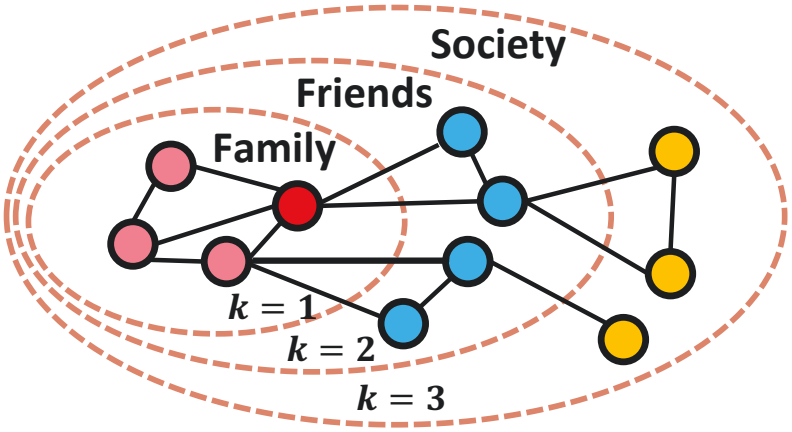


Figure 3.1 : Illustration of the multi-scale network with three scales.

recently, some autoencoder based methods, For example, DNGR [61] learns the node embedding through stacked denoising autoencoder from the multi-scale Positive Pointwise Mutual Information (PPMI) matrix. Similarly, SDNE [62] is realized by a semi-supervised deep autoencoder model. Besides, MVE [63] aims to learn embedding from several multi-viewed networks with the same nodes but different edges, which is different from our single network setting.

Despite their strong task performance, existing methods have the following limitations: (1) *Lack of weight learning.* To learn robust and stable node embeddings, the information from multiple scales needs to be integrated. During integration, as the importance of different scales can be quite different, their weights need to be carefully decided. For example, if we consider very young kids on a social network, and they may be very tightly tied to their family and loosely tied to the society members. However, for university students, they may have relatively more ties to their friends and the society than very young kids. Existing approaches usually assign equal weights to all scales. In other words, different scales are equally treated, which is not reasonable for most multi-scale networks. (2) *Insufficient constrain for embedding distribution.* Take the autoencoder based method for example, an autoencoder is a neural network trained to attempt to copy its input to its output, which has a typical pipeline like ($x \rightarrow E \rightarrow z \rightarrow D \rightarrow x'$). Autoencoder only requires x to approach $x' = D(E(x))$, and for that purpose the decoder may simply learn to reconstruct x regardless of the distribution obtained from E. This means that $p(z)$ can be very irregular, which sometimes makes the generation of new samples difficult or even infeasible.

In this chapter, we focus on the multi-scale network embedding problem and propose a novel Attention-based Adversarial Autoencoder Network Embedding (AAANE) method to jointly capture the weighted scale structure information and learn robust representation with adversarial regularization. We first introduce a set of scale-specific node vectors to preserve the proximities of nodes at different scales. The scales-specific node embeddings are then combined for voting the robust node repre-

sentations. Specifically, our work has two major contributions. (1) To deal with the weights learning, we propose an attention-based autoencoder to infer the weights of scales for different nodes, and then capture the highly non-linear network structure, which is inspired by the recent progress of the attention mechanism for neural machine translation [64]. (2) To implement regularisation of the distribution for encoded data, we introduce adversarial training component [65] to the attention-based autoencoder, which can discriminatively predict whether a sample arises from the low-dimensional representations of the network or from a sampled distribution. Adversarial regularisation reduces the amount of information that may be held in the encoding, forcing the model to learn an effective representation of the data. Through the attention-based weight learning together with the adversarial regularization, the proposed AAANE model can effectively combine the virtues of multiple scale information to complement and enhance each other.

3.2 Preliminary

Network Embedding(NE): An information network is represented as $G = (V, E)$, where $V = \{v_i\}_{i=1,\cdots,N}$ consist a set of nodes, $e_{i,j} = (v_i, v_j) \in E$ is an edge indicating the relationship between two nodes. The task of NE aims to build a low-dimensional representation $x_i \in \mathbb{R}^d$ for each node $i \in V$, where d is the dimension of embedding space and expected much smaller than node number $|V|$.

We define *adjacency matrix* $\widetilde{A} \in \mathbb{R}^{|V|\times|V|}$for a network and D is a diagonal degree matrix. To capture the transitions from one node to another, we can define the (first-order) probability transition matrix $A = D^{-1}\widetilde{A}$, where $A_{i,j}$ is the probability of a transition from node v_i to node v_j within one step. It can be observed that the matrix A is a normalized adjacency matrix where the summation of each row equals to 1.

In this chapter, multi-scale structural information serves two functions: 1) capturing of long-distance relationship between two different vertices and 2) the consideration of distinct connections in terms of different transitional orders.

The (normalized) adjacency matrix A characterizes the first-order proximity which models the local pairwise proximity between vertices. As discussed earlier, we believe that the k-order (with varying k) long scale relational information from the network needs to be captured when constructing such multi-scale network embedding [60]. To compute the various scale transition probabilities, we introduce the following k-order probability proximity matrix:

$$A^k = \underbrace{A \cdot A \cdots A}_{k}. \tag{3.1}$$

where the entry $A^k_{i,j}$ refers to the k-order proximity between node v_i and v_j.

Multi-Scale Network Embedding: Given a network $G = (V, E)$, the robust node representation $\{x_i\}_{v_i\in V} \subseteq R^d$ can be collaboratively learned from k successively network structural information representation, $A, A^2, \ldots, A^k$, where A^k captures the

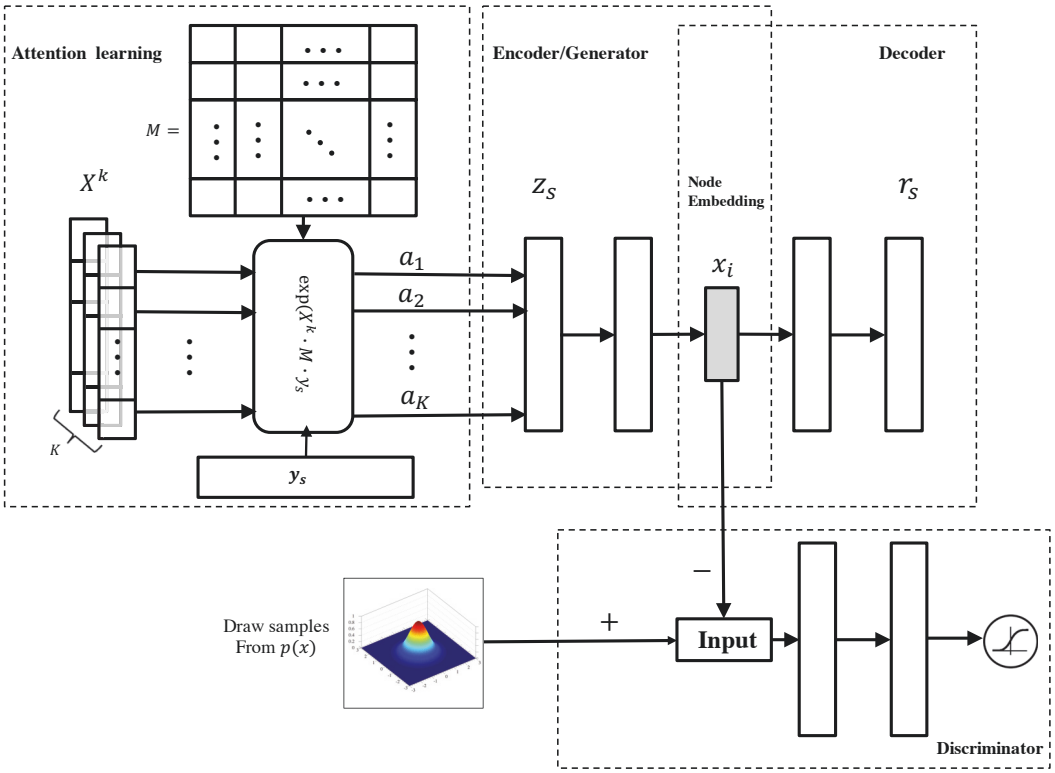


Figure 3.2 : The architecture of AAANE. To learn the low dimension of node representation, we employ autoencoder with non-linear activation function to capture complex, non-linear projections from the inputs to the outputs. The We first introduce an attention mechanism to the autoencoder for learning the weights of structure information with different scales. Finally, the adversarial learning component acts as regularization for the autoencoder, by matching the aggregated posterior, which helps enhance the robustness of the representationThe.

view of the network at scale k. Intuitively, each member of the family encodes a different view of social similarity, corresponding to shared membership in latent communities at different scales.

3.3 Method

3.3.1 An overview of the framework

In this work, we leverage attention-based adversarial autoencoder to help learn stable and robust node embedding. Figure 3.2 shows the proposed framework of Attention-based Adversarial Autoencoder for Network Embedding (**AAANE**), which mainly consists of two components, i.e., an attention-based autoencoder and an adversarial learning component.

We introduce an attention mechanism to the autoencoder for learning the weights of structure information with different scales. A standard autoencoder consists of an encoder network and a decoder network. The encoder maps the network structure information z_s into a latent code x_i , and the decoder reconstructs the input data as r_s. Then, the adversarial learning component acts as regularization for the autoencoder, by matching the aggregated posterior, which helps enhance the robustness of the representation x_i. The generator of the adversarial network is also the encoder of the autoencoder. The adversarial network and the autoencoder are trained jointly in two phases: the reconstruction phase and the regularization phase. In the reconstruction phase, the autoencoder updates the encoder and the decoder to minimize the reconstruction error of the inputs. In the regularization phase, the adversarial network first updates its discriminative network to tell apart the true samples (generated using the prior) from the generated samples (the hidden node embedding x_i computed by the autoencoder). As a result, the proposed AAANE can jointly capture the weighted scale structure information and learn robust representations.

3.3.2 Attention-based Autoencoder

The Attention-based Autoencoder for network embedding(**AANE**) model uses a stacked neural network to preserve the structure information. As we discussed in section 2, different k-order proximity matrices preserve network structure information in different scales. Scale vector X^k is column in each A^k, for $k = 1, 2 \dots K$, which denotes the k-th scale structure information for the node. The length of each scale vector X^k is the same as the node size. The autoencoder component tries to capture the full range of structure information. We construct a vector representation z_s for each node as the input of the autoencoder in the first step. In general, we expect this vector representation to capture the most relevant information with regards to different scales of a node. z_s is defined as the weighted summation of every scale vector $X^k, k = 1, 2, \dots K$, corresponding to the scale index for each node.

$$z_s = \sum_{k=1}^{K} a_k X^k \tag{3.2}$$

For each scale vector X^k of one node, we compute a positive weight a_k which can be interpreted as the probability that X_k is assigned by one node. Intuitively, by learning proper weights a_k for each node, our approach can obtain most informative scale information. Following the recent attention based models for neural machine translation, we define the weight of scales k for a node using a softmax unit as follows:

$$\begin{aligned} a_k &= \frac{\exp(d_k)}{\sum_{j=1}^{N} \exp(d_k)} \\ d_k &= {X^k}^\top \cdot M \cdot y_s \qquad y_s = \frac{1}{K} \sum_{k=1}^{K} X^k \end{aligned} \tag{3.3}$$

where y_s is the average of different scale vector, which can capture the global context of the structure information. M is a matrix mapping between the global context embedding y_s and each structure scale vector X^k, which is learned as part of the training process. By introducing an attentive matrix M, we compute the relevance of each scale vector to the node. The attention mechanism is performed on the input. So the dimension of matrix M will be very large for large-scale networks. We tried to apply the attention operation to the bottleneck layer to reduce this matrix, but the performance was not good. Therefore, to avoid using this attentive matrix, we consider using self-attention mechanism [66] in our future work. Self-attention is a special case of attention where the query stems from the input source itself without the help of attentive matrix M. If X^k and y_s have a large dot product, this node believes that scale k is an informative scale, i.e., the weight of scale k for this node will be largely based on the definition.

Once we obtain the weighted node vector representation $z_s \in \mathbb{R}^{|V|}$, a stacked autoencoder is used to learn a low-dimensional node embedding. An autoencoder performs two actions, an encoding step, followed by a decoding step. In the encoding step, a function $f()$ is applied to the original vector representation z_s in the input space and send it to a new feature space. An activation function is typically involved in this process to model the non-linearities between the two vector spaces. At the decoding step, a reconstruction function $g()$is used to reconstruct the original input vectors back from the latent representation space. The r_s is the reconstructed vector representation. After training, the bottleneck layer representations x_i can be viewed as the low dimension embedding for the input node v_i.

This attention-based autoencoder is trained to minimize the reconstruction error. We adopt the contrastive max-margin objective function, similar to previous work [67, 68, 69]. For each input node, we randomly sample m nodes from our training data as negative samples. We represent each negative sample as n_s, which is computed by averaging its scale vectors as y_s. Our objective is to make the reconstructed embedding r_s similar to the target node embedding z_s while different from those negative samples n_s. Therefore, the unregularized objective J is formulated as a hinge loss that maximizes the inner product between r_s and z_s, and minimizes

the inner product between r_s and the negative samples simultaneously:

$$J(\theta) = \sum_{s \in D} \sum_{i=1}^{m} max(0, 1 - r_s z_s + r_s n_i) \tag{3.4}$$

where D represents the training dataset and θ represents the model parameters..

3.3.3 Adversarial Learning

We hope to learn vector representations of the most representative scale for each node. An autoencoder consists of two models, an encoder and a decoder, each of which has its own set of learnable parameters. The encoder is used to get a latent code x_i from the input with the constraint. The dimension of the latent code should be less than the input dimension. The decoder takes in this latent code and tries to reconstruct the original input. However, we argue that training an autoencoder with contrastive max-margin objective function gives us latent codes with similar nodes being far from each other in the Euclidean space, especially when processing noisy network data. The main reason is the insufficient constrain for embedding distribution. Adversarial autoencoder (AAE) addresses these issues by imposing an Adversarial regularization to the bottleneck layer representation of autoencoder, and then the distribution of latent code may be shaped to match a desired prior distribution. Adversarial regularisation can reduce the amount of information that may be held in the encoding process, forcing the model to learn an efficient representation for the network data.

AAE typically consists of a generator $G()$ and a discriminator $D()$. Our main goal is to force output of the encoder to follow a given prior distribution $p(x)$(this can be normal, gamma .. distributions). We use the encoder as our generator, and the discriminator to tell if the samples are from a prior distribution or from the output of the encoder x_i. D and G play the following two-player minimax game with the value function V (G, D):

$$\min_G \max_D V(D, G) = \mathbb{E}_{p(x)}[\log D(x_i)] + \mathbb{E}_{q(x)}[\log(1 - D(x_i))] \tag{3.5}$$

where $q(x)$ is the distributions of encoded data samples.

3.3.4 Training Procedure

The whole training process is done in three sequential steps: (1) The encoder and decoder are trained simultaneously to minimize the reconstruction loss of the decoder as equation 3.4. (2) The discriminator D is then trained to correctly distinguish the true input signals x from the false signals x_i, where the x is generated from target distribution, and x_i is generated by the encoder by minimizing the loss function 3.5. (3) The next step will be to force the encoder to fool the discriminator by minimizing another loss function: $L = -\log(D(x_i))$. More specifically, we connect the encoder output as the input to the discriminator. Then, we fix the discriminator weights and fix the target to 1 at the discriminator output. Later, we pass in a node to the encoder and find the discriminator output which is then used to find the loss.

3.4 Experiments

In this section, we conduct node classification on sparsely labeled networks to evaluate the performance of our proposed model.

3.4.1 Datasets

We employ the following three widely used datasets for node classification.

Cora. Cora is a research paper set constructed, which contains 2, 708 machine learning papers which are categorized into seven classes. The citation network consists of 5429 links(citations). The citation relationships among them are crawled form a popular social network.

Citeseer. Citeseer is another research paper set constructed, which contains 3, 312 publications and 4, 732 links among them. These papers are from 6 classes:Agents, AI, DB, IR, ML and HCI. The papers were selected in a way such that in the final corpus every paper cites or is cited by at least one other paper.

Wiki. The Wikipedia is an online encyclopedia created and edited by volunteers around the world. The data set is a word co-occurrence network constructed from the entire set of English Wikipedia pages. Wiki contains 2, 405 web pages from 19 categories and 17, 981 links among them. Wiki is much denser than Cora and Citeseer.

3.4.2 Baselines and Experimental Settings

We consider a number of baselines to demonstrate the effectiveness and robustness of the proposed AAANE algorithm. For all methods and datasets, we set the embedding dimension $d = 128$.

DeepWalk [1]: DeepWalk first transforms the network into node sequences by truncated random walk, and then uses it as input to the Skip-gram model to learn representations.

LINE [7]: LINE can preserve both first-order and second-order proximities for the undirected network through modeling node co-occurrence probability and node conditional probability.

GraRep [60]: GraRep preserves node proximities by constructing different k-order transition matrices.

node2vec [41]: node2vec develops a biased random walk procedure to explore the neighborhood of a node, which can strike a balance between local properties and global properties of a network.

AIDW [70]: Adversarial Inductive DeepWalk (AIDW) is an Adversarial Network Embedding (ANE) framework, which leverages random walk to sample node sequences as the structure-preserving component.

Parameter setting: In our experimental settings, we vary the percentage of labeled nodes from 10% to 90% by an increment of 10% for each dataset. We treat network embeddings as vertex features and feed them into a one-vs-rest logistic

Table 3.1 : Accuracy (%) of node classification on Wiki.

% Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	57.2	62.98	64.03	65.78	66.74	68.69	68.36	67.85	67.22
LINE	57.09	59.98	62.47	64.38	66.5	65.8	67.31	67.15	65.15
GraRep	59.55	60.76	62.23	62.3	62.76	63.72	63.02	62.79	60.17
node2vec	58.47	61.38	63.9	63.96	66.08	66.74	67.73	67.57	66.8
AIDW	57.29	61.89	63.77	64.26	66.85	67.23	69.04	70.13	71.33
AANE	59.95	64.14	66.15	68.40	68.66	69.34	69.25	70.89	69.71
AAANE	**60.36**	**64.98**	**67.21**	**68.79**	**69.07**	**70.32**	**70.85**	**72.03**	**72.45**

Table 3.2 : Accuracy (%) of node classification on Cora.

% Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	76.37	79.6	80.85	81.42	82.35	82.1	82.9	84.32	83.39
LINE	71.08	76.19	77.32	78.4	79.25	79.06	79.95	81.92	82.29
GraRep	77.02	77.95	78.53	79.75	79.61	78.78	78.6	78.23	78.23
node2vec	75.84	78.77	79.54	80.86	80.43	80.9	80.44	79.7	77.86
AIDW	76.21	78.93	80.21	81.45.	82.03	82.74	82.81	83.69	83.92
AANE	77.65	81.50	82.49	84.43	84.71	84.69	84.75	85.98	86.03
AAANE	**78.23**	**82.14**	**82.76**	**85.31**	**85.69**	**86.12**	**86.02**	**86.74**	**87.21**

Table 3.3 : Accuracy (%) of node classification on Citeser.

% Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	53.47	54.19	54.6	57.55	57	59.02	58.95	58.22	55.72
LINE	48.74	50.87	52.82	52.72	52	52.3	53.12	53.54	52.41
GraRep	53.23	54.34	53.77	54.43	54.05	54.57	54.83	55.35	55.12
node2vec	53.94	54.08	56.23	57.34	57.55	60.3	61.17	61.24	59.33
AIDW	52.17	56.23	56.87	58.26	58.45	59.27	59.34	60.38	61.3
AANE	55.02	56.15	58.65	58.76	58.52	59.93	60.97	61.39	61.23
AAANE	**55.45**	**56.73**	**59.37**	**59.81**	**60.12**	**60.58**	**61.43**	**61.72**	**62.38**

regression classifier implemented by LibLinear[71]. For DeepWalk, LINE, GraRep, node2vec, we directly use the implementations provided by OpenNE*. For our methods AAANE, the maximum matrix transition scale is set to 8, and the number of negative samples per input sample m is set to 7. For attention-based autoencoder, it has three hidden layers, with the layer structure as $512 - 128 - 512$. For the discriminator of AAANE, it is a three-layer neural network, with the layer structure as $512 - 512 - 1$. And the prior distributions are Gaussian Distribution following the original paper [72].

3.4.3 Multi-label Classification

Table 3.1, Table 3.2 and Table 3.3 show classification accuracies with different training ratios on different datasets, where the best results are **bold-faced**. In these tables, AANE denotes our model AAANE without Adversarial component. From these tables, we have the following observations:

1. The proposed framework, without leveraging the adversarial regularization version AANE, achieving average 2% gains over AIDW on cora and wiki when varying the training ratio from 10% to 90% in most cases, and slightly better result on Citeseer, which suggests that assigning different weights to different scales of a node may be beneficial.

2. After introducing Adversarial component into AANE, our Method AAANE can achieve further improvements over all baselines. It demonstrates that adversarial learning regularization can improve the robustness and discrimination of the learned representations.

3. AAANE consistently outperforms all the other baselines on all three datasets with different training ratios. It demonstrates that attention-based weight learning together with the adversarial regularization can significantly improve the robustness and discrimination of the learned embeddings

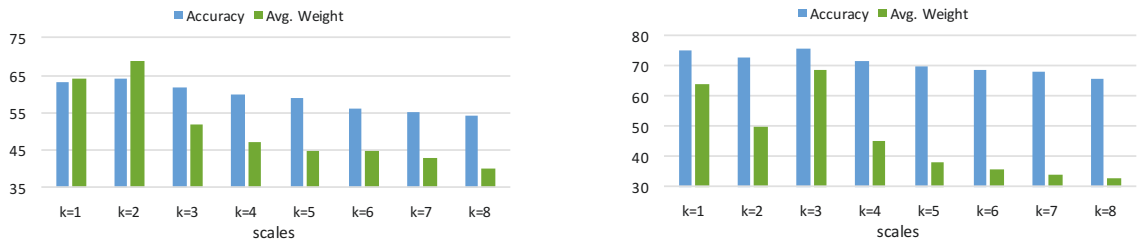
3.4.4 Detailed Analysis of The Proposed Model

Analysis of the Learned Attentions Over Scales

In our proposed AAANE model, we adopt an attention based approach to learn the weights of scales during voting, so that different nodes can focus most of their attentions on the most informative scales. The quantitative results have shown that AAANE achieves better results by learning attention over scales. In this part, we will examine the learned attention to understand why it can help improve the performances.

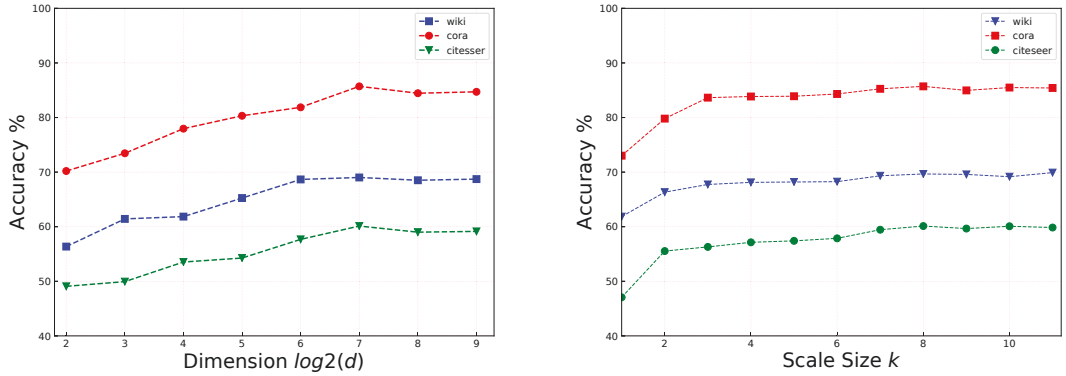
We study which scale turn to attract more attentions from nodes. We take the Cora and Wiki datasets as examples. For each scale, we report the results of the

*https://github.com/thunlp/OpenNE



(a) wiki. (b) cora.

Figure 3.3 : Comparison of performances on each individual scale and the average weights of scales. Scales with better performances usually attract more attentions from nodes.



(a) #Dimension. (b) #Scales.

Figure 3.4 : Parameter sensitivity of dimension d and scale size k.

scale-specific embedding corresponded to this scale, which achieves by taking only one scale vector A^k as an input of autoencoder. Then, we compare this scale-specific embedding with the average attention values learned by AAANE. The results are presented in Figure 3.3. Overall, the performances of single scale and the average attention received by these scales are positively correlated. In other words, our approach can allow different nodes to focus on the scales with the best performances, which is quite reasonable.

Parameter Sensitivity

We discuss the parameter sensitivity in this section. Specifically, we assess how the different choices of the maximal scale size K, dimension d can affect node classification with the training ratio as 50%. Figure 3.4(a) shows the accuracy of AAANE over different settings of the dimension d. The accuracy shows an apparent increase at first. This is intuitive as more bits can encode more useful information in the increasing bits. However, when the number of dimensions continuously increases, the

performance starts to drop slowly. The reason is that too large number of dimensions may introduce noises which will deteriorate the performance. Figure 3.4(b) shows the accuracy scores over different choices of K. We can observe that the setting $K = 2$ has a significant improvement over the setting $K = 1$, and $K = 3$ further outperforms $K = 2$. This confirms that different k-order can learn complementary local information. When K is large enough, learned k-order relational information becomes weak and shifts towards a steady distribution.

3.5 Conclusion

In this chapter, we study learning node embedding for networks with multiple scales. We propose an effective framework to let different scales collaborate with each other and vote for the robust node representations. During voting, we propose an attention-based autoencoder to automatically learn the voting weights of scales while preserving the network structure information in a non-linear way. Besides, an Adversarial regularization is introduced to learn more stable and robust network embedding. Experiments on node classification demonstrate the superior performance of our proposed method.

Chapter 4

Bayesian Semantic Embedding for social-aware Community Question Answering

Semantic embedding has demonstrated its value in latent representation learning of data, and can be effectively adopted for many applications. However, it is difficult to propose a joint learning framework for semantic embedding in Community Question Answer (CQA), because CQA data have multi-view and sparse properties. In this chapter, we propose a generic Multi-modal Multi-view Semantic Embedding (MMSE) framework via a Bayesian model for question answering. Compared with existing semantic learning methods, the proposed model mainly has two advantages: (1) To deal with the multi-view property, we utilize the Gaussian topic model to learn semantic embedding from both local and global views. (2) To deal with the sparse property of question answer pairs in CQA, social structure information is incorporated to enhance the quality of general text content semantic embedding from other answers by using the shared topic distribution to model the relationship between these two modalities (user relationship and text content). We evaluate our model for question answering and expert finding task, and the experimental results on two real-world datasets show the effectiveness of our MMSE model for semantic embedding learning.

4.1 Introduction

Community Question Answer (CQA) sites such as Quora, Yahoo! Answers, Baidu Knows and Zhihu are gaining popularity, for people can exchange knowledge with each other efficiently. These sites usually contain a large number of Question/Answer (Q/A) pairs with other metadata like question's categories and users' relationship. Semantic embedding is one of the most basic tasks to fully reuse these CQA archives, for its wide range of applications such as question retrieval [73, 74], expert finding [75] and question answering [76, 77]. The goal of semantic embedding is to represent CQA data in a latent semantic space. Let's take question answering as an example, which aims to identify the most relevant answers to the queried question within a collection of answers. Then a user asks a new question in CQA site and the best matched historical answer can be located, the lag time incurred by having to wait for a person to respond can be avoided, thus improving user satisfaction.

