Connecting Latent ReLationships over Heterogeneous Attributed Network for Recommendation

Ziheng Duan[1, 2] **Yueyang Wang**[2*]
Weihao Ye[1] **Zixuan Feng**[2] **Qilin Fan**[1] **Xiuhua Li**[1]

[1]School of Big Data and Software Engineering, Chongqing University, Chongqing 401331, China
[2]Department of Control Science and Technology, Zhejiang University, Hangzhou 310027, China
duanziheng@zju.edu.cn, yueyangw@cqu.edu.cn
yeweihao@cqu.edu.cnn, fengzixuan121@gmail.com, fanqilin@cqu.edu.cn, lixiuhua1099@gmail.com

Abstract

Recently, deep neural network models for graph-structured data have been demonstrating to be influential in recommendation systems. Graph Neural Network (GNN), which can generate high-quality embeddings by capturing graph-structured information, is convenient for the recommendation. However, most existing GNN models mainly focus on the homogeneous graph. They cannot characterize heterogeneous and complex data in the recommendation system. Meanwhile, it is challenging to develop effective methods to mine the heterogeneity and latent correlations in the graph. In this paper, we adopt Heterogeneous Attributed Network (HAN), which involves different node types as well as rich node attributes, to model data in the recommendation system. Furthermore, we propose a novel graph neural network-based model to deal with HAN for Recommendation, called HANRec. In particular, we design a component connecting potential neighbors to explore the influence between neighbors and provide two different strategies with the attention mechanism to aggregate neighbors' information. The experimental results on two real-world datasets prove that HANRec outperforms other state-of-the-arts methods[1].

Introduction

As it becomes more convenient for users to generate data than before, mass data appeared on the Internet are no longer with simple structure but usually with more complex types and structures. This makes the recommendation system (Ricci, Rokach, and Shapira 2011), which helps users discover items of interest, attract widespread attention, and face more challenges (Lekakos and Caravelas 2008). Traditional recommendation methods, such as matrix factorization (Ma et al. 2008), are designed to explore the user-item interactions and generate an efficient recommendation. However, because of calculating each pair of interactions, they have difficulty processing sparse and high-volume data. Furthermore, it is more challenging to deal with heterogeneous and complex data derived from different sources (Wu et al. 2015; Kefalas, Symeonidis, and Manolopoulos 2018).

As the data describing users and items in the recommendation system can be naturally organized as a graph structure. The graph-based deep learning methods (Kipf and Welling 2016) have been applied in recommendation fields (Fan et al. 2019a) and alleviate the enormous amount and the sparsity problem to some extent. Meanwhile, heterogeneous attributed network (HAN) (Wang et al. 2019b), consisting of different types of nodes representing objects, edges denoting the relationships, and various attributes, is a unique form of graph and is powerful to model heterogeneous and complex data in the recommendation system (Zhu et al. 2021). Therefore, designing a graph-based recommendation method to mine the information in HAN is a promising direction (Zhong et al. 2020).

Recently, Graph Neural Network (GNN) (Defferrard, Bresson, and Vandergheynst 2016), a kind of deep learning-based method for processing graphs, has become a widely used graph analysis method due to its high performance and interpretability. GNN based recommendation methods have proved to be successful to some extent because they could simultaneously encode the graph structure and the attributes of nodes (Derr, Ma, and Tang 2018). However, the advances of GNNs are primarily concentrated on homogenous graphs, so they still encounter limitations to utilize rich information in HAN. The major challenge is caused by the heterogeneity of the recommendation graph. The recommendation methods should be able to measure similarities between users and items from various aspects (Fan et al. 2019a). On the one hand, different types of nodes have various attributes that cannot be directly applied due to the different dimensions of node attributes (Wang et al. 2019b). For example, in movie rating graphs, a "user" is associated with attributes like interests and the number of watched movies, while a "movie" has attributes like genres and years. On the other hand, the influence of distinct relationships is different, which should not be treated equally (Chen et al. 2020). For instance, the correlation between "user-Rating-movie" is intuitively stronger than "movie-SameGenres-movie." Hence, how to deal with different feature spaces of various objects' attributes and how to make full use of heterogeneity to distinguish the impact of different types of entities are challenging.