One of the greatest challenges in question answering is the lexical gap between the query questions and the candidate answers, since data in CQA often contain incomplete and ambiguous information [73], such as question "What is the best laptop?" with an answer "For a programmer, I recommend Macbook". Although this Q/A pair shares no words in common, they are strongly associated with synonyms,

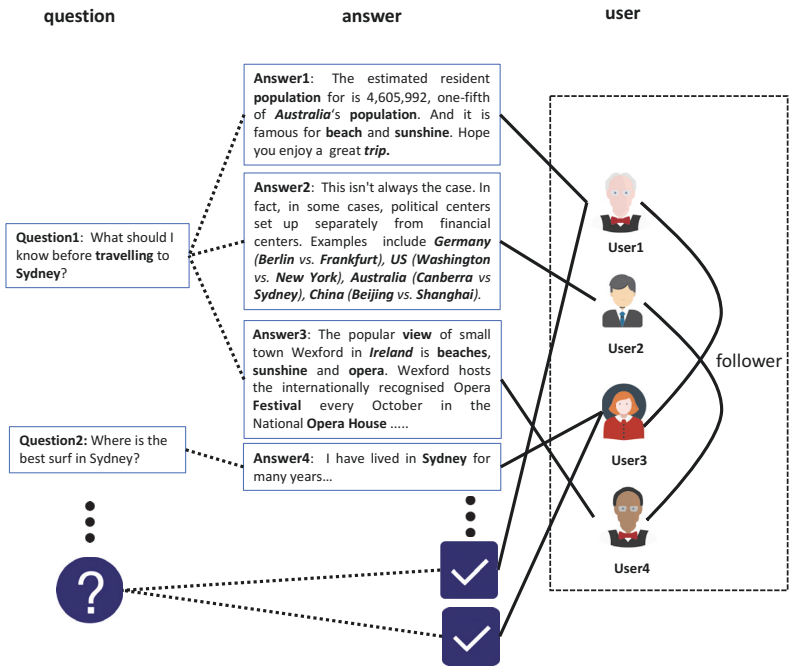


Figure 4.1 : An example of multi-modal multi-view semantic embedding for CQA data. (i) The two modalities, Q/A pairs and corresponding users are explored jointly. (ii) Semantic embedding was modeled from both local and global views. Single-view based embedding method may lead to a biased result, such as local embedding for answer 2 and global topical embedding for answer 3.

hyponyms, or other weaker semantic associations of words. Semantic embedding has been proven its ability to capture this latent similarity between different words and has attracted lots of attention in the fields of information retrieval and natural language processing research.

Semantic embedding for CQA has multi-view property. This is because semantic information with different granularities should be modeled simultaneously in CQA. In Figure 4.1, we show an example about different answers for the question "What should I know before traveling to Sydney?" Since the two main semantic embedding methods, topic model and word embedding, calculate similarity from two different views, we get different answers. Word embedding, such as Skip-gram, learn a vector-space representation for each word, based on statistics about how often each word occurs within a local context window of another word. This local view based word semantic embedding method may choose the answer 2, because the similarity between city names is very high in local word embedding circumstance. In contrast, topic models, such as Latent Dirichlet Allocation (LDA), take a more global view, which assumes that two words are similar if they are often found in the same document. With this global view, we may choose the answer 3, for "view", "opera house", "sunshine" and "festival" are more likely to appear in the same document to depict the the query term "Sydney" and "travel", while neglecting the important fact that local similarity between distinctive place name "Ireland" and "Sydney" is

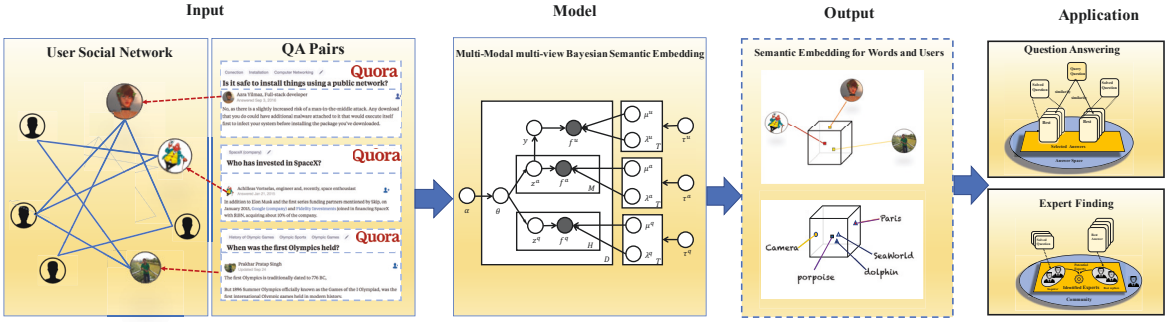


Figure 4.2 : The flowchart of the proposed multi-modal multi-view semantic embedding model.

very low. So neither of this two views can generate unbiased semantic embedding solely. Since local view can help disambiguate word meanings, the global view can also provide useful topical information, it is natural for us to expect to model this two kinds of information simultaneously.

The Sparsity problem of CQA data is another challenging issue [77]. In CQA site, each question is associated with a few answers and thus the question answer pairs are very sparse. Most of the existing method which only utilises the text content of the question answer pair can hardly learn the general semantic form other questions. As is shown in Figure 4.1 each question only connect with a few answers. Since User 1 and User 3 are friends with each other, what they answered such as Answer 1 and Answer 4 may have potential relation with each other, which can provide extra information for the semantic embedding of a single answer and can also alleviate the sparse problem of Q/A pairs. Specifically, the social relationship in CQA provides a natural avenue for enhancing the quality of general semantic embedding from other answers, because the Q/A pairs are all posted by users and CQA site is based on user interaction. Users and Q/A content are an indivisible whole and can promote each other. Therefore, it is necessary and important to take the multi-modal CQA data(text content and user relationship) into consideration simultaneously.

In this chapter, we adopt a Bayesian embedding method to exploit the comprehensive text semantic information from both local and global view, and exploit the user social relationship to solve the sparsity problem in CQA tasks. Our proposed framework is named as **MMSE** (**M**ulti-modal **M**ulti-view **S**emantic **E**mbedding). The goal is to learn latent semantic embedding for multi-modal data from multi-view. As shown in Figure 4.2, the input consists of Q/A pairs and the social network of answerers or repliers. Given the input, the proposed MMSE is adopted to learn the semantic embedding, which can preserve the latent relation between Q/A pairs as well as the user interaction structure. With the derived multi-modal multi-view semantic embedding, when a certain question is queried, MMSE can retrieve the best answer for it. Moreover, the expertise of users learned by our MMSE can be beneficial for other CQA tasks, such as expert finding.

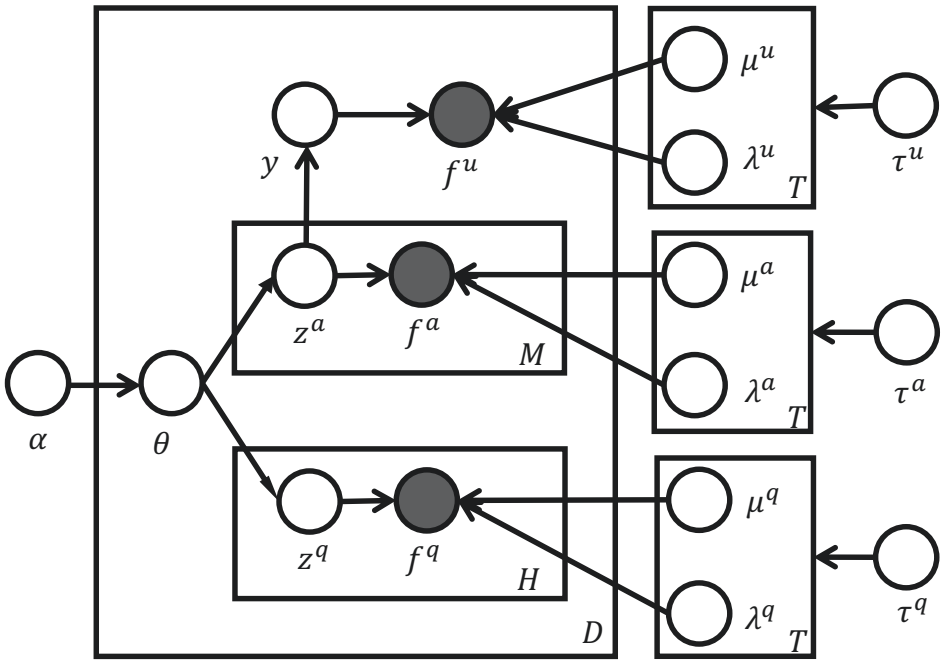


Figure 4.3 : The graphical representation of the proposed multimodal multi-view Semantic Embedding model (MMSE). Each document d contains a single social network user, one answer posted by the user and question related to the answer. Embeddings are observed variables.

4.2 The Proposed MMSE Model

We propose MMSE, a Bayesian embedding model for learning social graphs, which jointly models multiple modalities and multiple semantic views of CQA data. Our goal is to learn a shared latent topic space to generate network-based user embeddings and text-based word embeddings in two different embedding spaces.

The input of our problem is multi-modal CQA data, which includes pre-trained user embedding f^u and word embedding f^q, f^a of Q/A text. In other words, embeddings are given as observed variables in our model. We use the Skip-gram model to learn word embeddings, and use DeepWalk to learn network-based user embeddings. In brief, MMSE tries to get a more comprehensive word embedding and user embedding from pre-trained embedding representation by considering the multiple modalities and multiple semantic views.

4.2.1 Generative Process

The graphical representation of the proposed MMSE is shown in Figure 4.3. In our model, we assume there are two kinds of global topics in multi-modal CQA data: (1) z^q and z^a are latent topic assignments for question words and answer words respectively; Since the asker and answerer may express similar meanings with different words, Q/A pairs should share the same topic space. (2) To jointly model multiple modalities, we assume that the latent topic of users is generated from

Notations	Description
D	The number of Q/A pair documents
T	The number of topics
M	The number of words in answer
H	The number of words in question
W	The number of unique words
E^u, E^w	The dimension of user and word embedding
α	The hyperparameter of the Dirichlet prior on θ
θ	The multinomial distribution of topic specific to the Q/A text
y	The topic of each user
z^q, z^a	The topic of single word in question or answer
f^u	The network based user embedding
f^a, f^q	The word embedding of Q/A text
μ^u, μ^q, μ^a	The mean of Gaussian distribution for user embedding and word embedding
$\lambda^u, \lambda^q, \lambda^a$	The precision of Gaussian distribution for user embedding and word embedding
τ^u, τ^q, τ^a	The hyperparameter of the normal Gamma distribution

Table 4.1 : Key Notations of Our Proposed MMSE Model

their related answer topic space, thus each user is associated with a multinomial distribution over answer topics. Both user topic and Q/A content topic are modeled in a global semantic view.

To model the multi-view property of semantic embedding, MMSE introduce local semantically coherent to the topic model by replacing the original discrete word types in topic model with continuous word embedding and user embedding. This is implemented by considering embedding vector as drawing from several Gaussian distributions in the topic model framework, in view of word embedding and user embedding can capture a notion of centrality in space [78, 79]. Thus, in the joint model, the semantic embedding for multi-modal user and Q/A pairs can be learned from both global topical view and local embedding view. The MMSE takes pre-trained word embedding f^q, f^a and user embedding f^u as inputs, where the f^u is obtained by DeepWalk [1] to preserve the user social network structure information and $f^q, and f^a$ are learned from Skip-gram model [80]. The dimension of word and user are E^w and E^u respectively.

Our multi-modal semantic embedding is continuous vectors, we characterise each dimension of the embedding as a univariate Gaussian distribution with parameter (μ, λ) with Normal-Gamma distribution priors $Normal - Gamma(\tau =$

$\{\mu_0, k_0, \alpha_0, \beta_0\}$)[81]. There are two major reasons to choose univariate Gaussian rather than multivariate Gaussian to generate the embedding. First, model with univariate Gaussian distribution can obtain better performances than that with multivariate Gaussian, which might be caused by the relatively independence between each dimension of embedding representation. The word analogical task introduced by [82] means that the influence between words can be simply computed by addition and subtraction, such as [king] - [man] + [woman] $\approx$ [queen]. Because of these, we can assume that the dimensions between a word embedding are relatively independent which means there are no cross influence between them[83]. So it is much better to use the univariate Gaussian to describe the distribution of each dimension. Second, model univariate Gaussian distribution is more computationally efficient. A univariate normal distribution is described using just the two parameters namely mean μ and precision λ, $X \sim N(\mu, \lambda)$. For a multivariate distribution, we need a third parameters Σ, i.e., the correlation between each pair of random variables, $X \sim N(\mu, \Sigma)$. This is what distinguishes a multivariate distribution from a univariate distribution. If there are n dimensions in the embedding, we will have $n * (n-1)/2$ pairs of correlations.

The choice of Gaussian distribution is also justified by that Euclidean distances between Gaussian distributions is straightforward to calculate, naturally asymmetric, and has a geometric interpretation as an inclusion between families of ellipses [84]. The θ_d is the multinomial topic distribution of document d, which shared by both question and answer text. α is the hyperparameter of the Dirichlet distribution. $\tau = \alpha_0, \beta_0, k_0, \mu_0$ is hyperparameter of the normal Gamma distribution. Accordingly, the generative process of a document in the proposed MMSE model can be described as follows:

1. For each topic t, and for each dimension of embedding
   (a) Draw μ_t^u, λ_t^u from NormalGamma(τ^u)
   (b) Draw μ_t^q, λ_t^q from NormalGamma(τ^q)
   (c) Draw μ_t^a, λ_t^a from NormalGamma(τ^a)
2. For each document D
   (a) Draw a multinomial distribution θ from Dir(α)
   (b) Form each answer word w^a in d
      i. Draw a topic z^a from Multi(θ)
      ii. For each dimension of the embedding of w^a, draw f^a from $N(\mu_z^a, \lambda_z^a)$
   (c) Draw a topic y uniformly from all z^a
   (d) For each dimension of the embedding of user u, draw f^u from $N(\mu_z^u, \lambda_z^u)$
   (e) Form each question word w^q in d
      i. Draw a topic z^q from Multi(θ)
      ii. For each dimension of the embedding of w^q, draw f^q from $N(\mu_z^q, \lambda_z^q)$

where notations with superscript $\{u\ q\ a\}$ denote parameters defined for user, question and answer respectively. Particularly, during the model learning process, we assume the prior (τ^u, τ^q, τ^a) distributions follow symmetric NormalGamma, which is the conjugate prior for Gaussian distribution.

4.2.2 Model Inference

Exact inference is often intractable in many topic models and appropriate methods must be used, such as variational inference [85] and Gibbs sampling [86]. Let hyper-parameters $\{\alpha, \tau^u, \tau^a, \tau^q\}$ be denoted as Ψ, and hidden variables $\{\theta, \mu^u, \lambda^u, \mu^q, \lambda^q, \mu^a, \lambda^a\}$ as Φ and we need to estimate the latent variables conditioned on the observed variables, namely $p(z^q, z^a, y|f^u, f^q, f^a, \Psi, \Phi)$. We employ collapsed Gibbs sampling method to obtain samples of latent variables and estimate unknown parameters $\{\theta, \mu^u, \lambda^u, \mu^q, \lambda^q, \mu^a, \lambda^a\}$ in the proposed MMSE. In a Gibbs sampler, one iteratively samples new assignments of latent variables by drawing from the distributions conditions on the previous state of the model. We list the update rules for the latent variables $\{y, z^a, z^q\}$ as follows:

$$p(y_d|y_{-d}, z, f^u, f^q, f^a) \propto (n_d^t + l) \prod_{e=1}^{E^u} G'(f^u, y, t, e, \tau^u, d). \tag{4.1}$$

$$p(z_{dm}^a = t|z_{-dm}^a, z^q, y, f^u, f^q, f^a) \propto (n_{dm}^{y_d} + l)(n_{dm}^t + \alpha_t) \prod_{e=1}^{E^w} G'(f^a, z^a, t, e, \tau^a, dm). \tag{4.2}$$

$$p(z_{dh}^q = t|z_{-dh}^q, z^a, y, f^u, f^q, f^a) \propto (n_{dh}^t + \alpha_t) \prod_{e=1}^{E^w} G'(f^q, z^q, t, e, \tau^q, dh). \tag{4.3}$$

where the superscript $-d$ denotes a counting variable that excludes the user in document d, $-dm$ and $-dh$ means ruling out m-th answer word and h-th question word in a document d respectively. z_{dm}^a is the topic of the m-th answer word in document d, and z_{dh}^q is the topic of the h-th question word in document d. Similarly, n_{dm}^t is the number of answer words assigned to topic t in document d, and n_{dh}^t is the number of question words assigned to topic t in document d. $\tau = \{\alpha_0, \beta_0, k_0, \mu_0\}$ are the hyperparameter of the NormalGamma distribution. To eliminate zero counts, we use Laplace smoothing parameter l, which can simply add smoothing one to each count.

The function $G'(\cdot)$ is defined as follows:

$$G'(f, y, t, e, \tau, d) = \frac{\Gamma(\alpha_n)}{\Gamma(\alpha_{n'})} \frac{\beta_{n'}^{\alpha_{n'}}}{\beta_n^{\alpha_n}} (\frac{k_{n'}}{k_n})^{\frac{1}{2}} \frac{(2\pi)^{-n/2}}{(2\pi)^{-n'/2}}. \tag{4.4}$$

with

$$
\begin{aligned}
a_n &= a_0 + n/2,\ k_n = k_0 + n,\ \mu_n = \frac{k_0\mu_0 + n\bar{x}}{k_0 + n}, \\
\beta_n &= \beta_0 + \frac{1}{2}\sum_{i=1}^{n_t}(x_i - \bar{x})^2 + \frac{k_{0n}(\bar{x} - \mu_0)^2}{2(k_0 + n)}.
\end{aligned}
\tag{4.5}
$$

where n denotes the times of ith dimension in vector embedding assigned to topic t; x denotes the concatenated vector of the e-th dimension of f_i with $y_i = t$; $n' = n - n_d$, n_d denotes the number of f which satisfies $y = t$.

The conditional probabilities (4.1) can also be written as two terms p_{global} and p_{local}:

$$
p(y_d|y_{-d}, z, f^u, f^q, f^a) \propto (n_{dm}^t + l)\prod_{e=1}^{E^u} G'(f^u, y, t, e, \tau^u, d) \triangleq p_{global} * p_{local}. \tag{4.6}
$$

Das et al.[84] embed word types into the space of Gaussian distributions, and learn the embedding directly in that space, which demonstrated the effectiveness of Gaussian distribution on the modeling of word embedding. As we can see, p_{local} is related to the second part of (4.1) and depict the semantic embedding in local view. The first part of (4.1) $n_u^t + l$ is the statistic count of topic number, which can model the global topical context of the data. In fact, the conditional probabilities may largely depend on p_{local}, due to the high dimension of E^u. As a result, we cannot make full use of the global topic distribution information. To address this problem, we introduce a balance weight parameter c to control the weights of various parts. Then formula (4.1)can be written as :

$$
p(y_d|y_{-d}, z, f^u, f^q, f^a) \propto (n_{dm}^t + l)^c \prod_{e=1}^{E^u} G'(f^u, y, t, e, \tau^u, d). \tag{4.7}
$$

We can balance p_{local} and p_{global} by adjusting the weight c. When $c = 1$, it is the same with the original form in equation (4.1). Similarly, the conditional probabilities of the latent topic equation (4.2) (4.3) can also be modified by introducing this balance weight.

4.2.3 Parameter Update

After finishing the Gibbs sampling training, we can estimate the parameters $\theta, \mu^u, \lambda^u, \mu^q, \lambda^q, \mu^a, \lambda^a$ just like [86, 79]. Therefore, we have:

$$\begin{aligned}
\theta_d^t &= \frac{n_a^t + n_q^t + \alpha_t}{\sum_{t=1}^{t}(n_a^t + n_q^t + \alpha_t)}, \\
\mu_t &= \frac{k_0\mu_0 + n\bar{x}}{k_0 + n}, \\
\lambda_t &= \frac{\alpha_0 + n/2}{\beta_0 + \frac{1}{2}\sum_i (x_i - \bar{x})^2 + \frac{k_0 n(\bar{x}-\mu_0)^2}{2(k_0+n)}}.
\end{aligned} \tag{4.8}$$

We update the embedding during inference to fine-tuning pre-trained word embedding and user embedding for a task-specific representation with both global and local semantic. We define the objective functions as the log-likelihood of embedding representation given certain hidden parameters.

For simplicity, we first consider about one dimension of user embedding in document f_{de} as variables x. Our sample is made up of the first n terms of an IID sequence $\{X_n\}$ of Gaussian random variables having mean μ and variance λ. The probability density function of a generic term of the sequence is:

$$G_X(x_i) = \frac{1}{\sqrt{2\pi}}\sqrt{\lambda}e^{-\frac{\lambda}{2}(x_i-\mu)^2}. \tag{4.9}$$

The mean μ and the variance λ are the two parameters that need to be estimated.

1. Given the assumption that the observations from the sample are IID, the likelihood function can be written as:

$$\begin{aligned}
L(\mu, \lambda; x_1, x, \ldots, x_n) &= \prod_{i=1}^{n} G_X(x_i; \mu, \lambda) \\
&= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}}\sqrt{\lambda}e^{-\frac{\lambda}{2}(x_i-\mu)^2} \\
&= (\frac{\lambda}{2\pi})^{n/2} e^{-\frac{\lambda}{2}\sum_{i=1}^{n}(x_i-\mu)^2}.
\end{aligned} \tag{4.10}$$

2. By taking the natural logarithm of the likelihood function, we get the log-likelihood function:

$$\begin{aligned}
l(\mu, \lambda, ; x_1, x, \ldots, x_n) &= \ln(L(\mu, \lambda; x_1, x, \ldots, x_n)) \\
&= \ln\left((\frac{\lambda}{2\pi})^{n/2} e^{-\frac{\lambda}{2}\sum_{i=1}^{n}(x_i-\mu)^2}\right) \\
&= \ln\left((\frac{\lambda}{2\pi})^{n/2}\right) + \ln\left(e^{-\frac{\lambda}{2}\sum_{i=1}^{n}(x_i-\mu)^2}\right) \\
&= -\frac{n}{2}\ln\left(\frac{2\pi}{\lambda}\right) + \sum_{i=1}^{n}(-\frac{\lambda}{2})(x_i-\mu)^2 \\
&= -\frac{n}{2}\ln(2\pi) + \frac{n}{2}\ln(\lambda) + \sum_{i=1}^{n}(-\frac{\lambda}{2})(x_i-\mu)^2.
\end{aligned} \tag{4.11}$$

3. The first two terms of log-likelihood function for single embedding dimension are constant number. After omitting the constant term, log-likelihood function can be writtern as:

$$l(\mu, \lambda, ; x_1, x, \ldots, x_n) = \sum_{i=1}^{n} (-\frac{\lambda}{2})(x_i - \mu)^2. \tag{4.12}$$

Let W be the number of word embedding. We write the log likelihood of the all the data given the model parameters as:

$$\begin{aligned} L = &\sum_{s=1}^{S}\sum_{t=1}^{T}\sum_{e=1}^{E^u}(-\frac{\lambda_{te}^u}{2})(f_{se}^u - \mu_{te}^u)^2 + \sum_{w=1}^{W}\sum_{t=1}^{T} n_w^t \sum_{e=1}^{E^w}(-\frac{\lambda_{te}^q}{2})(f_{de}^q - \mu_{te}^q)^2 \\ &+ \sum_{w=1}^{W}\sum_{t=1}^{T} n_w^t \sum_{e=1}^{E^w}(-\frac{\lambda_{te}^a}{2})(f_{de}^a - \mu_{te}^a)^2. \end{aligned} \tag{4.13}$$

where n_w^t is the number of embedding with topic t.

4. We employ gradient ascent to maximize the log likelihood by updating the embeddings f^u, f^a and f^q. The gradients are computed as

$$\frac{\partial L}{\partial f_{se}^u} = \sum_{t=1}^{T} -\lambda_{te}^u (f_{se}^u - \mu_{te}^u). \tag{4.14}$$

$$\frac{\partial L}{\partial f_{we}^q} = \sum_{t=1}^{T} n_w^t (-\lambda_{te}^q)(f_{te}^q - \mu_{te}^q). \tag{4.15}$$

$$\frac{\partial L}{\partial f_{we}^a} = \sum_{t=1}^{T} n_w^t (-\lambda_{te}^a)(f_{te}^a - \mu_{te}^a). \tag{4.16}$$

The word embedding f^a and f^q share the same vocabulary, and they update in turn. To eliminate scale difference between user embedding and word embedding, the resulting embedding of f^q, f^a, f^u are then normalized by:

$$f \qquad \frac{f}{\parallel f \parallel_2}. \tag{4.17}$$

The model training procedure is summarised in Algorithm 1. With the conditional distributions and parameter update above, we can construct a Markov chain to learn our model with Gibbs sampling. The Gibbs sampling algorithm runs over the three periods: initialisation, burn-in and sampling. (1) We first initialise the algorithm by randomly assigning a topic to y, z^q, z^a with uniform distribution. (2) And then we finish the burn-in stage in t_b iterations based on the Markov chain, as outlined in Algorithm 1. In the burn-in stage, to eliminate the impact of the initial value of the topic implicit variable on the model, we do not update the model

Algorithm 1: Model Training

Input : Hyperparameter of model τ^u, τ^q, τ^q;Initial embedding representation f^u, f^q, f^a; Iteration times of burn-in t_b; Maximum iteration times t_m; Iteration times of hidden topic t_l; Iteration time of parameter updating t_p

Output: Hidden topic y, z^q, z^a; Model parameter $\lambda^u, \lambda^q, \lambda^a, \mu^u, \mu^q, \mu^a, \theta$; Updated embedding representation f^u, f^q, f^a

// Initialization

Random sample topic for y, z^a, z^q

// Burn-in

for t 1 *to* t_b **do**
    foreach *Q/A text word hidden topic* z^q, z^a **do**
        Sample topic z^q, z^a with equation 4.2 and 4.3 respectively
    foreach *User embedding topic* y **do**
        Sample topic y with equation 4.1

// Sampling

for t 1 *to* t_m **do**
    for $t' \leftarrow 1$ *to* t_l **do**
        foreach *Q/A text word hidden topic* z^q, z^a **do**
            Sample topic z^q, z^a with equation 4.2 and 4.3 respectively
        foreach *User embedding topic* y **do**
            Sample topic y with equation 4.1
        if t_p *iterations have occurred since the last time parameter was read in* **then**
            Calculate the parameter with equation 4.8
            Average the current parameter and last one
    Embedding updating(Cf.Algorithm 2)

Algorithm 2: Embedding updating

Input : Model parameter $\lambda^u, \lambda^q, \lambda^a, \mu^u, \mu^q, \mu^a, \theta$; Old embedding representation f^u, f^q, f^a; Iteration time of updating t_e; Initial learning rate of user embedding lr^u;Initial learning rate of word embedding lr^w; learning rate decay *decay*

Output: New embedding representation f^u, f^q, f^a

```
for t  1 to t_e do
    llh_old ← current log-likelihood of MMSE
    foreach User embedding topic f^y do
        calculate gradient g with equation 4.14
        f^u ← f^u + lr^u · g
    foreach User embedding topic f^q do
        calculate gradient g with equation 4.15
        f^q ← f^q + lr^w · g
    foreach User embedding topic f^a do
        calculate gradient g with equation 4.16
        f^a  f^a + lr^w · g
    llh_new  updated log-likelihood
    if llh_new ≥ llh_old then
        accept the embedding update
    else
        refuse the embedding update
        lr^u  lr^u · decay
        lr^w  lr^w · decay
return f^u, f^q, f^a
```

parameters or embedding representations. (3) To obtain the resulting model parameters from a Gibbs sampler, several approaches exist. One is to just use only one read out, another is to average a number of samples which we used, and often it is desirable to leave an interval of t_p iteration between subsequent read-outs to obtain decorrelated states of the Markov chain. This interval is often called thinning interval or sampling lag.

In Algorithm 2, we give the detailed procedure of embedding update for each embedding representation. When training the topic model, it is often useful to reduce learning rate as the training progresses. Every time before the gradient decent, we first calculate the log-likelihood of the model, and then calculate the log-likelihood after iteration. If the log-likelihood increase, it means that the learning rate is appropriate, and we adopt the embedding expression after the gradient descent. If the log-likelihood seems to drop, indicating that the current learning rate is too high, we multiply the learning rate by a decay, and abandon the update of the current iteration's embedding representation. We set the learning rate or embedding update $lr^u = 1\text{e}{-3}, lr^w = 1\text{e}{-5}$, and the decay is set to 0.5.

4.3 Question Answering

In this Section, we introduce how to leverage our semantic embedding model for question answering task. We consider creating a retrieval model with the learned se-

mantic embedding. There have been some successful attempts in retrieval task to use a more general representation method, namely Bag-of-Word-Embedding (BoWE), where both question and answer can be represented by variable length sets of word embeddings. Queried questions and the existing answers represented by BoWE, which provides a better foundation for semantic level matching than previous Bag-of-Words (BoW) method. A popular strategy for using BoWE in IR referred to as the Word Mover's Distance (WMD) [87], which estimates similarity between pairs of documents by estimating the minimum cumulative distance in the embedding space that words from a document need to travel to match words from a second document. In practice, however, this strategy is infeasible, since we have to calculate the distance with each word in the vocabulary. The time complexity is $O(n^3 \log n)$, where n denotes the number of unique words in the documents. It is too expensive for online retrieval and ranking.

We borrow an idea form Non-linear Word Transportation (NWT) [88], and consider reducing the computational complexity by pruning the document nodes and corresponding edge if the document words are too distant from the query words. Specifically, suppose we learned a word embedding matrix $W \in \mathbb{R}^{K \times |V|}$ for a vocabulary with finite $|V|$ word by previous proposed MMSE, where the i-th column, $w_i \in \mathbb{R}^K$, represents the embedding of the i-th word in the K-dimensional space. Both question and answer are represented as BoWE. We denote the answer text as $A = \{(w_1^a, tf_1), ..., (w_n^a, tf_n)\}$ where tf_i denotes the term frequency, and similarly the queried question as $Q = \{(w_1^q, qtf_1), ..., (w_n^q, qtf_n)\}$ where qtf_i denotes the term frequency of j-th word in the question. In the view of transportation view, NWT assume each exiting answer has fixed information capacity, while query question has unlimited capacity to accumulate as much relevance information from an answer text. The information gain of transporting denote as "profit", and the total profit on each query question should obey the law of diminishing marginal returns. Finally, the target of NWT is to find the answer that can bring the maximum net returns for a given query.

Based on the above idea, we estimate relevance between a query question and an answer by finding a set of optimal flows $F = \{f_{ij}\}$ that satisfy

$$\begin{aligned} \max \quad & \sum_{j \in Q} \log \sum_{i \in A} f_{ij} r_{ij}, \\ \text{subject to :} \quad & f_{ij} \geq 0 \qquad \forall i \in A, \forall j \in Q, \\ & \sum_{j \in Q} f_{ij} = c_i \qquad \forall i \in A. \end{aligned} \tag{4.18}$$

where f_{ij} denotes how much capacity of the i-th answer text word flow to j-th query question word, r_{ij} denotes corresponding transportation profit, and c_i denotes the information capacity of the i-th answer text word. To alleviate the bias problem of varying lengths in IR, here we define the document word capacity c_i using Bayesian smoothing with Dirichlet priors:

Dataset	**Quora**	**Zhihu**
Question Number	444,138	73,619
Answer Number	887,771	421,029
User Number	95,951	36,428

Table 4.2 : Statistics of the two CQA datasets

$$c_i = \frac{tf_i + \mu \frac{cf_I}{|C|}}{|D| + \mu}. \tag{4.19}$$

A straightforward idea to define transportation profit is the semantic closeness between two words, such as cosine similarity between word embedding.

$$r_{ij} = \widehat{cos}(w_i^a, w_j^q) = \max(cos(w_i^a, w_j^q), 0), \forall i \in A, \forall j \in Q. \tag{4.20}$$

The truncated cosine similarity $\widehat{cos}(w_i^a, w_j^q)$ is used to avoid negative profit. However, simply using cosine similarity as the transportation profit would over-emphasize the importance of semantic matching. Following the literature [88], we introduce a matching risk parameter α to control the profit gap between exact matching and semantic matching.

$$r_{ij} = \widehat{cos}(w_i^a, w_j^q)^{\alpha}. \tag{4.21}$$

Since the risk of matching semantically related words highly depend on the discriminative power of word, here we simply define the risk parameter as a function of *idf* and obtain the following profit definition.

$$r_{ij} = \widehat{cos}(w_i^a, w_j^q)^{g(idf_i)} \quad \forall i \in A, \forall j \in Q. \tag{4.22}$$

where

$$g(idf_i) = idf_i + b, \quad idf_i = \frac{N - df_i + 0.5}{df_i + 0.5}. \tag{4.23}$$

In [88] df_j denotes the document frequency of the j-th query word, N denotes the total number of documents in the corpus, and b is a free parameter denoting the default offset of the risk.

4.4 Experiments

In this section, we present the experiments on two popularly used question answering datasets to show the effectiveness of the proposed method.

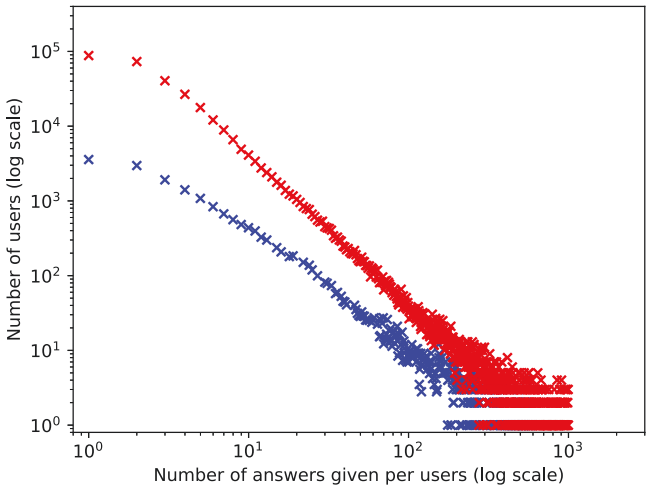


Figure 4.4 : Distribution of number of answers given per user. A small fraction of users answer a lot of questions while many users answer a few number of questions.

4.4.1 Datasets and Evaluation

The two datasets are Quora in English and zhihu in Chinese. Table 4.2 shows the details of the two datasets.

- **Quora** For English, we use a publicly available Question-Answering data [89] sets which is obtained from a popular question answering site, Quora. The dataset contains 444,138 questions, 95,915 users and 887,771 answers from Quora, together with user's following relationship in Twitter.
- **Zhihu** For Chinese, we use a crawler to collect the question-answer pairs and user relationship information from a popular Chines question answering site Zhihu. Chinese sentences are segmented into word in advance. According to the Weibo IDs extracted from the Zhihu user account, we have crawled their follower relationship from Weibo's social network (Limited by weibo API, only 200 followers at most can be obtained for single queried ID).

Due to the nature of the community, the contribution of each user is different based on their interests and availability. As each user directly connects to their answers, we plots the distribution of number of answers given per user in Figure 4.4. We observe user answer distribution is a power-law distribution, which means the answers for most users are relatively small.

For the ground truth, we generate (q, a^+, a^-), where the (q, a^+) is original question/answer pair, and y^- is randomly selected answers. In our experiment, a set of candidate answers is created with size 10 (one positive + nine negative) for each question in the testing. All models? parameters are learned from the training question answer pairs data. The hyperparameters are tuned on a validation set (as

part of the training set). For all English data, we remove stop words and conduct stemming, and for all Chinese data, we conduct Chinese word segmentation.

Since question answering is similar to the task of question answer retrieval, we use nNDCG and P@N [90] on random answers pool with size 10 (one positive + nine negative) to measure the performance of different retrieval models. The premise of nDCG is that highly relevant documents appearing lower in a ranking list should be penalised as the graded relevance value is reduced logarithmically proportional to the position of the result. nDCG accumulated at a particular rank position p is defined as:

$$nDCG = \frac{DCG}{IDCG}, \quad DCG = rel_1 + \sum_{i=2}^{\pi_i} \frac{rel_i}{log_2(i+1)}.$$

where the one positive answer $rel = 1$, for other answers $rel = 0$. Note that in a perfect ranking algorithm, the NCG will be the same as the IDCG producing an nDCG of 1.0. All nDCG calculations are then relative values on the interval 0.0 to 1.0 and so are cross-query comparable.[91, 90].

P@N This criterion is the fraction of the top N retrieved questions that are relevant to the queried questions, given by

$$P@N = \frac{1}{|Q|} \sum_{q \in Q} \frac{N_{q,N}}{N}.$$

where $N_{q,N}$ denotes the number of relevant questions among the top N returned rank list for question q. We use P@1 for evaluation in our experiment.

4.4.2 Baselines

We take the BOW based traditional information retrieval models such as BM25 and LM as the baseline models. In addition to that, we also compared our model with several state-of-art BoWE based method. The detailed descriptions of these methods are listed as follows.

- **BoW** is a simple representation method used in natural language processing and information retrieval. In our experiment, questions and answers are represented by BoW feature and then the similarity between them is calculated to rank the candidate answers for each query question.
- **BM25** [92] is a classical probabilistic IR model that consider the number of occurrences of each query term in the document (term-frequency) and the corresponding inverse document frequency of the same terms in the full collection.
- **LM** (Language Model) [93] approach build a probabilistic language model from each answer, and ranks answers based on the probability of the model generating the question.
- **LDA** [85] is a generative model that seeks to discover the global latent semantic embedding of question and answer.

- **DeepWalk** [1] learn the semantic embedding of CQA data merely utilising the structure information from the user social network.
- **LDA+LM** Language model with LDA smooth feature.
- **Doc2Vec** [94] provide document-level low-dimension embedding for question and answer text.
- **GLM** [95] (Generalized Language Model) use the word similarity in the Skip-gram embedding space as a way to estimate term transformation probabilities in a language modeling setting for retrieval.
- **CNTN** [74] (Convolutional Neural Tensor Network) use CNN to build the embedding representation for each question and answer and then calculate their semantic matching score into a single model.
- **LDA + NWT** We use the low-dimension topic distribution as the representation of question and answer word.
- **Skip-gram + NWT** learn a relevance model by extracting features from Skip-gram embedding method in addition to corpus statistics such as inverse document frequency.
- **TWE + NWT** Topic Word Embedding [96] trains a topic model and word embedding on the same corpus, and then represents question and answer by concatenating average topic embedding and average word embedding.
- **MMSE-sim + NWT** In the previous, we demonstrate that the utilisation of social relationship can mitigate the sparsity problem in CQA tasks. In order to verify this assumption, we evaluate a simplified version of MMSE without the user information.

4.4.3 Parameter Setting

We tune parameters for different models based on their performance on nDCG. In our experiment, the following three representative methods are selected to train the semantic embedding: Skip-gram (baseline), TWE and MMSE. We set the count of negative samples in 4; the context window size in 5; the learning rate is initialised as 0.03 and is set to decrease linearly so that it approached zero at the end of training [77]. LDA uses the specified 26 topics on Quora dataset, as this is the optimal topic number according to [97]. For Zhihu dataset, we set topic number in 30. For NWT, we tune the smoothing parameter μ in (100, 200, ..., 1000) and the offset b in (0, 1, 2, 3) suggested by the original papers [88]. We empirically set the hyperparameter of MMSE $\mu_0 = 0, k_0 = 1\text{e}{-5}, \beta_0 = 1, \alpha_0 = 1\text{e}{-3}, T = 300, \alpha_0 = 0.3$.

4.4.4 Performance Evaluations and Comparisons

As mentioned previously, we argue that the utilisation of both local and global semantic context for multi-modal data in CQA can effectively mitigate the lexical gap problem. To investigate the capability of comprehensive semantic modeling, we compare its performance with existing solely topic model based methods and word embedding based models. Besides, the social network information is crucial to alleviate the problem of data sparsity. Thus, the content information is utilised

Model	Objectives	Quora		Zhihu	
		nDCG	P@1	nDCG	P@1
BoW	—	0.714	0.487	0.683	0.473
BM25	—	0.738	0.541	0.721	0.485
LM	—	0.783	0.547	0.757	0.504
LDA	global	0.749	0.493	0.719	0.476
DeepWalk	user	0.666	0.458	0.639	0.446
LDA+LM	global	0.804	0.549	0.752	0.517
Doc2vec	local	0.761	0.521	0.723	0.494
GLM	local	0.821	0.579	0.773	0.532
CNTN	local	0.849	0.614	0.847	0.624
LDA+NWT	global	0.803	0.587	0.78	0.527
Skip-gram+NWT	local	0.839	0.601	0.817	0.568
TWE+NWT	local+global	0.852	0.613	0.825	0.588
MMSE-sim+NWT	local+global	0.871	0.635	0.837	0.623
MMSE+NWT	**local+global+user**	**0.902**	**0.679**	**0.862**	**0.673**

Table 4.3 : Evaluation Results on Quora and Zhihu Data

by all the method except DeepWalk, and the social information is introduced by DeepWalk and our proposed MMSE.

The question answering retrieval results of each method on Quora and Zhihu are reported in Table 4.3. The highest scores were highlighted with boldface. Based on these results, we have the following observations:

1. For language model, semantic embedding based language model (local embedding view for GLM, global topical view for LDA+LM) outperform the original LM, BoW and BM25 methods and yield large improvements in most cases, which demonstrates that both local and global views based method can effectively address the word lexical gap problem.

2. Local view based method Doc2vec, GLM, Skip-gram+NWT obtain better performances than global view based method LDA, LDA + LM, LDA + NWT respectively. These results indicate that the local semantic embedding is more effective at capturing semantic relations of words than that of global topic approach.

3. We also note that CNTN achieve comparable performance with the local semantic embedding, which indicates the effectiveness of the deep learning method for question answering with the help of large scale of dataset.

4. TWE+NWT and MMSE-sim+NWT outperform the LDA+NWT and Skip-gram+NWT respectively, which tells us that semantic embedding with multi-view can obtain more comprehensive semantic relation between words.

	Answer1: The estimated resident **population** for is 4,605,992, one-fifth of **Australia**'s population. And it is famous for **beach** and **sunshine**. Hope you enjoy a great **trip**.	**Answer2**: In fact, in some cases, political centers set up separately from financial centers. Examples include **Germany (Berlin vs. Frankfurt), US (Washington vs. New York), Australia (Canberra vs Sydney), China (Beijing vs. Shanghai)**.	**Answer3**: The popular view of small-town Wexford in **Ireland** is beaches, **sunshine** and **opera**. Wexford hosts the internationally recognized **Opera Festival** every October in The National **Opera House**.
LDA+NWT	Rank 3	Rank 7	Rank 1
Skip-gram+NWT	Rank 4	Rank 1	Rank 6
MMSE+NWT	Rank 1	Rank 5	Rank 7

Table 4.4 : Retrieval Results for "What should I know before **traveling** to **Sydney**?"

5. DeepWalk performs worse than other models in most cases, which indicates that the content analysis plays a more important role than the simple utilization of social information in CQA tasks.

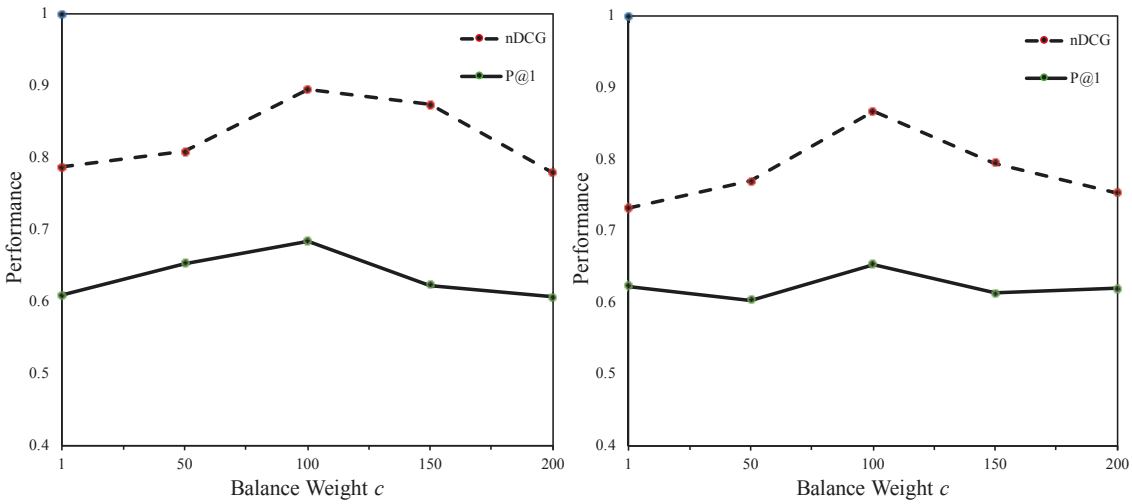
6. Since our method MMSE+NWT achieve much better performance than MMSE-sim+NWT and TWE+NWT, which shows that it is useful to model and exploit the multi-modal information to mitigate the sparsity problem in CQA tasks. We conclude that user social information can reinforce the effect of semantic embedding, and user similarity calculated in the vector space is very useful for identifying the semantic similarities between questions.

4.4.5 Impact of Parameter Values and Case Study

To get a better understanding how the multi-modal multi-view property benefit the question answering task, Table 4.4 gives part of the results of an example question "What should I know before traveling to Sydney?" The semantic embedding modeled by different methods for query words "traveling" and "Sydney" indicates different semantic similarity with words in answers. The two baseline can be easily

dataset	Dimension	nDCG	P@1
Quora	50	0.869	0.648
	100	0.883	0.662
	300	**0.902**	**0.679**
	500	0.889	0.654
Zhihu	50	0.831	0.635
	100	0.835	0.639
	300	**0.862**	**0.673**
	500	0.842	0.645

Table 4.5 : Performance comparison of MMSE over different dimensionality of word embedding



(a) Quora. (b) Zhihu.

Figure 4.5 : Effect of varying the balance weight c.

misled by some distinctive words in single-view. In contrast, our proposed MMSE can give consideration to both local and global similarity-based words.

In our approach, there are two essential parameters, which are the dimension of semantic embedding and the balance weight c. Here we evaluate performance results on the dataset using semantic embedding trained by MMSE model. As shown in Table 4.5, by setting dimensions on 50, 100, 300, and 500 dimensions, the performance first increases and then slightly drops as the embedding dimension size increasing. As we know, different dimensionality of embedding provides different levels of granularity of semantic similarity, which may also require different amounts of training data. Generally, larger dimension size will give better quality and may need more data to prevent overfitting problem. Our results suggest that 300 dimensions is sufficient for learning semantic embedding on both datasets.

Besides, balance weight c used in MMSE also plays an important role in producing high-quality semantic embedding. Overemphasizing the weight of the original objective of p_{local} may result in the weakened influence of global topical context, while putting too large weight on p_{globle} may hurt the generality of learned local word embedding. Based on our experience, it is a better way to decode the objective balance weight based on the scale of their respective derivatives during optimization. Therefore, we carry out an experiment on the validation set to determine the best value among $\{1, 50, 100, 150, 200\}$. Figure 4.5 shows the evolution of for the balance weight c on Quora data and Zhihu data. For all these plots, the two group methods are displayed. We can see that the best value achieved when setting the balance weight around 100.

4.5 Experiment for expert finding

In CQA, one of central issues is to find users with expertise and willingness to answer the given questions. Expert finding provides a platform to connect questions with experts who can contribute quality answers[98]. For example, CQA can help to find a mathematician for a chef with a math problem. At the same time, cooking tips from the chef will be returned to the mathematician if necessary. The goal of expert finding is to return a ranked list of experts with expertise on a given question. An essential part of an expert finding task is the ability to model the expertise of a user based on her answering history.

The textual contents of questions are fed into the corresponding CNN to calculate the latent representation with the leaned word embedding f^a, f^q[99]. Each question q_i is denoted by a d dimensional word vector. We then denote the collection of questions by $Q = \{q_1, ..., q_n\} \in R^{d \times n}$ where n is the total number of the questions. We denote the set of user embeddings by $U = \{u_1, u_2, ..., u_m\} \in R^{d \times m}$where u_i is the embedding vector for the latent expertise of the i-th user with the learned user embedding f^u

For ground truth, we consider all the answerers for each question as the target user set, and their received thumbs-up/down as the ground truth rating scores. We use the relative quality rank to model the performance of users for answering the questions, which is in the form of triplet constraints. Unlike the previous studies, our method MMSE learns the user embedding and word embedding from the proposed CQA network. We denote a triplet constraint by the ordered tuple (j, i, k), meaning that "the i-th user obtains more votes than the k-th user for answering the j-th question". Let $Tri = \{(j, i, k)\}$ denote the set of triplet constraints obtained from the community votes for a set of m users answering n different questions. More formally, we aim to learn the ranking metric function that for any $(j, i, k) \in Tri$, the inequality holds: $f_{u_j}(q_j) > f_{u_k}(q_j) \iff q_i^T u_j > q_i^T u_k$. We compare our proposed method with other popular expert finding algorithms in CQA systems as follows:

- **ExpertsRank** algorithm[100]: uses question ask-answer relation to construct the graph of users, and then finds the experts with link structure analysis based on PageRank algorithm.

Methods	nDCG		
	60%	70%	80%
ExpertsRank	0.6702	0.6926	0.6948
AuthorityRank	0.6723	0.7014	0.7108
DRM	0.6585	0.6646	0.6707
DeepWalk	0.6238	0.6296	0.6381
TSPM	0.6316	0.645	0.6627
MMSE	**0.6937**	**0.7428**	**0.7582**

Table 4.6 : Experimental results on nDCG with different proportions of Quora data for training

Methods	P@1		
	60%	70%	80%
ExpertsRank	0.4223	0.4637	0.4859
AuthorityRank	0.4441	0.4785	0.4904
DRM	0.4183	0.4242	0.4311
DeepWalk	0.3659	0.3717	0.3815
TSPM	0.3781	0.3944	0.4177
MMSE	**0.4827**	**0.5348**	**0.5527**

Table 4.7 : Experimental results on Precision@1 with different proportions of Quora data for training

- **AuthorityRank** algorithm[101]: computes user authority based on the number of provided best answers, which is an in-degree method.
- **DRM** method[102] is a topic-sensitive probabilistic model, which learns question representation via PLSA-based model.
- **DeepWalk** algorithm[1]: learns the embedding of both questions and users based on the network structure.
- **TSPM** algorithm[103]: The TSPM algorithm is a topic sensitive probabilistic method for expert finding in CQA systems, which fits a LDA-based probabilistic model to the question-answering activities.

Experimental results. We summarise our results for expert finding in Table 4.6 and Table 4.7. The evaluation were conducted with 60%, 70% and 80% of data for training. The DeepWalk, AuthorityRank and ExpertsRank methods are based on link analysis of users, which only consider the network structure. While DRM and TSPM methods are based on probabilistic model with question content. These experiments reveal several key points:

- The topic-oriented methods, both TSPM and DRM, outperform the AuthorityRank and ExpertsRank methods, which suggests the effectiveness of the latent user model for the problem of expert finding.
- The supervised methods, AuthorityRank, TSPM, DRM and ExpertsRank outperform the unsupervised DeepWalk method, which suggests that the supervised information such as users' relative quality rank and question contents are critical for the problem.
- In all the cases, our MMSE method achieves the best performance. This fact shows that the ranking metric network learning framework that exploits both deep representation of question contents and users' relative quality rank can further improve the performance of expert finding.

4.6 Conclusions

In this chapter, we propose a novel multi-modal multi-view semantic embedding learning method for community question answer analysis. The proposed MMSE can simultaneously model multi-modal CQA data as well as their corresponding semantic information from both local and global views in a unified and principled way. Experiments conducted on question answering task for both English and Chinese CQA datasets demonstrate the effectiveness of our approaches. In the future, we will evaluate our model on other CQA task and investigate more applications, such as the answers quality evaluation.

Chapter 5

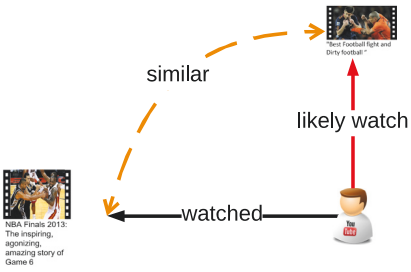
Context-Dependent Propagating-based Video Recommendation in Multimodal Heterogeneous Information Networks

With the emergence of online social networks (OSNs), video recommendation has come to play a crucial role in mitigating the semantic gap between users and videos. Conventional approaches to video recommendation primarily focus on exploiting content features or simple user-video interactions to model the users' preferences. Although these methods have achieved promising results, they fail to model the complex video context interdependency, which is obscure/hidden in heterogeneous auxiliary data from OSNs.

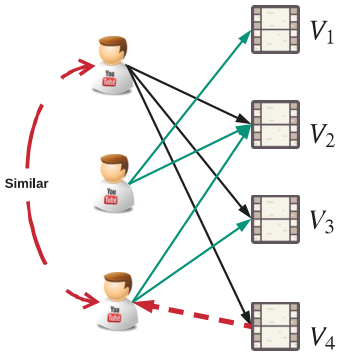
In this chapter, we study the problem of video recommendation in Heterogeneous Information Networks (HINs) due to its excellence in characterizing heterogeneous and complex context information. We propose a Context-Dependent Propagating Recommendation network (CDPRec) to obtain accurate video embedding and capture global context cues among videos in HINs. The CDPRec can iteratively propagate the contexts of a video along links in a graph-structured HIN and explore multiple types of dependencies among the surrounding video nodes. Then, each video is represented as the composition of the multimodal content feature and global dependency structure information using an attention network. The learned video embedding with sequential based recommendation are jointly optimized for the final rating prediction. Experimental results on real-world YouTube video recommendation scenarios demonstrate the effectiveness of the proposed methods compared with strong baselines.

5.1 Introduction

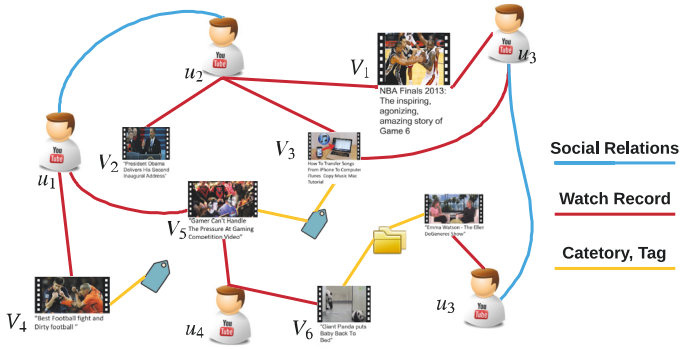
As the amount of data from different platforms grows at an unprecedented rate, video recommendation is becoming increasingly crucial to help users discover personalized content from this ever-growing corpus of data [2, 3]. For example, 500 hours of videos are uploaded to YouTube every minute [4]. It is thus impossible for the users to watch all available videos to identify their interested ones. As a result, there is an enormous demand for video recommendation to provide relevant information tailored to the users' interests or preferences. Furthermore, the rapid emergence of user-generated content (UGC) and online social networks (OSNs) has provided a new rich library of social media content [104, 105, 40], which introduces new challenges and opportunities to the recommendation process. In this chapter, we aim to design an effective and robust video recommendation method for OSNs.