Furthermore, GNNs are based on the connected neighbor aggregation strategy to update the entity's representation

---

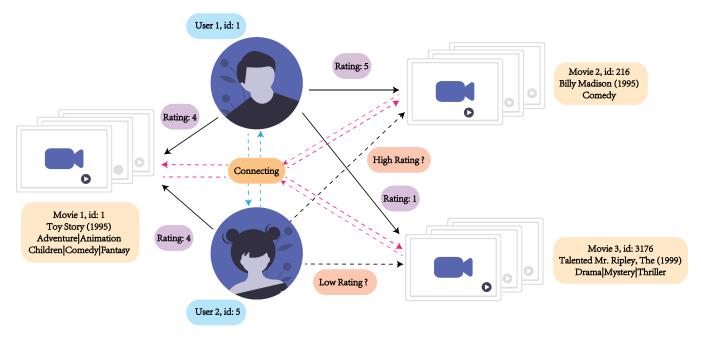[1]Code will be available when this paper is published.

Figure 1: An example of connecting the users and movies from the real-world dataset: MovieLen. The solid black line represents the rating scores of movies by users. We use the blue dashed line to indicate the potential relationships between users and the red dashed line to demonstrate the potential relationships between movies through the connect module. The black dotted line is the rating score of the movie 2 and 3 of user 2, which we want to predict.

(Zhou et al. 2018). However, the reality is that some potential relationships of entities are not directly connected, but implicit (Wu et al. 2020). For example, as shown in Fig. 1, both user 1 and user 2 score the movie 1 as the same rating 4. That in part reflects the interests of the two users are similar. Analogously, movie 1 and movie 2 have the same genres. This infers there was some potential relationship between the two movies. Nevertheless, these implicit relationships are hardly captured by GNNs. Meanwhile, the degree of rating could reflect the preferences of users. For instance, user 1 rates the movie 2 as 5, but the movie 3 as 1. This may infer that user 2 prefers movie 3 to movie 2. Therefore, how to explicitly model entities with potential relationships to provide references for each other's recommendation and distinguish the rating weights are significant.

To better overcome these challenges mentioned above, we propose a neural network model to deal with Heterogeneous Attributed Network for Recommendation, called HANRec, which can make full use of the heterogeneity of graphs and deeper encode the latent relationships. Specifically, we first design a component connecting potential neighbors to explore the influence between neighbors with potential connections. The connecting component also assigns users' rating information to different weights and integrates them into the potential relationships. Next, we design homogeneous aggregation and heterogeneous aggregation strategies to aggregate feature information of both connected neighbors and potential neighbors. It is worth mentioning that we introduce the attention mechanism (Vaswani et al. 2017) to measure the different impacts of heterogeneous nodes. The learned parameters in the two aggregations are different so that they

can model different patterns of the heterogeneous graph structure. Finally, we use the entities' high-quality embeddings to make corresponding recommendations through the recommendation generation component. To summarize, we make the following contributions:

- We propose a new method to connect the users, thus providing a potential reference for recommendations, where existing methods usually miss.

- We present a novel framework for the recommendation, making full use of the heterogeneity in graphs and design strategies for gathering information for entities of different types.

- We design an attention mechanism to characterize the different influences among entities.

- Our model outperforms previous state-of-the-art methods in two recommendation tasks.

Related Work

This section briefly introduces some related works on recommendation algorithms, mainly from traditional methods and deep learning methods.

For the past years, the use of social connections for the recommendation has attracted great attention (Tang, Aggarwal, and Liu 2016; Yang et al. 2017). Traditional methods to deal with recommendation problems mainly include content-based recommendation algorithms and collaborative filtering algorithms. The basic idea of Content-Based Recommendation (CB) (Pazzani and Billsus 2007) is to extract product features from known user preference records

and recommend the product that is most similar to his/her known preference to the user. The recommendation process of CB can be divided into three steps: Item Representation, Profiles Learning, and Generate Recommendation Generation. A further complication in generating a product representation is that the information for the product is unstructured. The most typical unstructured product information is text. The weight of words can be calculated by using Term Frequency (TF) (Ramos 2003) and Inverse Document Frequency (IDF) (Ramos 2003) to vectorize text. Feature learning is the process of modeling user interests. The training process of all user interest models is a supervised classification problem that can be solved using traditional machine learning algorithms. For example, decision tree algorithm (Safavian and Landgrebe 1991), k-nearest neighbor algorithm (Keller, Gray, and Givens 1985), Naive Bayes algorithm (Vazquez et al. 2003), support vector machine (Baesens et al. 2000), etc. CB uses different ways to generate recommendations depending on the recommendation task scenario. This strategy works well in situations where the product's attributes can be easily identified. Nevertheless, it is easy to over-specialize the user's interest, and it is challenging to find out the user's potential interest. The idea of Collaborative Filtering (CF) (Herlocker et al. 2004) is to find some similarity through the behaviors of groups and make recommendations for users based on this similarity. The recommendation process is: establishing a user behavior matrix, calculating similarity, and generating recommendations. Collaborative filtering based on matrix decomposition decomposes the user rating matrix into user matrix U and product matrix V. It then recombines U and V to get a new user rating matrix to predict unknown ratings. Matrix decomposition includes Funk-SVD model (Koren, Bell, and Volinsky 2009), PMF model (Mnih and Salakhutdinov 2007), etc.

Deep learning, as an emerging method, also has many achievements of learning on graph structure data (Bronstein et al. 2017). The purpose of graph representation learning is to find a mapping function that can transform each node in the graph into a low-dimensional potential factor. The low-dimensional latent factors are more efficient in calculations and can also filter some noise. Based on the nodes' latent factors in the graph, machine learning algorithms can complete downstream tasks more efficiently, such as recommendation tasks and link prediction tasks. Graphs can easily represent data on the Internet, making graph representation learning more and more popular. Graph representation learning includes random walk-based methods and graph neural network-based methods. The random walk-based methods sample the paths in the graph, and the structure information near the nodes can be obtained through these random paths. For example, the DeepWalk proposed by Perozzi et al. (Perozzi, Al-Rfou, and Skiena 2014a) applies the ideas of Natural Language Processing (NLP) to network embedding. DeepWalk treats the relationship between the user and the product as a graph, generating a series of random walks. A continuous space of lower latitudes represents the user vector and the product vector. In this graph representation space, traditional machine learning methods, such as Logistic Regression (LR), can predict users' ratings of products to obtain more accurate results. Collaborative Deep Learning (CDL) proposed by Wang et al. (Wang, Wang, and Yeung 2015) jointly deals with the representation of product content information and users' rating matrix for products. CDL relies on user reviews of the product and information about the product itself.

Some researchers have also used graph neural networks (GNNs) to complete recommendation tasks in recent years. Graph Convolutional Network (Kipf and Welling 2016) (GCN) is one of the representative methods. GCN uses convolution operators on the graph to iteratively aggregate the neighbor embeddings of nodes. It utilizes the Laplacian matrix of the graph to implement the convolution operation on the topological graph. In the multi-layer GCNs, each convolutional layer processes the graph's first-order neighborhood information. Superimposing multiple convolutional layers can realize information transfer on the multi-level neighborhood. On this basis, some GNN-based frameworks for recommendation tasks have been proposed. According to whether to consider the order of items, recommendation systems can be divided into regular recommendation tasks, and sequential recommendation tasks (Adomavicius and Tuzhilin 2005; Kang and McAuley 2018). The general recommendation considers users to have static interest preferences and models the degree of matching between users and items based on implicit or explicit feedback. GNN can capture user-item interactions and learn user and item representations. The sequential recommendation captures the serialization mode in the item sequence and recommends the next item of interest to the user. There are mainly methods based on Markov chain (MC) (Rendle, Freudenthaler, and Schmidt-Thieme 2010; He and McAuley 2016), based on RNN (Hidasi and Karatzoglou 2018; Tan, Xu, and Liu 2016), and based on attention and self-attention mechanisms (Liu et al. 2018; Li et al. 2017). With the advent of GNN, some works convert the item sequence into a graph structure and use GNN to capture the transfer mode. Since this paper focuses on discussing the former, we mainly introduce the work of the general recommendation.