(a) Content-based recommendation



(b) Collaborative filtering based recommendation



(c) General idea to adopt a multimodal HIN for recommendation

Figure 5.1 : (a) An example of content-based recommendation. Only the video content feature is used to model user preference. (b) Collaborative filtering uses simple user-video interaction for video recommendation. (c) Our general idea of introducing a multimodal HIN to model user historical information. We try to encode global video context and multimodal video content feature into the comprehensive video embedding for recommendation purposes.

Early works on video recommendation mainly include content-based methods [106] and collaborative filtering based methods [107]. As shown in Figure 5.1(a), content-based methods recommend a user to watch videos similar to those have been watched before. As shown in Figure 5.1(b), collaborative filtering based methods predict the interests of a user based on the user-video interactions, in the form of a graph/network structure. Content-based methods only calculate the video relevance using multimedia data analysis, which cannot directly reflect the users' general interests. Collaborative filtering based methods suffer from the cold-start problem and the sparsity problem of the user-video interaction matrix [108]. Currently, various types of auxiliary data, such as the user social relationships and video attributes (e.g., category and tag), are becoming increasingly available in OSNs [109]. Given the above limitations of the pure content-based and collaborative filtering based methods, many methods have been further proposed to leverage this auxiliary information to improve the recommendation performance [110, 105, 40, 38, 111]. These methods can be considered hybrid methods for recommendation systems. However, due to the heterogeneity and complexity of social data, it remains challenging for recommendation systems to effectively utilize such auxiliary context information.

Heterogeneous Information Networks (HINs) consist of various patterns of connections among different types of objects, which can reveal hidden interdependency among them [32, 112]. In this chapter, we study the problem of video recommendation in HINs considering their flexibility in exploiting rich relations in various types of auxiliary data. An illustration of a video recommendation problem in an HIN can be found in Figure 5.1(c). Our HIN includes all types of objects (e.g., videos, users and attribute information) and demonstrates different types of relations among objects, such as social relationships and user-attribute informaiton. These complex relations captured in the HINs provide us with a deep and latent perspective, from which to analyse the video interdependency relation among videos. In fact, all these relations can contribute to the recommendation task in an explicit or implicit manner. For example, the relation of two videos can be inferred by 1) "video-user-user-video" and 2) "video-category-video". These meta-paths capture the semantic relations of 1) being watched by friends or 2) belonging to the same genre. Hence, we can find potentially interesting videos based on the intuition that the paths connecting two videos represent the video relations of different semantics. Such an intuition facilitates the inference of user preferences based on video similarity to generate effective recommendations. This concept is known as HIN-based recommendation.

In the context of OSNs, video recommendation via HINs is still an open and challenging problem for two reasons. First, there is a general lack of consideration of how videos propagate through social connections, which can be used to model the complex and deep contextual interdependencies among videos. Nearly all existing models have leveraged the HIN-based structure in a simple manner by considering the local neighbours of each video [36, 113]. Social networks provide a fundamental medium for the diffusion and spread of information to neighbour videos via users' social connection and subsequently to the video neighbours, which forms a hierarchical interdependency among videos [114]. Instead of employing the one-hop local social network structure, how should we capture the influence-propagating process

among videos for better social recommendation performance is an urgent issue that must be investigated.

Second, most of the existing HIN-based recommendations use the path-based similarity without considering the multi-modal video content feature of nodes (such as video title and description). Limited efforts have been devoted to analysing both graph structure and node feature in a unified manner to model the user preference [112]. As shown in Figure 5.1(c), according to the meta-paths "video-user-user-video" (e.g., $V_1 - u_2 - u_1 - V_4$ and $V_1 - u_2 - u_1 - V_5$), video V_1 may have identical similarity to videos V_4 and V_5, since they have been watched by the same friends. However, videos V_4 and V_5 have different styles due to their different text and visual features. This example highlights that the same meta-path that connects a different video pair often carry relations of different semantics. In fact, V_1 and V_4 should be more similar, since they both belong to the category "ball game". The conventional meta-path does not allow a node to have content features [114], so it cannot reveal this subtle difference. To fully exploit paths in HINs for a recommendation, one must capture the semantics of different paths and their distinctive content feature in describing user preferences towards videos.

To exploit HINs for recommendation and overcome the above limitations, we propose a Context-Dependent Propagating Recommendation network, or CDPRec for brevity. Our goal is to explore both multimodal content features and context structure information from a multi-modal HIN to generate high-quality video embeddings for recommendation. The network embedding projects the nodes of a network* into a vector space. Compared to the pure meta-path based similarity, the network embedding approach is more resistant to noisy and sparse data. For a graph structure of the multi-modal HIN, we first extract a homogeneous video graph from the original HIN with a path-based random walk, which can be easier to handle while maintaining their original interdependency structure. For video content of the multi-modal HIN, we learn the shared video representation using a multimodal fusion layer that combines both visual and textual content. The proposed CDPRec has two main operations by which it learns the final video embedding from both graph structure and multimodal video features: (1) Propagation: The CDPRec can iteratively propagate a video's contexts along links in a graph-structured HIN and explore multiple types of dependencies among the surrounding video nodes. The propagating operation stimulates the diffusion of users' preference in a social network structure to discover the video in which she/he has the most hierarchical potential to be interested; (2) Aggregation: We collectively aggregate multimodal features of potential interests for each video node with an attention network, followed by a linear transformation to generate a new representation for a target video. Finally, an RNN-based sequential recommendation layer is seamlessly integrated with the network such that the CDPRec can be trained in an end-to-end manner.

The main contributions of this chapter are as follows:

1. We are the first to propose a multimodal HIN embedding method to simul-

*The terms 'graph' and 'network' have identical meaning here.

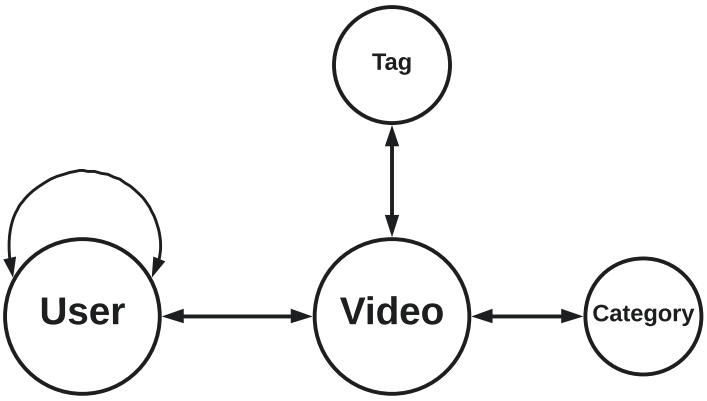


Figure 5.2 : Network schemas of a heterogeneous information network used for our social video recommendation scenario.

taneously uncover both multimodal video content and complex structural information of HINs. Moreover, we propose a meta-path-based random walk strategy to extract a homogeneous video graph and learn the shared video content feature using a multimodal fusion method.

2. We propose a Context-Dependent Propagating Recommendation network, called CDPRec for short. CDPRec is an embedding-based method that automatically discovers the hierarchically ranked potential interesting videos of users along links by iteratively propagating video context in the graph and aggregating them for video embedding.

3. We conduct experiments on a real-world YouTube dataset, the results of which demonstrate the efficacy of the CDPRec over strong baselines. In particular, the proposed model performs well in the cold-start scenario and has potentially good interpretability for the recommendation results.

5.2 Preliminary

The popularity of online social networks has resulted in a large amount of information being made available, such as social relationship and object attribute information (e.g., category and tag). Thus, it is straightforward to find the social network of users to alleviate the sparsity problem of simple user-video interaction. Furthermore, the emergence of tags and categories has allowed video uploaders to conveniently organize their videos, while also allowing the users to find interesting videos according to a given tag or category. Taking YouTube as an example, users may wish to seek out a group of interesting guitar artists by querying the "guitar" tag. We can use this rich auxiliary information to improve the recommendations.

HIN is a powerful tool to organize the above auxiliary information. Users and videos with few prior interactions can be linked together through different types of paths. We set up our recommendation system in the framework of the HIN, which can be defined as follows:

Notations	**Description**
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	a heterogeneous information network (HIN)
$\mathcal{V}$	HIN node set
$\mathcal{E}$	HIN link set
$\mathcal{A}$	HIN node type set
$\mathcal{R}$	HIN link type set
ρ	a meta-path
$G = (V, A, X)$	a homogeneous video graph
V	video node set
V_i	ith video in V
A	adjacency matrix
$e_{ij} \in A$	edge weight between video V_i and V_j
N	number of videos
$N(i)$	neighbor videos for video V_i
T_v	transformation matrix w.r.t. the visual content
T_t	transformation matrix w.r.t. the textual content
v_e	visual content embedding
t_e	textual content embedding
X	multimodal video feature
$x_i \in X$	embedding of video V_i
$N \times F$	dimension of X
X^L	output video embedding in layer L
$N \times F^L$	dimension of X^L
h_t^u	user preference

Table 5.1 : Notations and explanations

Definition 1. **Heterogeneous Information Network (HIN)** [34]. *A HIN is denoted as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which consists of a node-type mapping function $\phi : \mathcal{V} \to \mathcal{A}$ and a link-type mapping function $\varphi : \mathcal{E} \to \mathcal{R}$. $\mathcal{A}$ and $\mathcal{R}$ are the sets of pre-defined node and link types respectively, and satisfy $|\mathcal{A}| + |\mathcal{R}| > 2$.*

Generally, the HIN does not consider the content features on each node. However, real-world video recommendation contains rich multimodal content information on video nodes. Thus, we denote the HINs with multimodal video content features as *multimodal HINs*.

In HIN, the **network schema** is introduced to illustrate the high-level topology of a HIN, which describes the node types and their interaction relations.

Example 1. As shown in Figure 5.2, we present the meta schema of our video recommendation system, which consists of various types of objects (e.g., User (U), Video (V), Tag (T) and Category (C)) and their semantic relations (e.g., 'friends' relation among users, 'watching' relation among users and videos, and 'attribute' relation among videos and tags/categories).

In HIN, nodes can be connected via multiple types of semantic paths, which are defined as **meta-paths**.

Definition 2. **Meta-Path** Given a HIN, meta-path ρ is defined in the form of $\mathcal{A}_1 \xrightarrow{\mathcal{R}_1} \mathcal{A}_2 \xrightarrow{\mathcal{R}_2} \cdots \xrightarrow{\mathcal{R}_l} \mathcal{A}_{l+1}$ (abbreviated as $\mathcal{A}_1\mathcal{A}_2 \cdots \mathcal{A}_{l+1}$), which denotes a composite relation $\mathcal{R}_1 \circ \mathcal{R}_2 \circ \cdots \mathcal{R}_l$ between nodes $\mathcal{A}_1$ and $\mathcal{A}_{l+1}$, where $\circ$ is a composition operator on the relations.

Example 2. There exist multiple specific paths under the meta-path, which is called a path instance denoted by ρ. We can connect two videos with different meta-paths, e.g., "$Video-Category-Video$" (VCV) and "$Video-User-User-Video$" (VUUV). Commonly, different meta-paths can reveal the different interdependency information of two videos. Path "VCV" path illustrates that two videos belong to the same category, while path "VUUV" illustrates two videos watched by two friends.

Most existing HIN-based recommendations rely on meta-path based similarities for recommendation [32, 37], which may result in complex interdependencies among videos in the HINs being missed. In the following part, we propose a new graph embedding-based method for this task, which can mine both complex graph structure and multimodal node feature information hidden in HIN. Notations used throughout the article can be found in Table 5.1.

5.2.1 Framework

The overall framework of the proposed CDPRec is illustrated in Figure 5.3, which consists of four major components: (1) For the graph structure, we extract the homogeneous video graph from the HINs via a meta-path-based random walk (Figure 5.3(a)). This homogeneous video graph is easier to handle while still preserving its original contextual structure; (2) For the multimodal video node feature, we learn

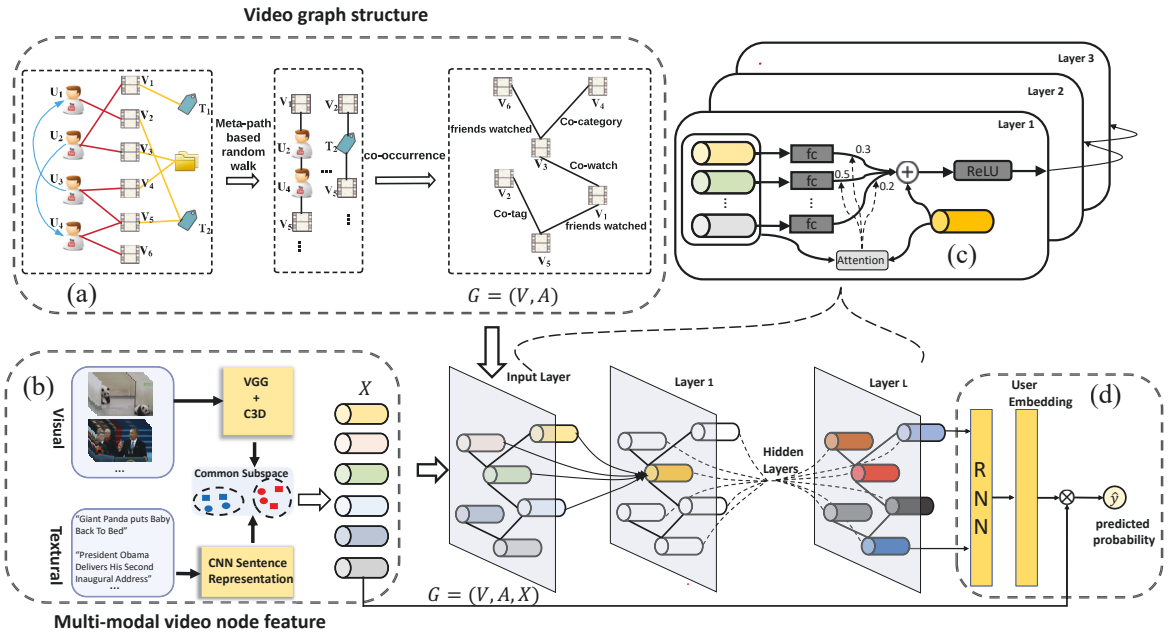


Figure 5.3 : The overall framework CDPRec.

the shared video representation using a multimodal fusion layer that combines both visual and textual content (Figure 5.3(b)). (3) To learn the final video embedding from the above graph structure and multi-modal video node feature, several propagating layers are used to iteratively extend the user's interests along graph links to discover his potential hierarchical interests and aggregate them for embedding learning purposes (Figure 5.3(c)). (4) Finally, we extend the classical RNN-based sequential recommendation by incorporating the learned video embedding to compute the predicted probability for potential videos (Figure 5.3(d)). In the following subsections, we describe each component of the CDPRec in detail.

5.2.2 Construction of video graph from HIN

For the convenience of network embedding learning for video, our first building block is to extract a homogeneous video graph from the HIN. Existing network embedding methods primarily focus on homogeneous networks where the nodes are of the same type, and they cannot be directly applied to heterogeneous networks. Since our main goal is to learn effective video embedding for a recommendation purpose, nodes of other types other than videos are of less interest in our task. Hence, we use a set of meta-path over the original heterogeneous graph to transform the original HIN into a homogeneous video graph, which is easier to handle while still preserving the original contextual structure among videos.

The key to generating a meaningful homogeneous video graph involves designing an effective method to capture the complex semantics reflected in HINs [112]. In general, three steps are required to generate the homogeneous video graph: (1) Inspired by [112], a meta-path-based random walk is adopted to generate a meaningful context sequence; (2) We filter out unrelated nodes to generate a pure video

sequence; (3) Finally, we construct the video graph by considering the co-occurrence of video in sequence and drawing edges among them. In the following, we describe each step in detail.

In this chapter, we design meta-path-based random walks to generate paths that are able to capture both the semantic and original structural correlations between different videos, facilitating the transformation of heterogeneous network structures into a homogeneous video graph. Given a HIN $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a meta-path $\rho : A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots A_t \xrightarrow{R_t} A_{t+1} \cdots \xrightarrow{R_l} A_{l+1}$, the transition probability of next walk is:

$$P(n_{t+1}|n_t, \rho) = \begin{cases} \frac{1}{|N^{(A_{t+1})}(v)|} & (n_t, n_{t+1}) \in \mathcal{E}, \phi(n_{t+1}) = A_{t+1} \\ 0 & (n_t, n_{t+1}) \in \mathcal{E}, \phi(n_{t+1}) \neq A_{t+1} \\ 0 & (n_t, n_{t+1}) \notin \mathcal{E} \end{cases} \tag{5.1}$$

where $n_t \in A_t$ is the t-th node in the random walk sequence, and $N^{(A_{t+1})}(n_t)$ denotes the neighbour node set for n_t with the type of A_{t+1}.

A meta-path-based random walk will iteratively follow the predefined path pattern, which ensures that different types of semantic relationships among video nodes can be properly preserved. For example, in the traditional random walk procedure in Figure 5.3(a), the next step of a walker on node V_1 can be all types of nodes surrounding it: U_2 and T_1. However, under the meta-path **(V-U-U-V)**, the walker is biased to choose U_2 in the next step. Repetitively following the semantics of this path until it reaches the pre-defined length, we can sample a sequence "$V_1 \rightarrow U_2 \rightarrow U_4 \rightarrow V_5 \rightarrow U_3 \rightarrow U_1 \rightarrow V_2$". The meta-path-based random walk strategy ensures that the semantic relationships among different types of nodes can be properly preserved. A fundamental property of a random walk is that in the limit, the long-term average probability of being at a particular node is independent of the start node. Thus, the random walk can initiate from different seed nodes. In this chapter, we use four types of video-related meta-path to generate a node sequence, where the starting type is V. (1) Two video co-occurrences appear in a user's watch list **(V-U-V)**. (2) Two videos have been watched by friends **(V-U-U-V)**. (3) Two videos have the same category **(V-C-V)**. (4) Two videos have the same tag **(V-T-V)**.

Since our main focus is to learn a representation for videos, we filter out the nodes of other types and extract the video sequence based on the node sequence generated above. Thus, the final random walk sequence will contain only video nodes. To construct the homogeneous video graph, we consider the co-occurrence of videos in the sampled paths by setting a slide window interval [115]. Two videos will be connected by an edge if they occur in the same window interval. Then, we embed the video node over this homogeneous video graph. The most significant advantage of this approach is that we can relax the challenge of handling complex multiple node types in the HIN. In addition, this homogeneous video graph can maintain the original relation structure information among videos in the HIN because the links among homogeneous video nodes are essentially extracted from the heterogeneous local neighbour videos. Taking the meta-path V-U-U-V as an example, we can sample a sequence "$V_1 \rightarrow U_2 \rightarrow U_4 \rightarrow V_5 \rightarrow U_3 \rightarrow U_1 \rightarrow V_2$" according to Eq. 5.1.

After the sequence is constructed via a pre-defined meta-path, we discard the nodes with different types from the video. Hence, we obtain the homogeneous video node sequence "$V_1 \rightarrow V_5 \rightarrow V_2$".

In addition to the node structure, we assign a weight to each edge e_{ij} based on the total number of occurrences of the two connected videos. More specifically, the weight of the edge is equal to the frequency of video i transitioning to video j in the entire behaviour, category and interaction history of the user. Thus, the constructed video graph can represent the similarity among different videos based on all user behaviours. The path length for the random walk and the fixed slide window size is empirically set. Finally, the homogeneous video graph structure is defined as $G = (V, A)$, where V is the set of video nodes, and A is the adjacency matrix. Each entry $e_{ij} \in A$ is the weight between video nodes V_i and V_j.

5.2.3 Multimodal video feature representation

In addition to the graph structure, we learn the high-level semantic representation of video content. Commonly used content features for video recommendation are textual features [116, 117], such as video title and description. However, recent user-generated textual descriptions for video may be incomplete and noisy, so the existing approaches fail to generate precise content-based video recommendations in most cases. Therefore, some methods use multimodal video content (e.g., visual and texture content) for video recommendation [118, 119]. These methods mainly use hand-crafted weights to fuse information across different modalities, which may not be able to handle large-scale video data in recommendation systems [120].

The multimodal video feature representation in this chapter aims to learn a joint video representation from textual and visual content by mapping them both into a shared space. Thus, the video content in the HINs can be comprehensively represented, which makes it easy to reveal the user preferences. For the visual content representation $\mathbf{v}$, we extract the $fc6$ layer in VGG19 [121] and the last fully connected layer from C3D [122] for the frame and clip representations, respectively. The sampled frames and clips are processed by mean pooling and concatenated together to generate a single feature vector. The textual content representation $\mathbf{t}$ is extracted from the last layer of a pre-train convolutional text classification model [99], which takes Glove [123] as the word representation input.

We assume that there is a common space across visual and textual modalities, where the similarity between visual and textual contents can be measured. To map these two modalities into a common space, we design two linear mapping functions:

$$\mathbf{v}_e = \mathbf{T}_v \mathbf{v} \quad \text{and} \quad \mathbf{t}_e = \mathbf{T}_t \mathbf{t} \tag{5.2}$$

where $\mathbf{T}_v \in \mathbb{R}^{F \times D_v}$ and $\mathbf{T}_t \in \mathbb{R}^{F \times D_t}$ are the trainable transformation matrices that map the video content and semantic sentences into the common embedding. D_t, D_v, and F are the dimensions of the textual content, visual content, and common multimodal video feature, respectively. Then, we obtain the shared video representation by concatenating them together: $\mathrm{X} = \mathbf{v}_e \oplus \mathbf{t}_e$.

As shown in Figure 5.3(b), we optimize the common embedding by minimizing a pairwise loss as proposed in [124]. We use $score(\mathbf{v}_e, \mathbf{t}_e)$ to measure the similarity score between a visual-text pair. Moreover, μ^+ and ρ^{2+} represent the mean and variance of the matched visual-text pair similarity distribution, while μ^- and ρ^{2-} denote the mean and variance of the non-matching pair similarity distribution. In this chapter, we try to (1) maximize the mean similarity score between non-matching pairs while also (2) minimizing the mean distance and variance of two distributions between matching pairs, which is defined as:

$$Loss = (\rho^{2+} + \rho^{2-}) + \lambda \max(0, \Delta - (\mu^+ - \mu^-)) \tag{5.3}$$

where λ is a term that balances the importance of two terms, and Δ is the max margin between the mean of the matching and non-matching distance distributions. $\mu^+ = \sum_{e=1}^{Q_1} \frac{score(\mathbf{v}_e, \mathbf{t}_e)}{Q_1}$ and $\rho^{2+} = \sum_{e=1}^{Q_1} \frac{score(\mathbf{v}_e, \mathbf{t}_e) - \mu^+}{Q_1}$ when visual feature v_e and text feature t_e belong to the same video. $\mu^- = \sum_{e=1}^{Q_2} \frac{score(\mathbf{v}_e, \mathbf{t}_e)}{Q_2}$ and $\rho^{2-} = \sum_{e=1}^{Q_2} \frac{score(\mathbf{v}_e, \mathbf{t}_e) - \mu^-}{Q_2}$ when visual feature v_e and text feature t_e belong to different videos. We train Eq. 5.3 by the gradient descent with a mini-batch size of 200. Specifically, we sequentially select $Q_1 + Q_2 = 200$ visual-text pairs from the training set for each mini-batch in the experiments. We use random negative sampling to choose the non-matching pair and resample for each epoch.

5.2.4 Context-dependent propagating network

After obtaining the video graph structure and multimodal feature vector of video nodes, we describe a deep neural network suitable for processing the node-feature-guided network embedding method to learn the video embedding. See Figure 5.3(c) for an overview.

Let us revisit the homogeneous video graph $G = (V, A, X)$ constructed above, which provides the following information:

- A set of N videos that constitute the nodes of the homogeneous video graph. Each video node V_i is represented by vector $x_i \in X$ of the video node. Let X be a $N \times F$ matrix representation of the multimodal video content feature representation.
- A set of pairwise relations among all videos, which form the edges among the videos. The weight between videos V_i and V_j is represented by $e_{ij} \in A$. Similarly, let adjacency matrix A be a $N \times N$ matrix that represents the structure of the graph.

For the graph-structured data, we introduce a propagating model to improve the recommendation performance. Intuitively, if we can mine relevant videos based on the user preference in which a user was previously interested, then by following the discovered relevant videos, we can make video recommendations to this user accordingly. Relevant videos from the global neighbour context can be considered the natural extensions of a user's historical interests with respect to the HIN. For

Algorithm 3: Propagating layer with single head

Input: Homogeneous video graph structure $G = (V, A)$; multimodal video feature X; depth L; weight matrices $W^l, \forall l \in \{1, \ldots, L\}$; ; non-linearity σ;

Output: New video embedding X^L for all video nodes;

$x_i^0 \leftarrow x_i, \forall v_i \in V$;
for $l = 1 \ldots L$ **do**
  for $v_i \in V$ **do**
    $\alpha \leftarrow softmax\big(ReLU((x_i^{l-1}||x_j^{l-1})e_{i,j})\big), V_j \in N(i)$;
    $x_i^l \quad \sigma\big(SUM(W^l \alpha x_j^{l-1}\big), V_j \in N(i))$;
  end for
end for
return $X^L \quad x_i^L, \forall V_i \in V$

example, video V_3 has been co-watched with V_1, while video V_1 is further linked with a "friends watched" video V_5. The more steps we can go into the graph, the more context information we can obtain from increasingly further away on the graph.

Based on this observation, we design a context-dependent propagating layer for the video graph that borrows the key ideas of Graph Convolutional Networks (GCN) [8, 9] to simulate the video propagation along different meta-paths based social connections, which also enables us to generalise CNN to graphs. GCN is proposed to collectively propagate and aggregate information from the graph structure, which can thus model input and output consisting of the nodes' feature and their interdependency. Our proposed propagating layers have a feedforward architecture, and the input is node feature X with dimension and graph structure A. After passing through the first propagating layer, the output is a new node feature matrix X^1 with dimension $N \times F^1$ obtained by aggregating the node features of neighbourhood videos.

By stacking multiple layers, the final representation of each video node receives dependency messages across further context steps. One propagating layer encodes only context information about immediate neighbours and L layers are required to encode the L-order neighbourhoods (i.e., context interdependency among nodes at most L hops away). Each extra propagating layer extends the neighbour to a further-away context. Within each hidden layer, nonlinear activations can be applied to the node features X. After L layers, the final output node features $X^{(L)}$ can be considered an embedding of the graph nodes in an F^L-dimensional space. For simplicity, in one single propagating layer, let X and $X^{'}$ be the input and output node feature matrices with sizes $N \times F$ and $N \times F^{'}$, respectively. This one-hop propagating process is defined as follows:

$$x_i^{'} = \sigma(\sum_{j \in N(i)} e_{ij} x_j W + b) \tag{5.4}$$

where W is the trainable parameter with dimension $F^{'} \times F$, and $N(i)$ is the neighbourhood of node i in the graph. $N(i)$ also includes v_i itself. The key idea behind

the propagating layer is that the nodes progressively aggregate information from connected neighbours into each node's own representation; as this process iterates, we can mine the user's hierarchical potential interests from the global context of the graph. Thus, a target video node can incorporate the extended interest videos from friends co-watched, with the same tag or category.

Node features guided attention

A critical concern about this aggregate mechanism is that different videos in a particular context may have different dependency weights for target nodes. We now introduce a form of attention into the aggregating process, which constitutes an essential part of the model. Graph Attention Network (GAT) [10] is a recently proposed method which computes the representations of each node in the graph. However, GAT only uses node features to decide the attention weights. The proposed attention function depends on both node features and edge features. The motivation is two-fold: (1) to identify the most relevant potential neighbour video to produce the recommendation; (2) to incorporate real-valued edge features that reveals the most frequently co-occurring videos. Practically speaking, we estimate the relevance weights of each possible neighbour video with the target video, which is also guided by the edge between them:

$$x_i^{'} = \sigma(\sum_{j \in N(i)} \alpha(x_i, x_j, e_{ij}) x_j W + b) \tag{5.5}$$

where α is the so-called attention coefficient. α is a function of x_i, x_j and e_{ij}, which can indicate the importance of node j's features to node i. Moreover, our attention function is chosen to be the following:

$$\begin{aligned} \alpha(x_i, x_j, e_{ij}) &= softmax\big(f(x_i, x_j) e_{ij}\big) \\ &= \frac{\exp(f(x_i, x_j) e_{ij})}{\sum_{k \in N(i)} \exp(f(x_i, x_j) e_{ij})} \end{aligned} \tag{5.6}$$

where the attention function f is a single-layer feedforward neural network, parametrized by a weight vector $a \in \mathbb{R}^{2F^{'}}$. Fully expanded out, the coefficients computed by the attention mechanism (illustrated by Figure 5.3 (c)) may then be expressed as:

$$\alpha(x_i, x_j, e_{ij}) = \frac{\exp(LeakyReLU\big(a^T [W x_i || W x_j]\big) e_{ij})}{\sum_{k \in N(i)} \exp(LeakyReLU\big(a^T [W x_i || W x_j]\big) e_{ij})} \tag{5.7}$$

where $||$ is the concatenation operation and the LeakyReLU has negative input slope 0.2. The algorithm for the propagating layer update is given in Algorithm 3.

Multi-head attention

The single-head graph attention may suffer from stability problems; thus, we employ multi-head attention in this chapter. The multi-head attention extracts multiple representations of the dependency among individual videos, which allows the model to jointly attend to information from different representation subspaces

for different videos [125]. Moreover, with multiple convolutional layers, high-order relation representations can be extracted, which effectively capture the effect of other agents and greatly assist in co-operative decision-making. Specifically, K independent attention mechanisms execute the transformation of Eq. 5.4, after which their features are concatenated, resulting in the following output feature representation:

$$x_i^{'} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in N(i)}\alpha^k(x_i, x_j, e_{ij})x_j W + b\right) \tag{5.8}$$

Unlike purely content or collaborative filtering based methods, the propagating layer method leverages both multimodal content information and graph structure. As a result, each video is modelled as a compact representation that considers the interdependency, heterogeneous and multimodal properties of the video recommendation.

5.2.5 Video Prediction

A recurrent neural network (RNN) has been shown effective in capturing and characterizing the temporal dependency in sequence data. Following [126], to use the learned video embeddings for recommendation purposes, we feed each user's watch sequence into the RNN to learn the user preference from the watch history. Specifically, we adopt the gated recurrent unit (GRU) [127] network as the base sequential recommender in our work, since it is simpler and contains fewer parameters than the LSTM.

As shown in Figure 5.3(c), we apply L-layers propagating over the video graph; then, each video node is represented by a new vector $x^L \in X^L$ based on equation 5.8; Given the historical watching sequence $\{i_1, \cdots, i_t\}$ of user u, our GRU-based recommender computes the current hidden state vector h_t^u conditioned on previous hidden state vector h_{t-1}^u as follows:

$$h_t^u = GRU(h_{t-1}^u, x_{i_t}^L; \Theta) \tag{5.9}$$

where GRU$(\cdot)$ is the GRU unit, $x_{i_t}^L$ is the embedding vector for item i_t, and Θ denotes all related parameters of the GRU networks. Thus, the predictor encodes the historical watching sequence of u into a hidden vector h_t^u, which models the sequential preference of u at t-th video in the user's interaction history. Hence, we call h_t^u the sequential user embedding of user u.

To generate the sequential recommendation, we rank a candidate video i by computing the recommendation score $\hat{y}_i$ according to:

$$\hat{y}_i = \sigma({h^t}^\top x_i) \tag{5.10}$$

where $\sigma(x) = 1/(1+\exp(-x))$ is a sigmoid function. To train this model, we define the training loss as:

$$\mathcal{L} = \sum_{h_t^u, x^p, x^n} \max(0, -{h_t^u}^\top x^p + {h_t^u}^\top x^n + \Delta) \tag{5.11}$$

where h_t^u, x^p, x^n is a triplet pair. x^p is the positive video representation that denotes the next video that the user liked in groundtruth. We also sample negative videos x^n that the user does not interact with or has disliked. The basic idea is that we want to maximize the inner product of positive examples. Simultaneously, we want to ensure that the inner product of negative examples is smaller than that of the positive samples by a pre-defined margin Δ. In the training phase, the negative samples x^n are randomly selected from the training set and resampled each epoch.

To train our model, the video node feature extraction (part a) and video embedding learning (part b) are first independently trained. The obtained node feature and node structure are fed into the GCN (part c). The GCN and sequential recommendation via the RNN are jointly trained for the final recommendation prediction.

5.3 Experiments

In this section, we conduct extensive experiments on real-world datasets to evaluate the proposed CDPRec method.

5.3.1 YouTube video dataset

To collect datasets containing both video and heterogeneous auxiliary information, we began with the Google+[†] site, which encourages its users to share their user accounts on other social networks. For simplicity, we adopted the user relationships from a cross-network dataset [3]. This dataset contains user account linkage between YouTube and Twitter and includes 143,259 Google+ users, among which 11,850 users provided both their Twitter and YouTube accounts. For each user relation, we downloaded the user's follower set from Twitter. For each user on YouTube, we further collected his/her watch-list over a one-year span of time from June 2013 to June 2014. However, the obtained data face data imbalance due to long-tail problems. Most users only watched a small proportion of the videos, and most of the videos were only watched a few times. If we were to train our model on such imbalanced datasets, the model would be dominated by observations with few active videos. To better evaluate the recommendation results, we filter the users by keeping the ones who watched over seven different videos on YouTube. Videos watched by fewer than three users are removed. Finally, we obtain 6,762 users and 10,894 videos in total for our experiment evaluation. For each video, we downloaded related information such as video categories, tags, titles, and descriptions via YouTube-dl[‡]. We only keep the top 1,000 frequent tags in our experiment to avoid noise. Table 5.2 summarizes the key statistics of the YouTube data.

5.3.2 Evaluation Metrics

Within the experimental dataset, we train our proposed model utilizing 80% of the users, and both validation and test sets contain 10% of users to evaluate the recommendation results. We further design two different experimental modes: the

[†]https://plus.google.com/

[‡]http://rg3.github.io/YouTube-dl

Table 5.2 : Statistics of YouTube dataset.

Users	Videos	Watch Records	Tags	Categories
6762	10894	169596	1000	33

short-range mode and the long-range mode. In the short-range mode, we use the first three months watch-list to model the user preference and keep the remaining nine months of watch-list as the objective for prediction. In the long-range mode, we use the first nine-month watch list to predict the remaining three-month watch-list. Following [46] and [37], we rank the videos by the predicted rating values and retrieve the top K videos, which is known as the top-K recommendation. The precision at rank K (Prec@K), recall at rank K (Recall@K) and F-score are employed as the evaluation metrics. We set $K = 5$ in our experiments.

To compare the performance of different algorithms, we add an evaluation metric AUC to evaluate the ranking performance of the recommendation results. AUC, which is the area under the ROC curve, is a widely used measure to evaluate the ranking performance:

$$\mathbf{AUC} = \frac{1}{u}\sum_{u\in U}\frac{1}{|J||J'|}\sum_{i\in|J|}\sum_{i\in|J'|}\delta(p_{u,j} > p_{u,j'}) \tag{5.12}$$

where J denotes the positive samples set, and $J^{'}$ denotes the negative. $delta(p_{u,j} > p_{u,j'})$ is an indicator function that returns 1 if $(p_{u,j} > p_{u,j'})$ is true and 0 otherwise. $p_{u,j}$ is the predicted probability that user u may act on video V_i in the test set. Better ranking performance will yield a higher AUC value. An AUC larger than 0.5 indicates that the result outperforms random guessing, and the best result is 1.

5.3.3 Baselines

We compare the proposed CDPRec with three types of representative recommendation methods: CF-based methods, which only leverages implicit user-video interaction information; content-based method, which analyses the video content to mine user preferences; HIN-based methods, which utilizes rich heterogeneous auxiliary information. To examine the effectiveness of incorporated multimodal content and the propagating mechanism, we prepare three variants of the CDPRec (CDPRec$_{dw}$, CDPRec$_{mg}$, CDPRec$_{propa}$). The comparison methods are given below:

- **LFM** [128]: A collaborative filtering based on the latent factor model that works by exploiting both explicit and implicit feedback by the users.
- **ItemKNN** [129]: The typical item-based top-N recommendation algorithm recommendation method, which recommends videos to a user using item-to-item similarities to compute the recommendations.

Table 5.3 : Experimental Results on YouTube Data. We indicate the data source for each model. (1) "CF" denotes the user-item interaction for Collaborative Filtering purpose. (2) "CT" denotes the content feature of the video. (3) "HIN" denotes the Heterogeneous Information Network that consists by users and attribute information of videos. (4) "Social" denotes the social relationship among users.

Model	Data Source	Short model			Long model			AUC
		Precision	Recall	F-score	Precision	Recall	F-score	
LFM	CF	0.0024	0.0019	0.0021	0.0032	0.0046	0.0038	0.6513
ItemKNN	CF	0.0163	0.0127	0.0142	0.0169	0.0255	0.0203	0.6942
BPR-MF	CF	0.0182	0.0121	0.0145	0.0174	0.0273	0.0214	0.7128
Bi-LSTM	CF+CT	0.0231	0.0143	0.0176	0.0196	0.0349	0.0251	0.7417
CMF	HIN	0.0296	0.0181	0.0225	0.0238	0.0367	0.0288	0.7824
HeteMF	HIN	0.0321	0.0234	0.0270	0.0258	0.0376	0.0306	0.7939
SemRec	HIN + Social	0.0347	0.0247	0.0288	0.0287	0.0390	0.0330	0.7965
Dynamic RNN	CF + CT + Social	0.0349	0.0257	0.0296	0.0293	0.0411	0.0342	0.8123
HERec	HIN + social	0.0361	0.0264	0.0305	0.0306	0.0425	0.0355	0.8174
CDPRec$_{dw}$	HIN + social	0.0292	0.0226	0.0254	0.0241	0.0373	0.0293	0.7815
CDPRec$_{mp}$	HIN + social	0.0352	0.0253	0.0294	0.0316	0.0419	0.0360	0.8023
CDPRec$_{prapa}$	HIN + social	0.0383	0.0287	0.0328	0.0315	0.0431	0.0363	0.8338
CDPRec	HIN + CT + social	**0.0401**	**0.0314**	**0.0352**	**0.0346**	**0.0478**	**0.0401**	**0.8615**

- **BPR-MF** [130]: A form of Bayesian personalized ranking based on the matrix factorisation method and a pairwise learning method for an item recommendation, which trains on pairs of positive observed interaction and negative unobserved counterparts of a user and takes the Matrix Factorization as the underlying predictor.

- **Bi-LSTM** [131]: Bi-LSTM (bi-irection long short-term memory) units are building units for the layer of a recurrent neural network (RNN), which makes sequential predictions. Here, we use the multi-modal video representation as features for the input.

- **Dynamic RNN** [126]: A dynamic RNN to model the dynamic interests of users over time by considering video semantic embedding and user relevance mining in a unified framework for video recommendation. Compared with the Bi-LSTM [42] method, [36] add additional user relationship constraint in the RNN. For a fair comparison, we did not include the additional twitter text for topic modelling.

- **CMF** [132]: A collective matrix factorization method, which extends the matrix factorization to multi-relational HINs by optimizing a joint objective over all relation types.

- **HeteMF** [113]: A matrix factorization-based recommendation method that uses meta-path-based item similarities for recommendation.

- **SemRec** [36]: A recommendation method utilized in weighted HINs that uses a weighted meta-path to obtain different preferences of users on the paths.

- **HERec** [112]: HERec is the state-of-art HIN-based ranking method that merges different meta-path guide node embeddings for item recommendation. For a fair comparison, we concatenate this video node embedding with the video content features to form the representation of a video.

- **CDPRec**$_{dw}$ A variant of the CDPRec, which incorporates the homogeneous network embedding method DeepWalk [1] and views all nodes in the HINs as being of the same type.

- **CDPRec**$_{mg}$ A variant of the CDPRec that incorporates the heterogeneous network embedding method of methpah2vec [13]. methpah2vec leverages a heterogeneous skip-gram model to exploit the meta-path heterogeneous neighbourhood of a node.

- **CDPRec**$_{prapa}$ Another variant of the CDPRec that ignores the multimodal video content feature. The video feature is obtained from the user/item interaction matrix with matrix factorisation model SVD.

- **CDPRec**: Our complete model.

We implement the CDPRec model using the Python library of TensorFlow. For our model, we randomly initialize the model parameters using the Gaussian distribution and optimize the model using adaptive moment estimation (Adam). During training, we apply L_2 regularization with $\lambda = 0.0005$. Furthermore, dropout with $p = 0.6$ is applied to the input of each propagating layer and the normalized attention coefficients. For MF-based recommendation methods, we follow the optimal configuration and architecture reported in [132]. For the other comparison methods, we optimize their parameters using 30% training data as the validation set. All experiments are conducted on a machine with four GPUs (NVIDIA GTX-1080 * 4), one CPU (i7-5960X CPU @ 3.00 GHz) and 48 GB memory.

5.3.4 Experimental Results

Table 5.3 reports the results of our proposed model and baselines on the YouTube dataset. We can make the following observations from the experimental results:

1. In all cases, our proposed complete model of the CDPRec outperforms all compared baseline methods in both short and long modes. Moreover, most of the improvements achieved by our method are significant compared to the best baseline methods. We attribute the superiority of the CDPRec to the following three properties: (a) semantic embedding of a video is learned to map multimodal visual and text information into a common semantic space to effectively integrate the content and extract deep interdependency structure information; (b) our context-dependent propagating mechanism adopts a more comprehensive method of improving the aggregation of the information of different meta-paths to improve the recommendation performance; (c) attention improves the representations for the meta-path-based local neighbour video in a mutually beneficial manner.

2. Among the baseline methods, HIN-based methods (CMF, HeteMF, SemRec and HERec) outperform CF methods (ItemKNN, LFM, BPR-FM) and the content-based method Bi-LSTM in most cases, which demonstrates the benefit of auxiliary heterogeneous information. Compared with the Bi-LSTM (CF + CT) [131] method, the Dynamic RNN (CF + CT + Social) [126] adds additional users' common interest (user relationship) constraint in the RNN. It achieves better performance than Bi-LSTM. It is noteworthy that the recently proposed HERec model works well among these baselines, since this method adopts a similar path guided random walk to generate a context sequence for user/item embeddings.

3. The proposed CDPRec (HIN + CT + Social) achieves better performance than Dynamic RNN (CF + CT + Social) [126]. It demonstrates that the powerful heterogeneity modelling ability of the HIN and our proposed context-dependent propagating method with attention provide additional ability to mine the complex context information and hidden interdependency in the HIN.

4. For the three embedding-based variants of the CDPRec, we notice that the order of overall performance is as follows: CDPRec $>$ CDPRec$_{propa}$ $>$ CDPRec$_{mp}$

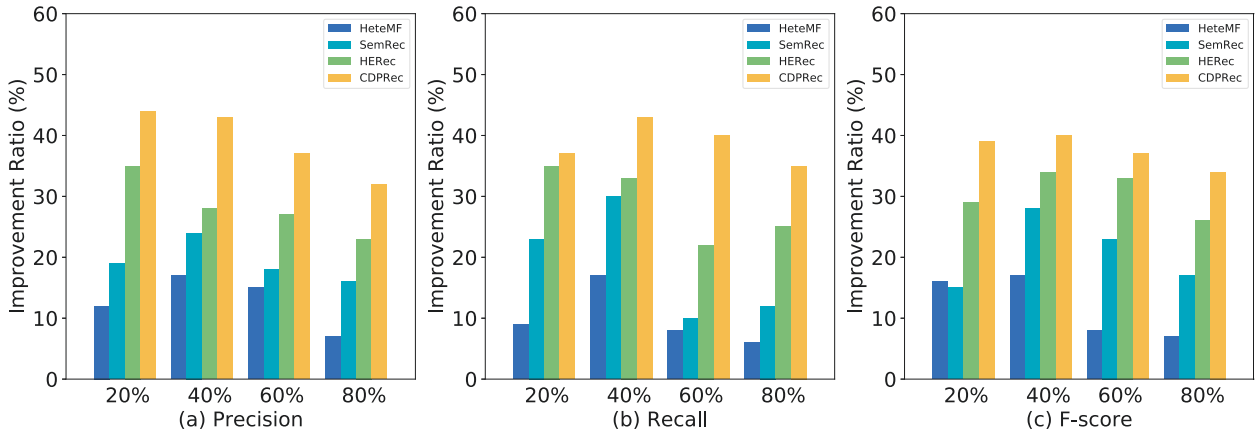


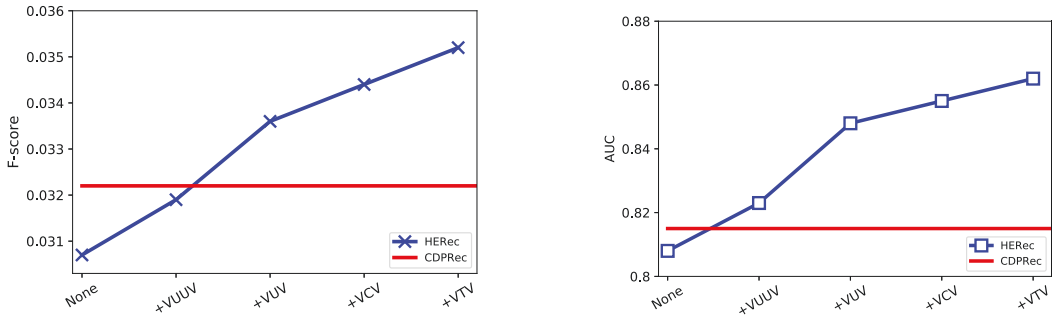
Figure 5.4 : Performance comparison of different methods for cold-start recommendation YouTube datasets. y-axis denotes the improvement ratio over CMF.

> CDPRec$_{dw}$. (a) CDPRec (with multimodal content feature) performs better than CDPRec$_{propa}$ (with sparse user/video interaction feature), which indicates that the video content with visual and text information can provide discriminative features for video modelling. This additional information is particularly useful when analysing users with a small number of rating records. (b) CDPRec$_{propa}$ (with propagating-based embedding) outperforms CDPRec$_{mp}$ (with the meta-path-based similarity embedding). More importantly, both CDPRec$_{propa}$ and the strongest baseline HERec only leverage the graph structure information of the HIN and without considering the content feature. However, CDPRec$_{propa}$ outperforms HERec. The reason is that the propagating mechanism can effectively mine the high-order potential interests of the users, which may result in more useful context information being gained. (c) In addition, the improved performance of CDPRec$_{mp}$ relative to CDPRec$_{dw}$ confirms the benefit of the HIN-based embedding in modelling relation data of multiple types. (d) In summary, CDPRec achieves the best performance under all circumstances. Based on these results, we argue that it is necessary to learn the deep independency structure and incorporate multi-modal video content features that can improve the recommendation performance.

5.3.5 Detailed analysis of the proposed model

Cold-start recommendation

The "cold start" originates from the data sparsity problem. It is a real challenge for social recommendation to perform accurate recommendation using less training data. HIN is particularly useful for alleviating the cold-start problem in recommendation by leveraging the deep interdependency context information. In this part, we conduct experiments with different levels of cold-start data to validate whether the CDPRec can handle the cold-start problem. For comparison, we run baseline methods on the YouTube dataset with different sparsities. Here, we only



(a) F-score (b) AUC

Figure 5.5 : Performance change of CDPRec when meta-paths are gradually added.

use the HIN-based recommendation baselines, including CMF, HeteMF, SemRec and HERec. Following [37], we first randomly shuffle the dataset and split it into five equal groups. We set aside the last group as a hold-out or test data set; then, we vary the number of remaining groups from one to four, which correspond to 20%, 40%, 60%, and 80% of data being used as training sets.

For simplicity, we use the ratios of improvement w.r.t. CMF. The experimental results on the YouTube video dataset are presented in Figure 5.4. Overall, all compared methods perform better than CMF. The proposed CDPRec method performs consistently better than all baselines; moreover, with fewer datasets being used, the improvement over CMF becomes more significant. The results show that various types of rich-context information in HINs are useful for providing better recommendation results, and the proposed CDPRec approach can utilize this information in a more principled manner.

Impact of different meta-paths

In this chapter, we use different meta-path-based random walks to model local video contexts. To analyse the effect of different meta-paths on our final recommendation performance, we gradually add meta-paths into the CDPRec one by one and investigate the impact. For ease of analysis, we consider the HERec to be the basic reference. Figure 5.5 shows, that the recommendation performance monotonically increases with the addition of more meta-paths. Meanwhile, meta-paths appear to have different effects on the recommendation performance. In particular, adding VUUV and VUV yields a significant performance boost, while the performance improvement resulting from adding VTV and VCV is relatively lower. The reason may be that user-generated video tags and categories will introduce more local context neighbours than social relation does, which can include noise or information that conflicts with existing neighbours when iteratively fusing them with the propagating mechanism.

Table 5.4 : The results w.r.t. different layer number.

Layer number L	1	2	3	4
Precision	0.0357	0.0374	0.0401	0.0386
Recall	0.0289	0.0291	0.0314	0.0307
F-score	0.0319	0.0327	0.0352	0.0341
AUC	0.8217	0.8464	0.8615	0.8357

Propagating layer number

We also vary the size of local graph context neighbourhoods by the maximal layer number L and observe the performance changes. The results are listed in Table 5.4, which clearly shows that when layer number L increases, the performance initially increases but eventually falls when L is too large (typically when $L = 3$). We attribute this phenomenon to the trade-off between useful information from the long-distance dependency and useless information from noise: (1) a too-small layer value L cannot adequately explore the deep interdependency and gain sufficient context information; and (2) a too-large value of L may lead to an overfitting problem when the number of parameters increases, while simultaneously introducing more undesired noise from the neighbours.

5.3.6 Case study

Extra user-video connectivity information derived from the HIN endows the recommender systems the ability of reasoning on paths to infer the user preferences towards target items and generating reasonable explanations. To demonstrate this case, we show an example drawn from the CDPRec on a video recommendation task as shown in Figure 5.6. We randomly sample a user with two historical watched videos $\{V_1, V_2\}$. Then, our system recommends video $\{V_6, V_7\}$ to this user. For each multi-hop relevant video of the user, we calculate the (unnormalized) relevance probability between the watched video $\{V_1, V_2\}$ and the predicted videos $\{V_6, V_7\}$. We have several observations.

The predicted video is connected to what the user has watched (e.g., V_1, V_2) by the shared video entities such co-friends viewed (V_3) and co-tag (V_4). This result shows that the CDPRec can extend the user interests along HIN paths. By analysing two paths for prediction V_7, we also find that different paths may describe the user-item connectivity from dissimilar angles, which can be treated as the evidence of why the item is suitable for the user. Our method provides a new viewpoint for the explainability by tracking the paths from a user's history to a new video with high relevance probability in the HIN, such as "$user \xrightarrow{watched} V_2 \xrightarrow{co-tag} V_5 \xrightarrow{friends-watched} V_7$" or "$user \xrightarrow{watched} V_1 \xrightarrow{co-tag} V_4 \xrightarrow{co-category} V_7$". Thus, recommendations tell the users what videos they may like while revealing the reason with which they may like them, which can help to improve the persuasiveness and user satisfaction of the recommender systems.

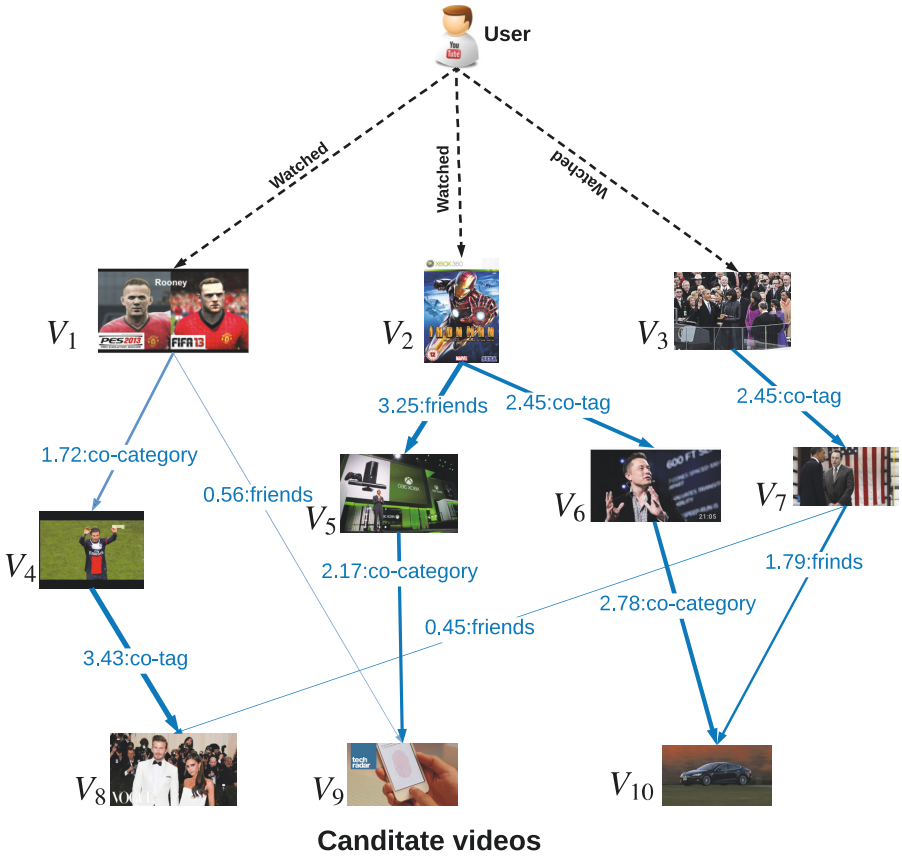


Figure 5.6 : Visualization of three paths with relevance probabilities for a user.

5.3.7 Parameter sensitivity

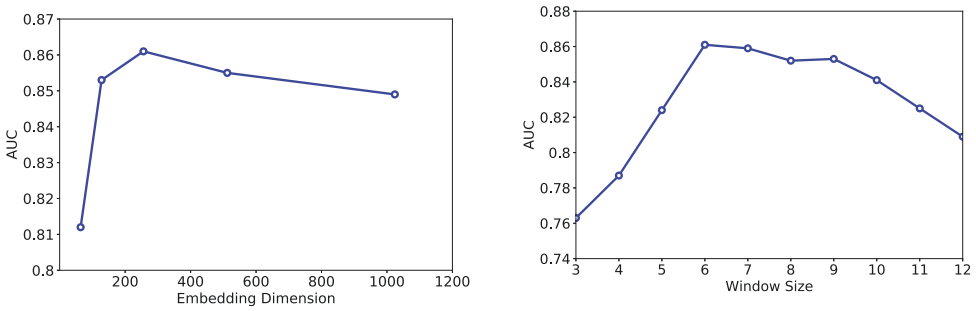
In this section, we investigate how the recommendation performance varies with the hyper-parameters in (1) the dimension of final video embedding and (2) the length of the co-occurrence window when constructing the homogeneous video graph.

Dimension of video embedding

We vary the dimension of video embedding to 64, 128, 256, 512 and 1024. Figure 5.7(a) shows the accuracy of the CDPRec over different video embedding dimensions. The accuracy shows an apparent increase at first because more bits can encode more useful information. However, the performance subsequently starts to slowly decrease when we continuously increase the number of dimensions. This result is intuitive, since a too-large dimension size requires more data to prevent an overfitting problem.