General recommendation uses user-item interaction to model user preferences. For example, GC-MC (Berg, Kipf, and Welling 2017) deals with rating score prediction, and the interactive data is represented as a bipartite graph with labeled edges. GC-MC only uses interactive items to model user nodes and ignores the user's representation. So the limitations of GC-MC are: 1) it uses mean-pooling to aggregate neighbor nodes, assuming that different neighbors are equally important; 2) it only considers first-order neighbors and cannot make full use of the graph structure to spread information. Online social networks also develop rapidly in recent years. Recommendation algorithms that use neighbors' preferences to portray users' profiles have been proposed, which can better solve the problem of data sparsity and generate high-quality embeddings for users. These methods use different strategies for influence modeling or preference integration. For instance, DiffNet (Wu et al. 2019) models user preferences based on users' social relationships and historical behaviors, and it uses the GraphSAGE frame-

work to model the social diffusion process. DiffNet uses mean-pooling to aggregate friend representations and mean-pooling to aggregate historical item representations to obtain the user's representation in the item space. DiffNet can use GNN to capture a more in-depth social diffusion process. However, this model's limitations include: 1) The assumption of the same influence is not suitable for the real scene; 2) The model ignores the representation of the items and can be enhanced by interactive users. GraphRec proposed by Fan et al. (Fan et al. 2019b) learns the low-dimensional representation of users and products through graph neural networks. GraphRec used the mean square error of predicted ratings to guide the neural network's parameter optimization and introduced users' social relationships and attention mechanisms to describe user preferences better. With the emergence of heterogeneous and complex data in the recommendation network, some work has introduced knowledge graphs (KG) into recommendation algorithms. The challenge of applying the KG to recommendation algorithms comes from the complex graph structure, multiple types of entities, and relationships in KG. Previous work used KG embedding to learn the representation of entities and relationships; or designed meta-path to aggregate neighbor information. Recent work uses GNN to capture item-item relationships. For example, KGCN (Wang et al. 2019a) uses user-specific relation-aware GNN to aggregate neighbors' entity information and uses knowledge graphs to obtain semantically aware item representations. Different users may impose different importance on different relationships. Therefore, the model weights neighbors according to the relationship and the user, which can characterize the semantic information in the KG and the user's interest in a specific relationship. The item's overall representation is distinguished for different users, and semantic information in KG is introduced. Finally, predictions are made based on user preferences and item representation. IntentGC (Zhao et al. 2019) reconstructs the user-to-user relationship and the item-to-item relationship based on the multi-entity knowledge graph, greatly simplifying the graph structure. The multi-relationship graph is transformed into two homogeneous graphs, and the user and item embeddings are learned from the two graphs, respectively. This work also proposes a more efficient vector-wise convolution operation instead of the splicing operation. IntentGC uses local filters to avoid learning meaningless feature interactions (such as the age of one's node and neighbor nodes' rating).

Although the previous work has achieved remarkable success, the exploration of potential relationships in social recommendations has not been paid enough attention. In this paper, we propose a graph neural network framework that can connect potential relationships in the network to fill this gap.

Proposed Framework

In this section, we present the proposed HANRec in detail. First, we define the heterogeneous attributed network, the formula expression of the recommendation and link prediction problem, and the symbols used in this paper. Later, we give an overview of the proposed framework and the details of each component. Finally, we introduce the optimiza-

tion objectives of HANRec. The schematic of connecting the users and the model structure is shown in Fig. 2.

Problem Formulation and Symbols Definition

Definition 1 *Heterogeneous Attributed Network*

A heterogeneous attributed network can be represented as $G = (V, E, A)$. V is a set of nodes representing different types of objects, E is a set of edges representing relationships between two objects, and A denotes the attributes of objects. The heterogeneity of the graph is reflected as: $type(nodes) + type(edges) > 2$. On the other hand, when $type(nodes) = type(edges) = 1$, we call it a homogeneous graph (Cen et al. 2019). Attributed graphs mean that each node in graphs has corresponding attributes. Entities with attributes are widespread in the real world. For example, in a movie recommendation network, each movie has its genres and users have their preferences; in an author-paper network, each author and paper has related research topics. Therefore, most networks in the real world exist as heterogeneous attribute graphs, which is the focus of this paper.