Length of co-occurrence window

When constructing the homogeneous video graph from the HIN, the length of the context window controls the "density" of the video graph. As shown in Figure 5.7(b), a larger window interval does not seem to help; on the contrary, a larger



(a) Video embedding dimension (b) Size of Co-occurrence window

Figure 5.7 : Parameter sensitivity of CDPRec.

window corresponds to lower precision likely because a relation among videos that are further apart is not sufficiently strong to define a connection in the graph.

5.4 Conclusion

In this chapter, we propose the CDPRec, a neural framework that incorporates HINs and multimodal video content into recommender systems in a natural and intuitive manner. The CDPRec overcomes the limitations of path-based similarity methods by introducing context-dependent propagation, which automatically propagates the potential preferences of the users and explores their hierarchical interdependencies in the HIN. The CDPRec encodes the global video context into comprehensive video embedding to facilitate the click-through rate prediction. Experimental results on real-world YouTube data demonstrate the significant superiority of our method over strong baselines.

Chapter 6

Knowledge Graph Enhanced Neural Collaborative Recommendation

Existing neural collaborative filtering (NCF) recommendation methods suffer from severe sparsity problem. Knowledge Graph (KG), which commonly consists of fruitful connected facts about items, presents an unprecedented opportunity to alleviate the sparsity problem. However, pure NCF models can hardly model the high-order connectivity in KG, and ignores complex pairwise correlations between user/item embedding dimensions. To address these problems, we propose a novel Knowledge graph enhanced Neural Collaborative Recommendation (K-NCR) framework, which effectively combines user-item interaction information and auxiliary knowledge information for recommendation task into three parts: (1) For items, the proposed propagating model learns the representation of item entity. It recursively aggregates information from its multi-hop neighbours in KG, and employs an attention mechanism to discriminate the importance of the relation type to mine users' potential preferences. (2) For users, another heterogeneous attention weights are leveraged to strengthen the embedding learning of users. (3) The user and item embeddings are then fed into a newly designed two-dimensional interaction map with convolutional hidden layers to model the complex pairwise correlations between their embedding dimensions explicitly. Extensive experimental results on three benchmark datasets demonstrate the effectiveness of our K-NCR framework. In the meantime, the high-order connectivity of KG is seamlessly incorporated into this recommendation framework. Finally, extensive experimental results on three real-world datasets clearly show the effectiveness of our proposed model.

6.1 Introduction

As the amount of data from different platforms grows at an unprecedented rate, the recommender system is becoming more and more crucial to help users discover personalised content of interest from these ever-growing corpus of data. For example, there are 11,895 movies released around the world in 2018*. It is very challenging for users to identify their interested ones. Due to their extraordinary capability of exploiting collective wisdom and experiences, Collaborative Filtering (CF) algorithms, especially Matrix Factorisation (MF) algorithms — a technique that predicts users' personalised preference from user-item interactions only — has proven to be the most widely used recommendation technology [133]. Matrix factorisation assumes that there are some latent factors that exist behind the interactions between

*https://www.imdb.com/search/title/?year=2018&title_type=feature

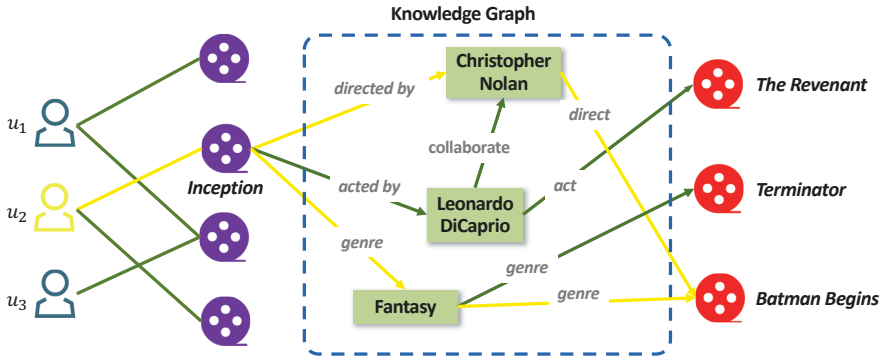


Figure 6.1 : Illustration of KG-enhanced movie recommender systems. The knowledge graph can provides extra user-item connectivity information, which are useful for alleviating the sparsity problem of target recommendation task.

users and items. Though widely studied in the past, traditional matrix factorisation suffers from the severe sparsity problem of user-item interaction [134].

Inspired by the recent popularity of deep learning, Neural Collaborative Filtering (NCF) is proposed, as a new class of CF methods, cast the traditional MF algorithm into an overall neural framework [46, 48, 47]. Generally speaking, there are two key components in learnable NCF models: 1) embedding, which transforms users and items to vectorized representations, and 2) interaction modelling, which reconstructs historical interactions based on the user/item embeddings. For example, in [46], the authors proposed to replace the MF interaction function of inner product with non-linear deep neural networks. The translation-based CF models were proposed to use Euclidean distance metric as the interaction function [135]. However, these methods may not be sufficient to yield satisfactory embeddings for CF. The possible reason is that most existing methods build the embedding function with the descriptive features only (e.g., ID or attributes). As a result, when the interaction function goes deeper to capture complex user-item relationship, these methods tend to overfit and make the sparsity problem even worse.

Therefore, researchers proposed using auxiliary data in recommender systems to enrich the semantics of the user or item representation, such as social networks, attributes, and multimedia [136, 137, 138]. Recently, Knowledge Graphs (KGs) have attracted increasing attention, which usually consist of fruitful connected facts about items [139, 140]. KG introduces semantic relatedness among items, which is a useful tool to help mine their latent connections and improve the quality of recommendation. Extra connectivity information derived from KG endows recommender systems the ability of reasoning and alleviates sparsity problem [141]. Taking movie recommendation as an example in Figure 6.1, a user is connected to the movie *"The Revenant"* since she likes the movie *"Inception"* by the same actor *"Leonardo DiCaprio"*. Such connectivity helps to reason about unseen user-item interactions (*i.e.,* a potential recommendation) by synthesizing information from long paths.

The prior works on KG-based recommendation can be roughly categorised into

two categories considering the path and embedding fashion. (1) Path-based methods use predefined meta paths with specific connectivity patterns to capture different semantic similarities between users and items carried in KGs. Path-based methods exploit KG in an intuitive way, but they fail to automatically uncover and reason on unseen connectivity patterns, since they rely heavily on hand-crafted meta-paths. (2) Another widely researched line is embedding-based methods, which pre-process a KG with knowledge graph embedding (KGE) [15] algorithms, such as TransE [16] and TransR [18], and then leverage the learned entity embeddings to regularise the representations of items. The high-order connectivity denotes the path that reaches users from any entity node with the path length larger than 1 as the above movie example shows. However, these methods only consider direct relations between entities, rather than the high-order connectivity paths.

Recent developments of graph neural networks [8, 29, 9] try to automatically capture high-order structure information in a graph, which has the potential of achieving the goal but has not been explored much for KG-based recommendation. Another key deficiency is that they model each interaction in KG as an independent instance and do not consider their relations, which makes them insufficient to discriminate the various user preference revealed by different relation type in KG. As shown in Figure 1, movie *"Inception"* connects to movie *"Batman Begins"* with two kinds of relations, which can be caused by user's different interest propensity (prefer a movie with the same "genre" or "director"). Moreover, these methods are proposed for a single recommendation task [46] with only KG data. And they fail to study knowledge enhanced recommendation using deep neural networks in an end-to-end way.

To this end, we investigate how to utilise neural collaborative based model for both user-item and auxiliary knowledge graph data, and enable the knowledge to enrich the CF model with sparse user-item interaction. Then three key challenges arise in this investigation. (1) how to learn item representation from complex heterogeneous structured KG. Traditional KGE regularisation only considers direct relations between entities, rather than the multi-hop relation paths. (2) how to model user profile from weighted items. Existing user profile modelling methods can be limited by its assumption that all historical items of a user profile contribute equally in estimating the similarity between the user profile and a target item. Intuitively, a user interacts with multiple items in the past, but it may not be true that these interacted items reflect the users interest to the same degree. (3) how to capture the complex interactions between users and items, which is not exploited by the prior shallow interaction-based methods. The traditional inner product matching function assumes that the dimensions of users and items embedding are independent with each other, and contribute equally for the prediction of all data points [46]. To some extent, this assumption is impractical, since the embedding dimensions could be interpreted as certain properties of items [142, 48], which are not necessarily independent.

In this chapter, we propose a novel Knowledge Graph enhanced Neural Collaborative Recommendation(K-NCR) approach. K-NCR tackles above three challenges in a recommendation scenario as shown in Figure 6.1. The proposed K-NCR has

three key modules: (1) For item modelling, we exploit the high-order structural proximity among entities in KG instead of the one-hop local neighbour structure. In particular, we capture the influence diffusion process among items with multi-hop based on Graph Convolutional Network (GCN), which is capable of stimulating the propagation of user preferences over the set of knowledge entities by iteratively and automatically extending potential interests along with links in the knowledge graph. Moreover, we employ an attention mechanism to discriminate the importance of the relation type to mine users' potential preferences. (2) For user modelling, to alleviate the limitation of traditional mean-based aggregator in NCF, the varying importance weights are learned from the interacted items with an attention network. These weights can distinguish historical item in a user profile is playing a more important role for the user preference modelling. (3) To capture the complex interaction between user and item embeddings, our proposal of using outer product above the user and item embedding layer results in a two-dimensional interaction map. The interaction map is rather suitable for the CF task, since it not only subsumes the interaction signal used in MF (its diagonal elements correspond to the intermediate results of inner product), but also learns possible complex dimension correlations with CNN layers. Thus, the auxiliary knowledge information and complex nonlinear interaction are modelled in the proposed unified architecture. Given the proposed neural architecture, we design a joint learning framework that allows the KG embedding part and the neural collaborative recommendation part to enhance each other mutually.

The main contributions of this chapter are as follows:

1. We are the first to combine KG structure information with collaborative recommendation in an end-to-end neural style. We highlight the importance of explicitly modelling 1) the high-order connectivity of knowledge graph to provide a better recommendation with item side information and 2) the pairwise correlations between the dimensions of the embedding space in the matching function.

2. We propose a new recommendation framework K-NCR, which encodes the high-order connectivities of KG and complex matching interaction in an explicit and end-to-end manner by performing embedding propagation and outer product-based convolutional NCF, respectively.

3. We conduct empirical studies on three million-size real-world dataset. Extensive experiment results demonstrate the state-of-the-art performance of K-NCR and its effectiveness in improving the embedding and matching quality for the final prediction.

6.2 PRELIMINARY

Before introducing the proposed approach, we first define the task of knowledge graph enhanced recommendation, and then shortly recapitulate the standard neural collaborative filtering, highlighting its limitations for dealing with the task.

Table 6.1 : Notations and explanations

Notations	Description
$y_{ui} \in Y$	Users' implicit feedback
$\hat{y}_{ui}$	Predicted engaging probability
$\mathcal{G} = (\mathcal{E}, \mathcal{R})$	Knowledge graph
(h, r, t)	A knowledge triple (head, relation, tail)
$\mathcal{E} = \{e_1, e_2, \cdots\}$	Set of entities
$\mathcal{R} = \{r_1, r_2, \cdots\}$	Set of relations
$\mathcal{N}_i$	Neighbors of entity e_i
$\mathcal{N}_i^r$	Neighbors of entity e_i under relation r
$\mathbf{h}_i^{(l)}$	Entity embedding in layer l
$\mathbf{k}_u \in \mathbb{R}^S$	latent vector for user u
$\mathbf{j}_i \in \mathbb{R}^S$	latent vector for item i
$\mathbf{p}_i \in \mathbb{R}^S$	item embedding i
$\mathbf{q}_j \in \mathbb{R}^S$	item embedding j in user history

6.2.1 Problem Formulation

In the study of KG-enhanced recommendation, we have a target recommendation domain with user-item interaction data and an auxiliary knowledge graph data domain . For the data in the target recommendation domain, we have a set of users $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$ and a set of items $\mathcal{I} = \{i_1, i_2, \ldots, i_N\}$, where M and N are the number of users and items, respectively. The user-item interaction data is usually presented as a bipartite graph. In particular, the matrix of user-item interaction $Y \in \mathbb{R}^{M \times N}$ is constructed according to users' implicit feedback as follows:

$$y_{ui} = \begin{cases} 1 & \text{if interaction } (u,i) \text{ is observed;} \\ 0 & \text{otherwise.} \end{cases}$$

Here an observed interaction ($y_{ui} = 1$) implies that user u engaged with item i, e.g., users browsed a news or user purchased a product; however, it can only indirectly reflect users' preference. Similarly, $y_{ui} = 0$ does not naturally mean user u has no interest in item i, it could be that user u is not aware of the item since the overwhelming amount of items available in a system. The unobserved entries of user-item interaction matrix can be just missing data, and there is a natural scarcity of negative feedback, which poses challenges in learning from these implicit interaction data [**?**]. The problem in the target recommendation domain can be formulated as evaluating the scores of unobserved entries in Y.

In the auxiliary data domain, we have access to a knowledge graph $\mathcal{G}$, which is a directed graph composed of subject-relation-object triple facts (h, r, t), which indicates a fact that there is a relationship r from head entity h to tail entity t. For example, the triple (*Christopher Nolan*, *film.director.film*, *Batman Begins*) states the fact that *Christopher Nolan* directs the film *Batman Begins*. KG provides deep factual knowledge and rich semantics on items. More important, by exploring the

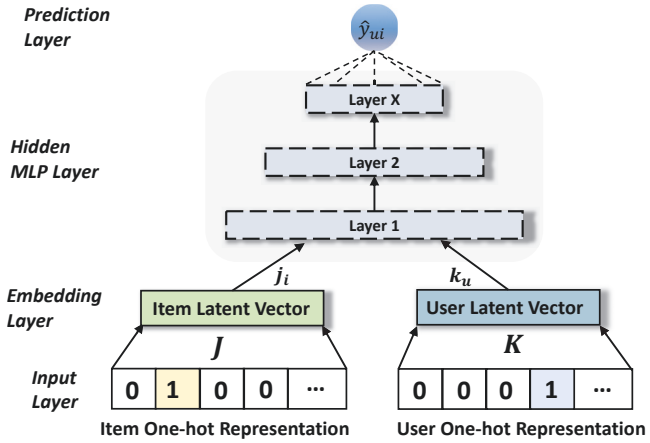


Figure 6.2 : Neural Collaborative Filtering (NCF): MF as a shallow neural network model.

interlinks within a KG, the connectivity between users and items reflects their underlying relationships, which are complementary to the sparsity user-item interaction [141].

Given the user-item interaction matrix Y as well as the knowledge graph $\mathcal{G}$, our goal is to predict the potential user interests for the unobserved feedbacks. Specifically, we try to learn a prediction function $\hat{y}_{ui} = \mathcal{F}(u, i|\theta, Y, \mathcal{G})$ to calculate the probability that user u may choose item i, where θ is the model parameters of function $\mathcal{F}$.

6.2.2 Neural collaborative filtering framework

Collaborative filtering (CF) is the fundamental methods for personalised recommendation systems, of which model based Matrix Factorization (MF) is one of the simplest yet effective implementations [51]. MF characterises both user and item with real-valued low-dimensional latent vectors, which infer a recommendation based on the high correspondence between item and user with the inner product:

$$\hat{y}_{ui} = f_{ui}(u, i|\mathbf{k}_u, \mathbf{j}_i) = \mathbf{k}_u^\top \mathbf{j}_i = \sum_{s=1}^{S} k_{us} j_{is}, \tag{6.1}$$

where $\mathbf{k}_u \in \mathbb{R}^S$ and $\mathbf{j}_i \in \mathbb{R}^S$ denote the latent vector for user u and item i, respectively. And S denotes the dimension of the latent space, which is much smaller than the size of users or items. Despite its effectiveness, we note that using the simple and fixed inner product can hardly estimate complex user-item interactions, which will limit the expressiveness of MF. It is straightforward to understand the inner product function as first applying the element-wise product on latent vectors $\mathbf{k}_u$ and $\mathbf{j}_i$, followed by linearly combining each element with the same weight. And each dimension of latent factors is treated independently from all others. As such, MF with one hidden layer only can be deemed as a linear model of latent factors [143].

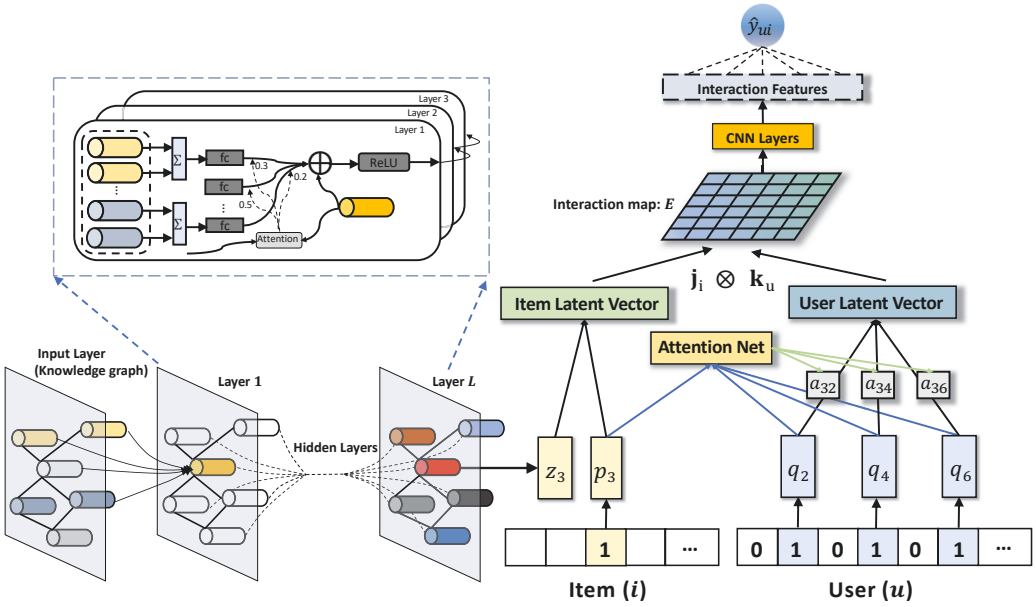


Figure 6.3 : Overall architecture of K-NCR. There are three components in the framework: (1) an knowledge aggregating-based Item modelling (Left) (2) attention-based User Modelling (Low right) and (3) Convolutional NCF for item/user interaction modelling (Upper right).

To overcome the limitation of linear MF model, it is natural to model the interaction between users and items with a two-way neural network. Neural Collaborative Filtering (NCF) [46] is the representative works as shown in Figure 6.2. Let M_u and M_i denote the feature information, or just one-hot representation of user u and item i. The prediction function is defined as:

$$\hat{y}_{ui} = f(K^T M_u, J^T M_i | \mathcal{U}, \mathcal{I}, \theta) \tag{6.2}$$

where $K \in \mathbb{R}^{M \times S}$ and $J \in \mathbb{R}^{N \times S}$ are the embedding matrix for user features and item features, respectively; function $f(\cdot)$ represents the multilayer perceptron, and θ is the parameters of this network. Traditional MF-based recommendation method can be viewed as a simple case of NCF, while the non-linear MLP hidden layer of NCF can model more complex user-item interaction.

Although using MLP to learn the matching function directly endows the model with great flexibility, there is no practical guarantee that the dimension correlations can be effectively captured with current optimisation techniques, especially with complex user and item interaction [48]. The case can be even worse if we take the auxiliary attributes information (such as Knowledge Graph, and multimedia content) into account. To overcome the above issue and further improve the performance of CF methods, we incorporate them under the proposed recommendation K-NCR framework.

6.3 Knowledge Enhanced Neural Collaborative Model

6.3.1 Framework

The goal of KG-enhanced recommendation is to select relevant information from the knowledge graph to assist the target recommendation prediction. In this chapter, we propose a knowledge graph enhanced Neural Collaborative Recommendation (K-NCR), an end-to-end framework that utilises KG to alleviate the sparsity problem of recommender systems. We now present the proposed K-NCR model, and the architecture of which is illustrated in Figure 6.3. There are three components in the framework: (1) knowledge propagation based item modelling, (2) attention based user preference modelling and (3) convolutional NCF for item/user interaction modelling.

6.3.2 Knowledge propagation based item modelling

Item modelling try to mine latent semantic embedding of items for recommendation. In the KG-enhanced recommendation, we assume each items embedding is composed of two parts: a KG embedding part $\mathbf{z}$ that could be captured by the correlation phenomenon of users interests in the Knowledge Graph. Besides, each user has his/her unique interest, which could not be modelled in the KG.

For entity embedding, a novel knowledge propagation model is proposed to embed an item entity in KG into a low-dimensional semantic embedding space, such that entities with neighbouring structure have similar vector representations. We propose a GCN (Graph Convolutional Networks) based entity embedding method, which is a prominent technique for feature representation learning in various graph structured data. Traditional KGE methods only consider the local structure of an entity, which is far from modelling users' varying interest in the knowledge. Different from traditional KGE, we show how to use GCN explore entities' potential high-order structural proximity on KG entities with deep learning-based models. For every entity node in the KG, GCN encodes relevant information about its neighbourhood as a real-valued feature vector in an unsupervised manner. Noting that, each item $i \in \mathcal{I}$ have its corresponding entity e_i in the knowledge graph.

As shown in the left part of Figure 6.3, one layer GCN encodes only information about immediate neighbours and L layers are needed to encode L-order neighbourhoods (*i.e.,* information about nodes at most L hops aways). Specially, representation propagation process of entity embedding captures high-order structural proximity among entities in a knowledge graph, which consists of two basic operation: (1) **Propagation**: We iteratively propagate an entity's contexts along links in a graph-structured KG and explore long-range dependencies among the surrounding entity nodes; (2) **Aggregation**: We then collectively aggregate feature of long-range neighbour entities to get the embedding of target entity. The key idea is to learn the iterative convolutional operation in graphs as a message-passing framework, where each convolutional operation means generating the current node

representations from the aggregation of local neighbours in the previous layer:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i} g(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)})\right), \tag{6.3}$$

where $\mathbf{h}_i^{(l)} \in \mathbb{R}^{d^{(l)}}$ is the latent embedding of entity e_i in the l-th layer of the neural network, and $\mathcal{N}_i$ is the neighbour entity of e_i. $g(\cdot,\cdot)$ is typically chosen to be a (message-specific) neural network-like function [144].

Considering the heterogeneous structure of knowledge graph, entities are connected by different type of relations, so it is unreasonable to ignore this crucial node type information. To this end, we define the following simple knowledge aggregating process by considering various relation type:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r} \frac{1}{c_{i,r}} W_r^{(l)}\mathbf{h}_j^{(l)} + W_0^{(l)}\mathbf{h}_i^{(l)} + b\right), \tag{6.4}$$

where W_r^l and b are the weight matrix and bias, respectively, σ is an activation function, e.g. a ReLU. $\mathcal{N}_i^r$ is the neighbour entity set of target entity i under relation $r \in \mathcal{R}$. $c_{i,r}$ denotes a task-specific normalizing constant, and here we set it as $c_{i,r} = |\mathcal{N}_i^r|$, the number of neighbors with relation type r. Intuitively, Equation (6.4) aggregates transformed hidden vectors of neighbouring entity with a normalised sum. Different from classical GCN, we propose a relation-specific transformations, which depends on the type of a relation edge. Besides, a self-connection relation edge is added to ensure that the initial representation $\mathbf{h}^{(l)}$ at layer l directly informs its new representation $\mathbf{h}^{(l+1)}$ at layer $l+1$.

Attention based aggregation

Equation (6.4) uses a sum operator as the aggregation function, which takes the element-wise mean of the neighbour vectors. It assumes that all type of neighbor contribute equally to the representation of the target entity e_i. However, as mentioned before, strong and weak ties are mixed together in a KG, and users are likely to share more similar tastes with strong ties than weak ties. For example, a user may have more potential interests in the movies that share the same "star" with his or her historically liked ones, while another user may be more concerned about "genre" of movies. Thus, we perform an attention mechanism with a two-layer neural network to extract these neighbour entity node that is important to target entity e_i, and model their tie strengths, by relation attention β_r with $\mathbf{t}_r$ (sum of neighbour

Algorithm 4: Propagating layer of KGE

Input: Knowledge graph $\mathcal{G} = (\mathcal{E}, \mathcal{R})$; depth L; weight matrices $W^l, \forall l \in \{1, \ldots, L\}$; non-linearity σ;

Output: Entity embedding $\mathbf{z}_i$;

```
for l = 1 ... L do
    for e_i ∈ E do
        t_r = AVG(h_j^(l)), j ∈ N_i^r;
        β_r      softmax(ReLU(t_r ⊕ h_i^(l)));
        h_i^(l+1)      σ(SUM(W^l β_r t_r) + h_i^(l));
    end for
end for
return z_i      h_i^(L), ∀e_i ∈ E
```

embedding with relation type r) and the target entity embedding $\mathbf{h}_i$, as below,

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}} \beta_r \mathbf{t}_r + \mathbf{W}_0^{(l)} \mathbf{h}_i^{(l)} + b_0\right) \tag{6.5}$$

$$\mathbf{t}_r = \sum_{j\in\mathcal{N}_i^r} \frac{1}{c_{i,r}} \mathbf{W}_1^{(l)} \mathbf{h}_j^{(l)} \tag{6.6}$$

$$\beta_r^* = \mathbf{w}_2^T \cdot \sigma(\mathbf{W}_1 \cdot [\mathbf{t}_r \oplus \mathbf{h}_i^{(l)}] + \mathbf{b}_1) + b_2 \tag{6.7}$$

$$\beta_r = \frac{exp(\beta_r^*)}{\sum_{r\in N(r)} exp(\beta_r^*)} \tag{6.8}$$

where the β_r^* can be seen as the weight for relation type r, and $N(r)$ is the number of entity relation types.

Starting from a bottom layer of node representations as their node features, GCN stacks multiple convolutional operations to simulate the message passing of graphs. We get the final entity embedding $\mathbf{z}_i = \mathbf{h}_i^{(L)}$ for entity e_i after L layers aggregation. Therefore, both the information propagation process with graph structure and node attributes are well leveraged in GCNs. Typically, the embedding fed into the first layer $\mathbf{h}_i^{(0)}$ is chosen as a unique one-hot representation if no other features are present. Here, we further feed this one-hot representation to a single linear transformation and get a dense representation as the first layer entity node feature.

Unsupervised training methods

To learn predictive entity representations, we apply an unsupervised graph-based loss function to the output entity representation in a fully unsupervised way, and tune the parameters via stochastic gradient descent. Specifically, we feed both positive and negative pairs to the loss model, and try to push positive nearby entity to the target entity while pushing negative disparate entity away from it in the

low-dimensional vector space:

$$J_{\mathcal{G}}(\mathbf{z}_u) = -\log\left(\sigma(\mathbf{z}_u^\top \mathbf{z}_v)\right) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log\left(\sigma(-\mathbf{z}_u^\top \mathbf{z}_{v_n})\right), \tag{6.9}$$

where v denotes the paired positive entity that co-occurs in the context of target entity u within fixed-length random walk, and σ is the sigmoid function. In order to sample the negative context entity v_n, we also define sampling distribution P_n based on the node popularity, and Q denotes the number of negative samples.

Item latent Vector

In KG-enhanced recommendation, we assume each item's latent vector representation is consists of two parts: an entity embedding part $\mathbf{Z}$ that could be captured by fruitful facts and connections about items in the knowledge graph. Besides, each item has its unique character, and we use the embedding matrix $\mathbf{P}$ to denote items' individual latent property. The learned aggregated entity embedding $\mathbf{z}_i \in \mathbf{Z}$ could complement the item embedding $\mathbf{p}_i \in \mathbf{P}$, thus organically alleviating sparsity problem by exploiting the extended information of knowledge graph. Then, each item i's latent vector $\mathbf{j}_i$ can be denoted as the sum of these two parts:

$$\mathbf{j}_i = \mathbf{z}_i + \mathbf{p}_i \tag{6.10}$$

6.3.3 Attention-based user modelling

User modelling aims to learn user latent factors, denoted as $\mathbf{k}_u \in \mathbb{R}^d$ for user u, which characterises a user's profile with his/her historically interacted items. After that, we can recommend new items that are highly related to the user's profile. In this section, an item aggregation model is proposed to learn factors from user-item interaction graphs, as shown in the right part in Figure 6.3. To mathematically represent this aggregation, we use the following function as:

$$\mathbf{k}_u = \sigma(\mathbf{W} \cdot Aggre_{items}(\{\mathbf{q}_j, \forall j \in C(u)\}) + \mathbf{b}) \tag{6.11}$$

where $C(u)$ denotes the set of items that user u has interacted with (or user u's neighbours in the user-item graph), $\mathbf{q}_j$ denotes the embedding vector for user interacted item j, and $Aggre_{items}$ is the items aggregation function. In addition, σ denotes non-linear activation function (*i.e.,* a rectified linear unit), and $\mathbf{b}$ and $\mathbf{W}$ are the bias and parameters of a neural network, respectively. It's worth noting that, we encode each item with two embedding vectors $\mathbf{p}$ and $\mathbf{q}$ to differentiate its role of a prediction target or a historical interaction, which can also increase model expressiveness [145]. Next we will discuss how to define the aggregation function $Aggre_{items}$.

Mean operator is one typical aggregation function for $Aggre_{items}$, where we take the element-wise mean of the vectors in $\{\mathbf{q}_j, \forall j \in C(u)\}$. The mean-based aggregator is a linear approximation of a localized spectral convolution [8], as the following function:

$$\mathbf{k}_u = \sigma(\mathbf{W} \cdot \left\{ \sum_{j \in C(u)} \frac{1}{|C(u)|} \mathbf{q}_j \right\} + \mathbf{b}) \tag{6.12}$$

However, the equal treatments on all historical items can limit the representation ability of user profile due to the fact that the influence of interactions on users may vary dramatically. Hence, we should allow interactions to contribute differently to a user's latent factor.

To alleviate the limitation of mean-based aggregator, inspired by attention mechanisms [145, 146], an intuitive method is to tweak a trainable parameter that assigns a unique weight to the target item $\mathbf{p}_i$:

$$\mathbf{k}_u = \sigma(\mathbf{W} \cdot \left\{ \sum_{j \in C(u)} a_{ij} \mathbf{q}_j \right\} + \mathbf{b}) \tag{6.13}$$

$$a_{ij} = f(\mathbf{p}_i, \mathbf{q}_j) \tag{6.14}$$

where a_{ij} denotes the attention weight of the interaction with $\mathbf{q}_j, j \in C(u)$ in contributing to user u's item-space latent factor when characterizing user u's preference from the interaction history $C(u)$. We consider the item attention a_{ij} as a function with $\mathbf{p}_i$ and $\mathbf{q}_j$. The rationale is that the embedding vectors are supposed to encode the information of items, thus they can together to vote the weight of an interaction (i, j). Specifically, we parameterise the item attention a_{ij} with a two-layer neural network, which we call as the attention network:

$$a^*_{ij} = \mathbf{c}^T \cdot \sigma(\mathbf{W}_1 \cdot [\mathbf{q}_j \odot \mathbf{p}_i] + \mathbf{b}_1) + b_2 \tag{6.15}$$

where $\mathbf{W}_1$ is the corresponding input transformation's weight matrix that transform the input pair correlation into a hidden layer, and vector $\mathbf{c}^T$ aims to transform the hidden layer into the output attention weight. Larger value of attention weight means stronger influence to the user preference representation. We use the LeakyReLU nonlinearity (with negative input slope 0.2) as the activation function σ.

To make weight easily comparable across different items, we normalise the above attentive scores across all choice of j with softmax function:

$$\mathbf{k}_u = \sigma(\mathbf{W} \cdot \left\{ \sum_{j \in C(u)} a_{ij} \mathbf{q}_j \right\} + \mathbf{b}) \tag{6.16}$$

$$a_{ij} = \frac{exp(a^*_{ij})}{\sum_{j \in C(u)} exp(a^*_{ij})} \tag{6.17}$$

Then, the normalised attention is used to compute the weighted combination of the item features, to serve as the final output features for each user.

However, this kinds of attention still may encounter performance problem in practice due to the long-tail distribution of user interaction history length (*i.e.,* number of historical items users have interacted). The history length of users can vary in a large ranges as shown in Figure 6.4. Attention mechanisms are commonly used in CV and NLP tasks, where the size of attention units does not vary much, such as regions in an image [147, 148] and words in a sentence [66, 64]. Thus,

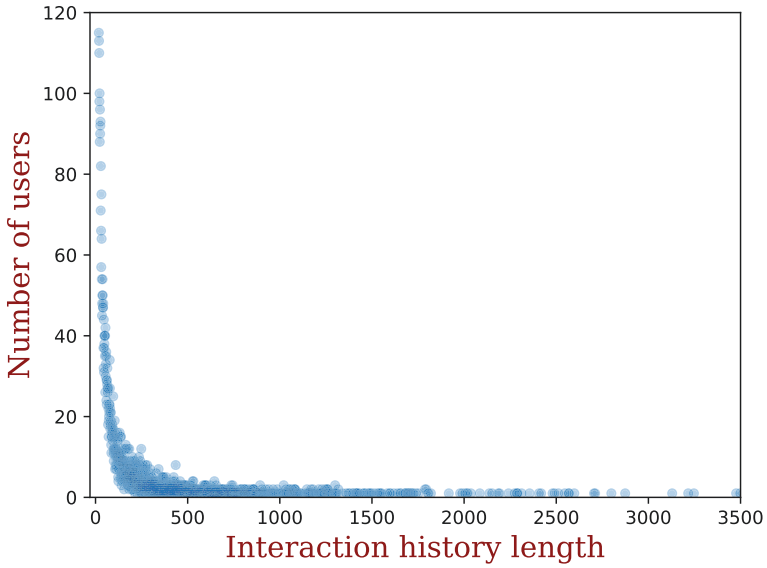


Figure 6.4 : The long-tail distribution of Movielens-20M dataset.

attention weights can be properly normalized by softmax function and successively can reasonablly explained based on the probability. However, when applying in the varying length of recommendation scenarios, the $L1$ normalization of softmax function may overly punish the attention weights of active users with a long history [145]. To address the problem, we leverage exponential smoothing to ease the weights punishment to active users, which can also achieve a variance reduction across all attention weights. Finally, the attention model can be modified to:

$$\mathbf{k}_u = \sigma(\mathbf{W} \cdot \left\{ \sum_{j \in C(u)} \alpha_{ij} \mathbf{q}_j \right\} + \mathbf{b}) \tag{6.18}$$

$$a_{ij} = \frac{exp(a^*_{ij})}{\sum_{j \in C(u)} exp(a^*_{ij})^\beta} \tag{6.19}$$

$$a^*_{ij} = \mathbf{c}^T \cdot \sigma(\mathbf{W}_1 \cdot [\mathbf{q}_j \odot \mathbf{p}_i] + \mathbf{b}_1) + b_2 \tag{6.20}$$

Here, β represents the smoothing exponent, which is a hyperparameter between 0 and 1. When β is smaller than 1, the value of denominator will be suppressed, as a result the attention weights will not be overly punished for active users. It was apparent that, when β is set to 1, it will degenerate to the original softmax function.

6.3.4 Convolutional NCF

After obtaining the latent vector representation of item $\mathbf{j}_i$ and user $\mathbf{k}_u$, another crucial problem is how to model their interaction based on the representation. Traditional shallow NCF models simply concatenate or sum user and item embedding for final prediction, assuming each dimension of the latent space is independent

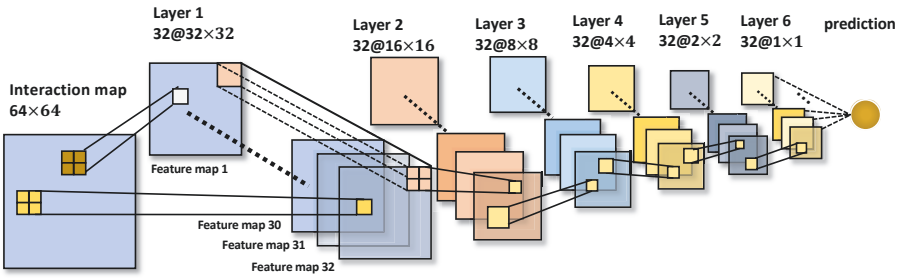


Figure 6.5 : Hidden CNN layers with embedding size 64.

and linearly combined with the same weight for the prediction. In this section, we propose a new convolutional collaboration layer model by integrating the complex correlations between embedding dimensions into modelling. Specifically, we use the outer product above the latent user and item vector, and get a two-dimensional interaction map, which is more expressive and semantically reasonable. The target of this section is to estimate the matching score between user u and item i, (*i.e.,* $\hat{y}_{ui}$); and then we can generate a personalised recommendation list of items for a user based on the scores as shown in upper right in Figure 6.3.

Interaction Map

To obtain the complex correlation matrix, we propose to use an outer product operation on user latent vector $\mathbf{k}_u$ and item latent vector $\mathbf{j}_i$. This correlation matrix is term as the *interaction map* with the dimension of $S \times S$:

$$\mathbf{E} = \mathbf{k}_u \otimes \mathbf{j}_i = \mathbf{k}_u \mathbf{j}_i^T, \tag{6.21}$$

where $\mathbf{E}$ is a $S \times S$ matrix, in which each element is evaluated as: $e_{s_1,s_2} = \mathbf{k}_{u,s_1}\mathbf{j}_{i,s_2}$. Here S is set to 64.

The idea of interaction map is a critical design to ensure the effectiveness of K-NCR for the recommendation. Compared to prior work [46, 149], we argue that outer product can benefit the recommendation from three aspects: 1) interaction map can encode more dimension correlation information than matrix factorisation (MF), due to the fact that MF considers only diagonal elements in the interaction map; 2) interaction map with rich semantics facilitate the following non-linear layers to generalise well on sparse data. and 3) the 2D matrix format of interaction map makes it feasible to learn the interaction function with the effective CNN, which is known to generalise better and is more easily to go deep than the fully connected MLP.

Hidden CNN Layers.

Above the interaction map is a stack of hidden layers, which targets at extracting the useful signal from the interaction map. It is subjected to design and can

be abstracted as $\mathbf{g} = f_\Theta(\mathbf{E})$, where f_Θ denotes the model of hidden layers that has parameters Θ, and $\mathbf{g}$ is the output vector to be used for the final prediction. Here we choose convolutional networks as the hidden layer, which stacks layers in a locally connected manner and utilises much fewer parameters than MLP. The similar technique was originally applied in recommendation for personalized ranking [48]. This allows us to build deeper models than MLP easily, and benefits the learning of high-order correlations among embedding dimensions.

Figure 6.5 shows the model of hidden CNN layers. After getting the interaction map $\mathbf{E}$, we apply 32 kernels of size 2×2 filters to the input interaction map (with size 64×64). Since the stride size is set to 2, the size of feature map c in hidden layer l ($\mathbf{E}^{lc}$) is is half of its previous layer $l-1$. Thus the size of feature maps in the first hidden layer is $32 \times 32 \times 32$. Following the similar convolution operation, we can get the feature maps for the following layers. The output of the 6-th layer is a tensor of dimension $1 \times 1 \times 32$, which can be seen as a vector and is projected to obtain the final prediction score. We use the rectifer unit as the activation function, a common choice in CNN to build deep models. Note that convolution filter can be seen as the "locally connected weight matrix" for a layer, since it is shared in generating all entries of the feature maps of the layer. This significantly reduces the number of parameters of a convolutional layer compared to that of a fully connected layer.

The prediction layer takes in the 6-th layer vector $\mathbf{g}$ and outputs the predicted probability as: $\hat{y}_{ui} = \mathbf{w}^T\mathbf{g}$, where vector $\mathbf{w}$ re-weights the interaction signal in $\mathbf{g}$.

6.3.5 Learning Algorithm

The complete loss function of our model is as follows:

$$\min \mathcal{L} = \alpha\big[-\log\big(\sigma(\mathbf{z}_u^\top \mathbf{z}_v)\big) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log\big(\sigma(-\mathbf{z}_u^\top \mathbf{z}_{v_n})\big)\big] + \sum_{(u,i)\in\mathcal{D}} \mathcal{J}(\hat{y}_{ui}, y_{ui}) \tag{6.22}$$

where the first part captures the KG embedding loss as defined in Equation (6.9), and the second part models the prediction loss ($\mathcal{J}$ is the cross-entropy loss function). α denotes the balance parameter.

In our experiment all model parameters are trained in a joint manner, which is called joint training. Particularly, joint training optimises the parameters in the KG embedding part and the neural collaborative filtering part simultaneously at the training procedure. In this way, the two parts could mutually influence each other and can learn a more general representation for final prediction. Besides, we can also conduct multi-task learning by alternately calling optimiser for each part, where the propagating-based knowledge embedding is learnt first. Then we fine-tune neural collaborative filtering for the predicted rating with the fixed parameter in first part [150]. Actually, the second manner works as the pre-training and fine tuning of the deep learning [151]. In this alternative approach, the rating prediction process entirely relies on the KG embedding part, while the KG embedding part could not

Dataset	MovieLens-20M	Book-Crossing	Last.FM
# users	114,571	18,390	1,872
# items	13,482	16,374	3,846
# interactions	10,653,587	143,630	42,346
# entities	93,323	31,227	10,828
# KG triples	397,698	61,175	16,753
# relations	32	18	60

Table 6.2 : Basic statistics settings for the three datasets

benefit from rating information. We would show the superiority of joint training in the experimental part.

6.4 Experiment

6.4.1 Datasets

We use the following three datasets in our experiments for movie, book and music recommendation, respectively:

- **MovieLens-20M**[†] dataset contains almost 20 million ratings (ranging from 1 to 5) on the MovieLens website, which is a widely used benchmark dataset in movie recommendations.
- **Book-Crossing**[‡] dataset contains 1 million explicit ratings (ranging from 0 to 10) of books in the Book-Crossing community.
- **Last.FM**[§] dataset contains music artist listening information from a set of 2 thousand users at Last.fm online music system. Last.FM provides the listening count as the weight for each user-item interaction.

Following most existing item recommendation methods, designed to explore implicit feedback, we transform explicit feedback of all three datasets into implicit one. Specifically, each implicit feedback is marked as 1, which denotes a positive rate, while we also randomly sample negative examples from the unknown items. By following existing practices [139], the threshold of positive rating for MovieLens-20M is set to 4, while all ratings in Book-Crossing and Last.FM are set to positive sample due to their sparsity. To construct the knowledge graph for a specific task from DBPedia[¶](Release 2016-10), we retrieve related triplet facts from DBpedia using SPARQL queries. We consider the mapping from item indices in the raw rating file to entity IDs in the KG. Then, we extract the triplets that are directly related to the

[†]https://grouplens.org/datasets/movielens/20m/

[‡]http://www2.informatik.uni-freiburg.de/ cziegler/BX/

[§]https://grouplens.org/datasets/hetrec-2011/

[¶]https://wiki.dbpedia.org/

entities aligned with items, no matter which role (i.e., subject or object) it serves as. Finally, we construct a new edge for each triplet connecting two entities. The basic statistics of the three datasets are presented in Table 6.2. For simplicity, items with no matched or multiple matched entities are excluded. Besides, we only consider those triplets that are directly related to the entities with the mapped items, no matter which role (*i.e.,* head or tail) the entity serves as.

6.4.2 Baselines

We compare the proposed K-NCR with two kinds of representative recommendation methods. One is CF-based methods, which only leverage implicit user-item interaction information. Another is the KG-based methods, which incorporate extra KG.

- **LibFM** [152] is a commonly used feature-based factorisation model for Click Through Rate (CTR) prediction. Only user ID and item ID are concatenated together as the input for LibFM.
- **LibFM + TransE**. We concatenate the raw features of users and items, as well as the corresponding averaged entity embeddings learned from TransE [16] as input for LibFM. The dimension of TransE is 32.
- **NCF** [46] is a neural network architectures for collaborative filtering. It replaces the inner product with a neural architecture to learn an arbitrary matching function from data.
- **CKE** [39] is a typical embedding-based recommendation methods, which exploit items' semantic embedding from structural, textual and visual content to assist CF prediction. For a fair comparison, we implement CKE with only additional structural knowledge information. The dimension of entity embeddings is 32.
- **PER** [32] regards the KG as a special case of heterogeneous information networks (HIN) and exploits meta-path based representation to model the rich connectivity between users and items. Specifically, we manually design "user-item-attribute-item" meta-path as similarity feature.
- **Wide&Deep** [57] is a general deep model for recommendation combining a (wide) linear channel with a (deep) nonlinear channel. Similar to LibFM + TransE, we use the embeddings of users, items, and entities to feed Wide&Deep. The input for Wide&Deep is the same as in LibFM + TransE. The dimension of user, item, and entity is 64, and we use a two-layer deep channel with dimension of 100 and 50 as well as a wide channel.
- **RippleNet** [38] is a knowledge graph embedding-based method that propagates and aggregates user potential preference on the KG for the recommendation.

Table 6.3 : The results of *AUC* and *F*1 in CTR prediction.

Model	MovieLens-20M		Book-Crossing		Last.FM	
	AUC	F1	AUC	F1	AUC	F1
LibFM	0.921 (-5.3%)	0.878 (-5.7%)	0.629 (-14.0%)	0.598 (-16.0%)	0.694 (-8.6%)	0.648 (-7.7%)
LibFM+TransE	0.928 (-4.6%)	0.889 (-4.5%)	0.649 (-11.2%)	0.613 (-13.9%)	0.709 (-6.6%)	0.652 (-7.1%)
NCF	0.925 (-4.9%)	0.884 (-5.0%)	0.642 (-12.2%)	0.597 (-16.2%)	0.704 (-7.2%)	0.646 (-8.0%)
PER	0.794 (-18.4%)	0.753 (-19.1%)	0.593 (-18.9%)	0.577 (-19.0%)	0.603 (-20.6%)	0.568 (-19.1%)
CKE	0.887 (-8.8%)	0.842 (-9.6%)	0.622 (-14.9%)	0.608 (-14.6%)	0.684 (-9.9%)	0.643 (-8.4%)
W&D	0.932 (-4.2%)	0.891 (-4.3%)	0.684 (-6.4%)	0.645 (-9.4%)	0.716 (-5.7%)	0.654 (-6.8%)
RippleNet	0.947 (-2.7%)	0.896 (-3.8%)	0.697 (-4.7%)	0.651 (-8.6%)	0.724 (-4.6%)	0.675 (-3.8%)
KGCN	0.955 (-1.8%)	0.913 (-1.9%)	0.705 (-3.6%)	0.684 (-3.9%)	0.736 (-3.0%)	0.687 (-2.1%)
NCR	0.923 (-5.1%)	0.887 (-4.7%)	0.650 (-11.1%)	0.598 (-16.0%)	0.695 (-8.4%)	0.664 (-5.4%)
KCR	0.952 (-2.2%)	0.894 (-4.0%)	0.694 (-5.1%)	0.654 (-8.1%)	0.729 (-4.0%)	0.682 (-2.8%)
K-NCR_A	0.961 (-1.2%)	0.918 (-1.4%)	0.704 (-3.7%)	0.678 (-4.8%)	0.738 (-2.8%)	0.684 (-2.6%)
K-NCR_J	**0.973**	**0.931**	**0.731**	**0.712**	**0.759**	**0.702**

Figure 6.6 : Computational cost comparison.

- **KGCN** [44] extends non-spectral GCN approaches to the knowledge graph by aggregating neighbourhood information, which captures inter-item relatedness by mining their associated attributes on the KG. Specifically, we use variant KGCN-sum as our baseline.

Besides, to examine the effectiveness of incorporated knowledge graph, we prepare two simplified variants of K-NCR. **KCR** does not use interaction map and attention based user preference modelling in the collaboration layer. **NCR** is another variant that does not incorporate the Knowledge Graph for the recommendation. We also use two training methods for K-NCR, a joint training approach named K-NCR_J and an alternative training approach as K-NCR_A.

6.4.3 Experiment Settings

In K-NCR, we set the dimension of user and item latent vector as $S = 64$, and the number of CNN hidden layers is set as 6. The dropout rate is set as $\rho = 0.2$ for default, which are determined by optimising AUC on a validation set. For the knowledge propagation item modelling, we set $ReLU$ for non-last-layer aggregator, and $tanh$ for last-layer aggregator. We use open-source implementation of baselines to obtain performance result in all three datasets.

The ratio of training, validation and test for each dataset is set as 6:2:2. After random choosing a set of data (such as test subset), we will filter out this test subset from the rating list. And then we make the next random selection from the rest of ratings. Thus, different partition (training, validation, test) do not overlap with each other. And we choose the subset all at once, not choose rating one by one. In this way, each rating will only be chosen once. We repeat each experiment for 10 times, and report the average performance. For the other comparison methods, we optimise their parameters in the validation set. We evaluate our method in click-through rate (CTR) prediction scenarios. For CTR recommendation, the threshold of positive rating is set to 0.5. If $\hat{y}_{ui} >= 0.5$, we will recommend item i to user u. We apply the trained model to each piece of interactions in the test set and output the predicted click probability. Average AUC (Area under the ROC Curve) and $F1$

on test sets are used as the evaluation criterion for CTR prediction. Statistically significant improvement by unpaired two-sample t-test is with p = 0.05.

6.4.4 Overall Performance

Comparison with baselines

The results of all methods in CTR prediction is presented in Table 6.3. The : computational cost comparison results are shown in Fig 6.6. We have the following observations:

- In all cases, our proposed K-NCR outperform all the baseline methods on all the datasets. In particular, K-NCR improves over the strongest baselines *w.r.t* AUC by 1.8%, 3.6%, and 3.0% in MovieLens-20M, Book-Crossing, and Last-FM, respectively. By stacking multiple relation-aware attentive embedding propagation layers, K-NCR is capable of exploring the high-order connectivity of knowledge graph in an explicit way, so as to capture collaborative item relation signal effectively.

- CF-based methods that without using any KG information (*i.e.* LibFM and NCF) actually achieve better performance than the two KG-aware baselines CKE and PER in most cases. This indicates that the hand-crafted meta-paths can not fully exploit the complex semantic interaction in knowledge graph, and user-defined meta-paths can also hardly be optimal in reality.

- LibFM + TransE outperforms LibFM, which demonstrates the benefit of the introduction of auxiliary KG for recommendation in general. As a generic recommendation tool, Wide&Deep method achieve satisfactory performance, demonstrating that they can make well use of knowledge from KG into their algorithms.

- It is noteworthy that the recently proposed RippleNet model and KGCN work well among these baselines. Both RippleNet and KGCN are Structure-based methods that use breadth-first-search to obtain multi-hop neighbours of an entity in the KG. RippleNet adopts a similar preference propagation approach to exploit the potential preference in KG, which shows that capturing proximity information in the KG is essential for the recommendation. KGCN is a representative of inward aggregation that learns the representation of an entity by aggregating information from its multi-hop neighbours to mine users' potential preferences. However, both RippleNet and KGCN can hardly capture the complex pairs interaction between users and items.

- Training takes up significantly more time compared to testing. Thus, we only report the training time here. It can be observed that simpler methods take less training time. LibFM is most efficient in terms of training, while KG-based methods all require the longer training time. Nevertheless, our method is still very efficient, and can be trained in less than seven minutes on the largest dataset Movielens-20M.

Model	r				
	20%	40%	60%	80%	100%
LibFM	0.725	0.814	0.846	0.892	0.921
LibFM+TransE	0.773	0.834	0.856	0.904	0.928
NCF	0.757	0.816	0.847	0.882	0.925
PER	0.638	0.693	0.724	0.769	0.794
CKE	0.725	0.784	0.831	0.857	0.887
W&D	0.769	0.839	0.876	0.914	0.932
RippleNet	0.852	0.873	0.886	0.922	0.947
KGCN	0.863	0.881	0.893	0.928	0.955
K-NCR_J	**0.904**	**0.934**	**0.951**	**0.962**	**0.973**

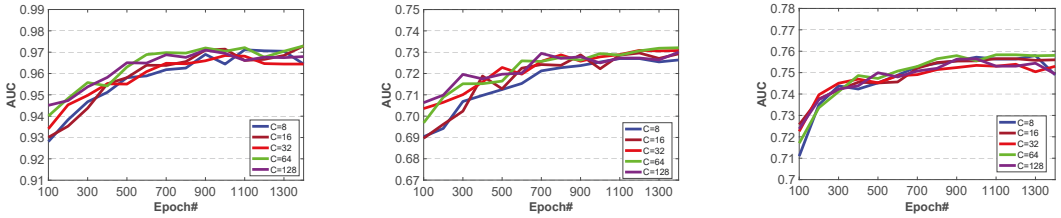
Table 6.4 : Results of *AUC* on MovieLens-20M in CTR prediction with different ratios of training set r.

Comparison with model variants.

We compare K-NCR with our proposed two simplifications. We can see from the table 6.3 that removing knowledge graph embedding and attention components degrades the model's performance. Specifically, KCR justifies the effectiveness of the attention mechanism and interaction map in interaction matching function modelling, specifying the attentive weights w.r.t. compositional items. Also, incorporating the knowledge graph embedding could further alleviate the sparsity problem and improve prediction performance in K-NCR. Besides, NCR (without KG) achieve better performance than NCF (Neural Collaborative Filtering). Because the proposal of using outer product above the embedding layer results in a two-dimensional interaction map that is more expressive and semantically plausible, which can learn high-order correlations among embedding dimensions. When comparing the two training variants of proposed K-NCR with different training strategy, K-NCR_J consistently achieve better performance than K-NCR_A with about 1.2% to 5% improvement. This experimental result shows that the joint training methods can exploit shared features across two domains (user-item interaction and extra KG), which can mutually benefit both the KG embedding part and the NCR part.

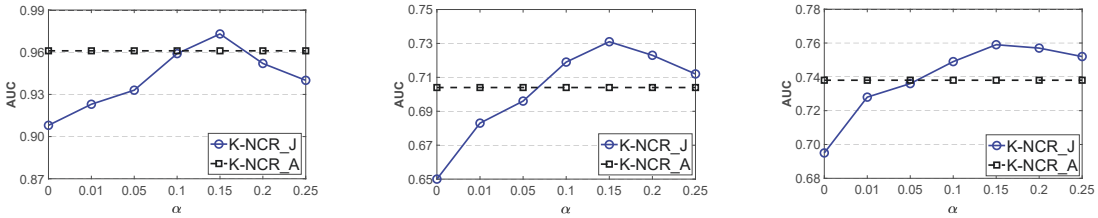
Results in sparse scenarios

One major advantage of incorporating knowledge graph in K-NCR is to alleviate the sparsity problem of recommender systems. To study the effect of the knowledge graph embedding module in sparse scenarios, we vary the ratio of MovieLens-20M training set from $r = 20\%$ to $r = 100\%$ (while the validation and test set are kept fixed), and show the CTR prediction performance with *AUC*. The reason for choosing MovieLens-20M dataset is that MovieLens-20M have relatively sufficient training data than the other two, which make it easier in test to simulate the sparse scenario. We show the final results in Table 6.4. With the reducing of the training set, we observe that the performance of all methods deteriorate. When $r = 20\%$, the *AUC* score decreases by 21.3%, 16.7.2%, 18.2%, 19.6%, 18.3%, 17.5%, 10.0%



(a) **MovieLens-20M** (b) **Book-Crossing** (c) **Last.FM**

Figure 6.7 : Performance of K-NCR *w.r.t.* different numbers of feature maps per convolutional layer (denoted by C) in each epoch.



(a) **MovieLens-20M** (b) **Book-Crossing** (c) **Last.FM**

Figure 6.8 : Impact of parameter α.

and 9.6% for the baselines compared with the case when full training set is used ($r = 100\%$). In contrast, the AUC score of K-NCR only decreases by 7.1%, which shows that K-NCR can still maintain a decent performance even when the user-item interaction is sparse.

6.4.5 Parameter Sensitivity

In this section, we investigate how the recommendation performance varies with the hyper-parameters in (1) impact of the Number of Feature Maps, (2) impact of tradeoff parameter α and (3) the impact of KG propagating layers.

Impact of the Number of Feature Maps

The number of feature maps in each CNN layer decides the dimension of final prediction layer, and will finally affect the representation ability of our K-NCR. Figure 6.7 shows the performance of K-NCR with respect to different numbers of feature maps. We can see that although there are some slight differences in the convergence curve, all the curves increase steadily and finally achieve similar performance. This reflects the strong expressiveness and generalisation of using CNN under the K-NCR framework since dramatically increasing the number of parameters of a neural network does not lead to overfitting.

Impact of Tradeoff Parameter

There is an important balance parameter α in the final loss function (6.22), which decides the weights between the knowledge graph embedding loss and the

Table 6.5 : Effect of embedding propagation layer numbers (L).

	MovieLens-20M		Book-Crossing		Last.FM	
	AUC	F1	AUC	F1	AUC	F1
K-NCR-1	0.892	0.872	0.693	0.683	**0.759**	**0.702**
K-NCR-2	**0.973**	**0.931**	**0.731**	**0.712**	0.743	0.683
K-NCR-3	0.968	0.909	0.725	0.706	0.647	0.589
K-NCR-4	0.726	0.653	0.524	0.593	0.424	0.448

prediction loss. The larger value of α will put more weight on the KG embedding loss. We show the results with the varying parameter α in Figure 6.8. For K-NCR_A, we first learn the KG embedding and then incorporate the KG embedding in neural collaborative part, which makes it not sensitive to α. For K-NCR_J, the performance shows an apparent increase at first, however, the performance then starts to decrease, when we continuously increase the balance over $\alpha = 0.15$ on all datasets. The reason is that, when $\alpha = 0$, K-NCR_J degenerates to the NCR without any knowledge graph information. When we increase α from 0, we actually strengthen the KG embedding for item modelling. However, a too large value of α would drive the objective function bias to the KG embedding learning. Given the experimental finding, we set $\alpha = 0.15$ for all three datasets.

Impact of KG propagating layers

We also vary the size of local knowledge graph context neighbourhoods by the maximal layer number L and observe the performance changes. The results are listed in Table 6.5, which clearly shows that an L with 1 or 2 is enough for real cases. Specifically, the relative smaller scale of Last.FM data need only one layer of neighbours to achieve best performance. We attribute this phenomenon to the trade-off between the useful information from long-distance dependency neighbours and useless information from noise: (1) a too-small layer value L is unable to adequately explore deep interdependency and gain enough context information, which is carried by the second- or third-order connectivities in larger scale graphs; (2) a too-large value of L may lead to overfitting problem as the number of parameters increases, while at the same time bringing in more undesired noise from neighbours. When the number of layers is $L = 1$, representation of each entity is a mixture of itself and its immediate neighbours. The experiment results indicate that the discrimination of the nodes is decreasing as the number of layers increases. We suggest that the increasing layers makes the neighbours of nodes more similar, and further makes node representations more similar or had a low degree of differentiation. The neighbour size grows exponentially with an increase of L-layers, so might introduce noise, and consequently decrease accuracy.

6.5 Conclusion

In this chapter, we develop an end-to-end neural architecture K-NCR that jointly incorporates the knowledge graph structure and user-item interaction in a unified neural network model for recommendation. Specifically, we propose a propagation based knowledge graph embedding model that exploits the high-order structural proximity among entities in KG to enhance the item embedding learning in a unified framework. Besides, an attention-based user modelling part is proposed to learn the varying importance weights of the interacted items. We also design an interaction map based collaboration layer to model the complex user-item interaction behaviour in recommender systems. All the parameters in K-NCR are optimised in a joint manner. Thus, the proposed model can capture both the enriched semantic information and the complex hidden relationships between users and items for recommendation. In the meantime, the high-order connectivity of KG is seamlessly incorporated into this recommendation framework. Finally, extensive experimental results on three real-world datasets clearly show the effectiveness of our proposed model.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The information explosion caused by the booming Internet makes it difficult for users to obtain accurate and valid information. Personalized recommendation attempts to mine user preferences and product characteristics, so as to find a small number of related products in line with users' personalized interests from a large number of choices. Nowadays, there are more and more diversified auxiliary information of users and items in online platforms. Combining these auxiliary information with collaborative reasoning can better capture the potential complex relationship between users and items and further achieve higher recommendation quality. Generally speaking, there are many kinds of graph structure data in the recommendation system, such as the interaction graph between users and items, the social network of users, the knowledge graph of items, and so on. In practice, we can't directly use the original representation of grpah data to assist the recommendation system, so we need to learn the feature representation of network structure data first. Therefore, this thesis considers using the method of graph feature learning to process the graph structure data in the recommendation system. The main work of this thesis is concluded as follows:

(1)To learn graph embedding with multiple scales, we propose an effective framework to let different graph scales collaborate with each other and vote for the robust node representations. During voting, we propose an attention-based autoencoder to automatically learn the voting weights of scales while preserving the network structure information in a non-linear way. Besides, an Adversarial regularization is introduced to learn more stable and robust network embedding.

(2) To mine the semantic of social network assisted community question answering, we propose a novel multi-modal multi-view semantic embedding learning method for community question answer analysis. The proposed MMSE can simultaneously model multi-modal CQA data as well as their corresponding semantic information from both local and global view in a unified and principled way. Experiments conducted on question answering task for both English and Chinese CQA datasets demonstrate the effectiveness of our approaches.

(3) To exploit both the graph structure and video content of the video recommendation, we propose the CDPRec, a neural framework that incorporates HINs and multimodal video content into recommender systems in a natural and intuitive manner. The CDPRec overcomes the limitations of path-based similarity methods by introducing context-dependent propagation, which automatically propagates the

potential preferences of the users and explores their hierarchical interdependencies in the HIN. The CDPRec encodes the global video context into comprehensive video embedding to facilitate the click-through rate prediction. Experimental results on real-world YouTube data demonstrate the significant superiority of our method over strong baselines.

(4) To incorporates the knowledge graph structure and user-item interaction in a unified neural network model for recommendation, we develop an end-to-end neural architecture K-NCR that jointly incorporates the knowledge graph structure and user-item interaction in a unified neural network model for recommendation. Specifically, we propose a propagation based knowledge graph embedding model that exploits the high-order structural proximity among entities in KG to enhance the item embedding learning in a unified framework. Besides, an attention-based user modelling part is proposed to learn the varying importance weights of the interacted items. We also design an interaction map based collaboration layer to model the complex user-item interaction behaviour in recommender systems. All the parameters in K-NCR are optimised in a joint manner.

7.2 Future Work

While many progress in both graph representation learning and recommendation have been made, there remain a number of directions in our future research, some of which we'll discuss below.

(1) In addition to the combination of traditional collaborative filtering, one direction of future work is to study how to use auxiliary network structure data for other types of recommendation, Such as sequence recommendation, focusing on the change of user preference. That is to say, when a new user-item interaction comes in, user preferences should be updated in time. We hope to express users explicitly with auxiliary data, and adopt sequential neural model to simulate the change of dynamic user preferences with time.

(2) Cold-start recommendation has been a challenging problem due to sparse user-item interactions for new users or items. Existing efforts have alleviated the cold-start issue to some extent, most of which approach the problem at the data level. For example, earlier methods often incorporate auxiliary data as user or item features, while other methods leverage heterogeneous information networks (HIN) to capture richer semantics via higher-order graph structures. Recently, meta-learning paradigm sheds light on addressing cold-start recommendation at the model level, given its ability to rapidly adapt to new tasks with scarce labeled data, or in the context of cold-start recommendation, new users and items with very few interactions. Thus, we are inspired to develop a novel meta-learning approach to address cold-start recommendation on HINs.

Bibliography

[1] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014.

[2] R. Zhou, S. Khemmarat, and L. Gao, "The impact of youtube recommendation system on video views," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*. ACM, 2010, pp. 404–410.

[3] M. Yan, J. Sang, and C. Xu, "Mining cross-network association for youtube video promotion," in *Proceedings of the 22nd ACM International Conference on Multimedia*. ACM, 2014, pp. 557–566.

[4] [online], "http://tubularinsights.com/hours-minute-uploaded-youtube/."

[5] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: statistical mechanics and its applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[6] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding." in *The Association for the Advancement of Artificial Intelligence (AAAI)*, 2017, pp. 203–209.

[7] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web(WWW)*, 2015, pp. 1067–1077.

[8] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[9] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems(NIPS)*, 2017, pp. 1024–1034.

[10] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *International Conference on Learning Representations(ICLR)*, 2018. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ

[11] C. Tu, W. Zhang, Z. Liu, and M. Sun, "Max-margin deepwalk: Discriminative learning of network representation." in *International Joint Conferences on Artificial Intelligence(IJCAI)*, 2016, pp. 3889–3895.

[12] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information." in *International Joint Conferences on Artificial Intelligence(IJCAI)*, 2015, pp. 2111–2117.

[13] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 135–144.

[14] T.-y. Fu, W.-C. Lee, and Z. Lei, "Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management(CIKM)*. ACM, 2017, pp. 1797–1806.

[15] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[16] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, 2013, pp. 2787–2795.

[17] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes." in *AAAI*, vol. 14, 2014, pp. 1112–1119.

[18] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[19] M. Nickel, L. Rosasco, T. A. Poggio *et al.*, "Holographic embeddings of knowledge graphs." in *AAAI*, 2016, pp. 1955–1961.

[20] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data." in *ICML*, vol. 11, 2011, pp. 809–816.

[21] H. Liu, Y. Wu, and Y. Yang, "Analogical inference for multi-relational embeddings," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2168–2178.

[22] H. Zhong, J. Zhang, Z. Wang, H. Wan, and Z. Chen, "Aligning knowledge and text embeddings by entity descriptions," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 267–272.

[23] R. Xie, Z. Liu, and M. Sun, "Representation learning of knowledge graphs with hierarchical types." in *IJCAI*, 2016, pp. 2965–2971.

[24] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[25] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[26] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5115–5124.

[27] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 3189–3197.

[28] X. Chen, L.-J. Li, L. Fei-Fei, and A. Gupta, "Iterative visual reasoning beyond convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7239–7248.

[29] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," *Proceedings of the 24rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018.

[30] Y. Wu, H. Liu, and Y. Yang, "Graph convolutional matrix completion for bipartite edge prediction," in *International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (KDIR)*, 2018, pp. 51–60.

[31] L. Gao, H. Yang, J. Wu, C. Zhou, W. Lu, and Y. Hu, "Recommendation with multi-source heterogeneous information," in *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

[32] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han, "Personalized entity recommendation: A heterogeneous information network approach," in *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, 2014, pp. 283–292.

[33] H. Zhao, Q. Yao, J. Li, Y. Song, and D. L. Lee, "Meta-graph based recommendation fusion over heterogeneous information networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 635–644.

[34] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 992–1003, 2011.

[35] X. Yu, X. Ren, Y. Sun, B. Sturt, U. Khandelwal, Q. Gu, B. Norick, and J. Han, "Recommendation in heterogeneous information networks with implicit user feedback," in *Proceedings of the 7th ACM conference on Recommender systems*, 2013, pp. 347–350.

[36] C. Shi, Z. Zhang, P. Luo, P. S. Yu, Y. Yue, and B. Wu, "Semantic path based personalized recommendation on weighted heterogeneous information networks," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management(CIKM)*. ACM, 2015, pp. 453–462.

[37] B. Hu, C. Shi, W. X. Zhao, and P. S. Yu, "Leveraging meta-path based context for top-n recommendation with a neural co-attention model," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1531–1540.

[38] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo, "Ripple network: Propagating user preferences on the knowledge graph for recommender systems," *arXiv preprint arXiv:1803.03467*, 2018.

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

[40] Z. Zhao, Q. Yang, H. Lu, T. Weninger, D. Cai, X. He, and Y. Zhuang, "Social-aware movie recommendation via multimodal network learning," *IEEE Transactions on Multimedia*, vol. 20, no. 2, pp. 430–440, 2018.

[41] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 855–864.

[42] H. Wang, F. Zhang, M. Hou, X. Xie, M. Guo, and Q. Liu, "Shine: Signed heterogeneous information network embedding for sentiment link prediction," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 2018, pp. 592–600.

[43] H. Wang, F. Zhang, X. Xie, and M. Guo, "Dkn: Deep knowledge-aware network for news recommendation," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2018, pp. 1835–1844.

[44] H. Wang, M. Zhao, X. Xie, W. Li, and M. Guo, "Knowledge graph convolutional networks for recommender systems," in *The world wide web conference*, 2019, pp. 3307–3313.

[45] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, and Z. Wang, "Knowledge-aware graph neural networks with label smoothness regularization for recommender systems," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 968–977.

[46] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide*

Web. International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.

[47] Z.-H. Deng, L. Huang, C.-D. Wang, J.-H. Lai, and P. S. Yu, "Deepcf: A unified framework of representation learning and matching function learning in recommender system," *arXiv preprint arXiv:1901.04704*, 2019.

[48] X. He, X. Du, X. Wang, F. Tian, J. Tang, and T.-S. Chua, "Outer product-based neural collaborative filtering," *arXiv preprint arXiv:1808.03912*, 2018.

[49] S. Funk, "Netflix update: Try this at home," *https://sifter.org/simon/journal/20061211.html. accessed 12-June-2019*, 2006.

[50] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 880–887.

[51] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[52] H. Ma, "An experimental study on implicit social recommendation," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, 2013, pp. 73–82.

[53] L. Hu, A. Sun, and Y. Liu, "Your neighbors affect your ratings: on geographical neighborhood influence to rating prediction," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, 2014, pp. 345–354.

[54] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 111–112.

[55] H.-J. Xue, X.-Y. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, ser. IJCAI'17, 2017.

[56] T. Bai, J.-R. Wen, J. Zhang, and W. X. Zhao, "A neural collaborative filtering model with interaction-based neighborhood," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 1979–1982.

[57] H. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide & deep learning for recommender systems," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016*, 2016, pp. 7–10.

[58] L. Sang, M. Xu, S. Qian, and X. Wu, "Multi-modal multi-view bayesian semantic embedding for community question answering," *Neurocomputing*, 2018.

[59] B. Perozzi, V. Kulkarni, H. Chen, and S. Skiena, "Don't walk, skip!: Online learning of multi-scale network embeddings," in *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. ACM, 2017, pp. 258–265.

[60] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 891–900.

[61] ——, "Deep neural networks for learning graph representations." in *AAAI*, 2016, pp. 1145–1152.

[62] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 20th ACM SIGKDD*, 2016.

[63] M. Qu, J. Tang, J. Shang, X. Ren, M. Zhang, and J. Han, "An attention-based collaboration framework for multi-view network representation learning," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 1767–1776.

[64] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[65] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[66] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, and C. Zhang, "Disan: Directional self-attention network for rnn/cnn-free language understanding," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[67] J. Weston, S. Bengio, and N. Usunier, "Wsabie: Scaling up to large vocabulary image annotation," in *IJCAI*, vol. 11, 2011, pp. 2764–2770.

[68] M. Iyyer, A. Guha, S. Chaturvedi, J. Boyd-Graber, and H. Daumé III, "Feuding families and former friends: Unsupervised learning for dynamic fictional relationships," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1534–1544.

[69] R. He, W. S. Lee, H. T. Ng, and D. Dahlmeier, "An unsupervised neural attention model for aspect extraction," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2017, pp. 388–397.

[70] Q. Dai, Q. Li, J. Tang, and D. Wang, "Adversarial network embedding," in *In Proceedings of AAAI*, 2018.

[71] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871–1874, 2008.

[72] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[73] K. Zhang, W. Wu, F. Wang, M. Zhou, and Z. Li, "Learning distributed representations of data in community question answering for question retrieval," in *WSDM*, 2016.

[74] X. Qiu and X. Huang, "Convolutional neural tensor network architecture for community-based question answering." in *IJCAI*, 2015.

[75] Z. Zhao, Q. Yang, D. Cai, X. He, and Y. Zhuang, "Expert finding for community-based question answering via ranking metric network learning." in *IJCAI*, 2016.

[76] T. Sagara and M. Hagiwara, "Natural language neural network and its application to question-answering system," *Neurocomputing*, vol. 142, pp. 201–208, 2014.

[77] H. Fang, F. Wu, Z. Zhao, X. Duan, Y. Zhuang, and M. Ester, "Community-based question answering via heterogeneous social network learning," in *AAAI*, 2016.

[78] R. Das, M. Zaheer, and C. Dyer, "Gaussian lda for topic models with word embeddings." in *ACL*, 2015.

[79] Z. Yang, J. Tang, and W. W. Cohen, "Multi-modal bayesian embeddings for learning social knowledge graphs." in *IJCAI*, 2016.

[80] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.

[81] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Heidelberg: Springer, 2006.

[82] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv*, 2013.

[83] W. Hu, J. Zhang, and N. Zheng, "Different contexts lead to different word embeddings," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 762–771.

[84] L. Vilnis and A. McCallum, "Word representations via gaussian embedding," in *ICLR*, 2015.

[85] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *JMLR*, vol. 3, no. Jan, pp. 993–1022, 2003.

[86] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *Proceedings of the National academy of Sciences*, vol. 101, no. suppl 1, pp. 5228–5235, 2004.

[87] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *ICML*, 2015.

[88] J. Guo, Y. Fan, Q. Ai, and W. B. Croft, "Semantic matching by non-linear word transportation for information retrieval," in *CIKM*, 2016.

[89] Z. Zhao, L. Zhang, X. He, and W. Ng, "Expert finding for question answering via graph regularized matrix completion," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 4, pp. 993–1004, 2015.

[90] Y. Shen, W. Rong, Z. Sun, Y. Ouyang, and Z. Xiong, "Question/answer matching for cqa system via combining lexical and sequential information." in *AAAI*, 2015, pp. 275–281.

[91] Z. Lu and H. Li, "A deep architecture for matching short texts," in *Advances in Neural Information Processing Systems*, 2013, pp. 1367–1375.

[92] S. Robertson, H. Zaragoza *et al.*, "The probabilistic relevance framework: Bm25 and beyond," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[93] C. Zhai and J. Lafferty, "A study of smoothing methods for language models applied to ad hoc information retrieval," in *SIGIR*, 2017.

[94] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *ICML*, 2014.

[95] D. Ganguly, D. Roy, M. Mitra, and G. J. Jones, "Word embedding based generalized language model for information retrieval," in *SIGIR*, 2015.

[96] Y. Liu, Z. Liu, T.-S. Chua, and M. Sun, "Topical word embeddings." in *AAAI*, 2015.

[97] H. Wu, W. Wu, M. Zhou, E. Chen, L. Duan, and H.-Y. Shum, "Improving search relevance for short queries in community question answering," in *WSDM*, 2014.

[98] S. Yuan, Y. Zhang, J. Tang, and J. B. Cabotà, "Expert finding in community question answering: A review," *arXiv preprint arXiv:1804.07958*, 2018.

[99] Y. Kim, "Convolutional neural networks for sentence classification," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[100] J. Zhang, M. S. Ackerman, and L. Adamic, "Expertise networks in online communities: structure and algorithms," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 221–230.

[101] M. Bouguessa, B. Dumoulin, and S. Wang, "Identifying authoritative actors in question-answering forums: the case of yahoo! answers," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 866–874.

[102] F. Wu, X. Lu, J. Song, S. Yan, Z. M. Zhang, Y. Rui, and Y. Zhuang, "Learning of multimodal representations with random walks on the click graph," *IEEE Transactions on Image Processing*, vol. 25, no. 2, pp. 630–642, 2016.

[103] J. Guo, S. Xu, S. Bao, and Y. Yu, "Tapping on the potential of q&a community by recommending answer providers," in *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008, pp. 921–930.

[104] L. Sun, X. Wang, Z. Wang, H. Zhao, and W. Zhu, "Social-aware video recommendation for online social groups." *IEEE Transactions on Multimedia*, vol. 19, no. 3, pp. 609–618, 2017.

[105] Z. Wang, L. Sun, W. Zhu, S. Yang, H. Li, and D. Wu, "Joint social and content recommendation for user-generated videos in online social network," *IEEE Transactions on Multimedia*, vol. 15, no. 3, pp. 698–709, 2013.

[106] T. Mei, B. Yang, X.-S. Hua, and S. Li, "Contextual video recommendation by multimodal relevance and user feedback," *ACM Transactions on Information Systems (TOIS)*, vol. 29, no. 2, p. 10, 2011.

[107] Y. Huang, B. Cui, J. Jiang, K. Hong, W. Zhang, and Y. Xie, "Real-time video recommendation exploration," in *Proceedings of the 2016 International Conference on Management of Data*. ACM, 2016, pp. 35–46.

[108] F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: introduction and challenges," in *Recommender Systems Handbook*. Springer, 2015, pp. 1–34.

[109] S. D. Roy, T. Mei, W. Zeng, and S. Li, "Towards cross-domain learning for social video popularity prediction," *IEEE Transactions on multimedia*, vol. 15, no. 6, pp. 1255–1267, 2013.

[110] B. Chen, J. Wang, Q. Huang, and T. Mei, "Personalized video recommendation through tripartite graph propagation," in *Proceedings of the 20th ACM International Conference on Multimedia*. ACM, 2012, pp. 1133–1136.

[111] J. Zhang, Y. Yang, Q. Tian, L. Zhuo, and X. Liu, "Personalized social image recommendation method based on user-image-tag model," *IEEE Transactions on Multimedia*, vol. 19, no. 11, pp. 2439–2449, 2017.

[112] C. Shi, B. Hu, X. Zhao, and P. Yu, "Heterogeneous information network embedding for recommendation," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[113] X. Yu, X. Ren, Q. Gu, Y. Sun, and J. Han, "Collaborative filtering with entity similarity regularization in heterogeneous information networks," *International Joint Conferences on Artificial Intelligence(IJCAI)*, vol. 27, 2013.

[114] Q. Huang, B. Chen, J. Wang, and T. Mei, "Personalized video recommendation through graph propagation," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 10, no. 4, p. 32, 2014.

[115] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 2004.

[116] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston *et al.*, "The youtube video recommendation system," in *Proceedings of the fourth ACM Conference on Recommender systems*. ACM, 2010, pp. 293–296.

[117] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1235–1244.

[118] P. Cui, Z. Wang, and Z. Su, "What videos are similar with you?: Learning a common attributed representation for video recommendation," in *Proceedings of the 22nd ACM International Conference on Multimedia*. ACM, 2014, pp. 597–606.

[119] X. Ma, H. Wang, H. Li, J. Liu, and H. Jiang, "Exploring sharing patterns for video recommendation on youtube-like social media," *Multimedia Systems*, vol. 20, no. 6, pp. 675–691, 2014.

[120] Z. Ma, F. Nie, Y. Yang, J. R. Uijlings, N. Sebe, and A. G. Hauptmann, "Discriminating joint feature analysis for multimedia data understanding," *IEEE Transactions on Multimedia*, vol. 14, no. 6, pp. 1662–1672, 2012.

[121] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations (ICLR)*, 2014.

[122] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4489–4497.

[123] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[124] B. Kumar, G. Carneiro, I. Reid *et al.*, "Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5385–5394.

[125] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems(NIPS)*, 2017, pp. 5998–6008.

[126] J. Gao, T. Zhang, and C. Xu, "A unified personalized video recommendation via dynamic recurrent neural networks," in *Proceedings of the 2017 ACM on Multimedia Conference*. ACM, 2017, pp. 127–135.

[127] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *EMNLP*, 2014.

[128] Y. 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*. ACM, 2008, pp. 426–434.

[129] G. Karypis, "Evaluation of item-based top-n recommendation algorithms," in *Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM)*. ACM, 2001, pp. 247–254.

[130] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the twenty-fifth Conference on Uncertainty in Artificial Intelligence(AUAI)*. AUAI Press, 2009, pp. 452–461.

[131] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, and T.-Y. Liu, "Sequential click prediction for sponsored search with recurrent neural networks." in *The Association for the Advancement of Artificial Intelligence (AAAI)*, vol. 14, 2014, pp. 1369–1375.

[132] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2008, pp. 650–658.

[133] C.-D. Wang, Z.-H. Deng, J.-H. Lai, and S. Y. Philip, "Serendipitous recommendation in e-commerce using innovator-based collaborative filtering," *IEEE transactions on cybernetics*, no. 99, pp. 1–15, 2018.

[134] Z. Sun, B. Wu, Y. Wu, and Y. Ye, "Apl: Adversarial pairwise learning for recommender systems," *Expert Systems with Applications*, vol. 118, pp. 573–584, 2019.

[135] Y. Tay, L. A. Tuan, and S. C. Hui, "Latent relational metric learning via memory-based a ention for collaborative ranking," in *Proceedings of the 27th International Conference on World Wide Web*, 2018.

[136] S. Qian, T. Zhang, and C. Xu, "Cross-domain collaborative learning via discriminative nonparametric bayesian model," *IEEE Transactions on Multimedia*, vol. 20, no. 8, pp. 2086–2099, 2018.

[137] G. Hu, Y. Zhang, and Q. Yang, "Conet: Collaborative cross networks for cross-domain recommendation," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, 2018, pp. 667–676. [Online]. Available: https://doi.org/10.1145/3269206.3271684

[138] E. Zheng, G. Y. Kondo, S. Zilora, and Q. Yu, "Tag-aware dynamic music recommendation," *Expert Systems with Applications*, vol. 106, pp. 244–251, 2018.

[139] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, and M. Guo, "Multi-task feature learning for knowledge graph enhanced recommendation," *arXiv preprint arXiv:1901.08907*, 2019.

[140] Y. Cao, X. Wang, X. He, T.-S. Chua *et al.*, "Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences," *arXiv preprint arXiv:1902.06236*, 2019.

[141] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, and T. Chua, "Explainable reasoning over knowledge graphs for recommendation," *CoRR*, vol. abs/1811.04540, 2018. [Online]. Available: http://arxiv.org/abs/1811.04540

[142] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma, "Explicit factor models for explainable recommendation based on phrase-level sentiment analysis," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 2014, pp. 83–92.

[143] X. Wang, X. He, L. Nie, and T.-S. Chua, "Item silk road: Recommending items from information domains to social users," in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2017, pp. 185–194.

[144] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*. Springer, 2018, pp. 593–607.

[145] X. He, Z. He, J. Song, Z. Liu, Y.-G. Jiang, and T.-S. Chua, "Nais: Neural attentive item similarity model for recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 12, pp. 2354–2366, 2018.

[146] Y. Peng, Y. Fang, Z. Xie, and G. Zhou, "Topic-enhanced emotional conversation generation with attention mechanism," *Knowledge-Based Systems*, vol. 163, pp. 429–437, 2019.

[147] J. Mun, M. Cho, and B. Han, "Text-guided attention model for image captioning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[148] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, 2015, pp. 2048–2057.

[149] Y. Zhang, Q. Ai, X. Chen, and W. B. Croft, "Joint representation learning for top-n recommendation with heterogeneous information sources," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 1449–1458.

[150] L. Wu, P. Sun, R. Hong, Y. Ge, and M. Wang, "Collaborative neural social recommendation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–13, 2018.

[151] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[152] S. Rendle, "Factorization machines with libfm," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 3, no. 3, p. 57, 2012.