Definition 2 *Recommendation*

In this paper, the task of recommendation is focused. For a graph $G = (V, E)$, v_i and v_j are two entities in this graph. In the triple (r, v_i, v_j), r represents the relationship between v_i and v_j. A recommendation model learns a score function, and outputs the relationship r between v_i and v_j: $r_{i,j} = Recommendation\_Model(v_i, v_j)$. For example, in the user-movie-rating recommendation network, the main focus is to recommend movies that users might be interested in. This requires the recommendation model to accurately give users the possible ratings of these movies (commonly a 5-point rating, etc.). So in most cases, this is a regression problem.

Definition 3 *Link Prediction*

We also discussed link prediction in this paper. For a graph $G = (V, E)$, v_i and v_j are two entities in this graph. In the triple (r, v_i, v_j), r represents the relationship between v_i and v_j. A link prediction model learns a score function, and outputs the relationship r between v_i and v_j: $r_{i,j} = Link\_Prediction\_Model(v_i, v_j)$. In most cases, the value of r is 0 or 1. This means that there is no edge or there is an edge between the two entities, which is a binary classification problem.

The purpose of link prediction is to infer missing links or predict future relations based on the currently observed part of the network. This is a fundamental problem with a large number of practical applications in network science. The recommendation problem can also be regarded as a kind of link prediction problem. The difference is that recommendation is a regression task that needs to predict specific relationships (such as the ratings of movies by people in a movie recommendation network). At the same time, link prediction is a binary classification task, where we need to determine whether there is a link connected between the two entities.

The mathematical notations used in this paper are summarized in Table 1.

Table 1: Symbols' Definition and Description

Symbols	Definitions
$B(v_i)$	The set of entities of the same type as entity v_i.
$C(v_i)$	The set of entities of different types as entity v_i.
$N(v_i)$	The set of neighbor entities of entity v_i.
$N'(v_i)$	The set of entities that have potential connections with entity v_i.
$r_{i,j}$	The relationship between entity v_i and v_j.
p_i	The embedding vector of entity v_i.
$e_{i,j}$	The rating embedding for the rating level (entity v_i to v_j).
d	The embedding dimension.
h_i^0	The initial representation vector of entity v_i.
h_i^B	The influence embedding among entity v_i and entities of the same type.
h_i^C	The influence embedding among entity v_i and entities of different types.
$f_{i,j}$	The opinion-aware interaction representation between entity v_i and v_j.
$\alpha_{i,j}^*$	Attention parameter between entity of the same type (v_j to v_i).
$\beta_{i,j}^*$	Attention parameter between entity of different types (v_j to v_i).
$\alpha_{i,j}$	The normalized attention parameter between entity of the same type (v_j to v_i).
$\beta_{i,j}$	The normalized attention parameter between entity of different types (v_j to v_i).
$\boldsymbol{W}, \boldsymbol{b}$	The weight and bias in neural networks.

An Overview of HANRec

HANRec consists of four components: Connecting Potential Neighbors, Homogeneous Network Aggregation, Heterogeneous Network Aggregation, and Recommendation Generation. Connecting Potential Neighbors aims to fully explore the influence between neighbors with potential connections. Like the common user-movie-rating network, there is a lot of data for users' ratings of movies, but there is little or no interaction between users. However, the interaction between users is important, and there may be potential influences between users. Users with the same interests will have a high probability of giving similar ratings to the same movie. This component uses shared entities to generate a connection path, which utilizes scores and features to model the potential influence and generate high-quality entity representations.

Homogeneous Network Aggregation is responsible for aggregating information of entities of the same type and quantifying the influence of different neighbors on the entity through an attention mechanism. This component is mainly to portray the influence between entities of the same type. Heterogeneous Network Aggregation is responsible for aggregating information of entities of different types. We also use the user-movie-rating network to illustrate. Different movie genres and the user's ratings will describe the user's preferences from a particular perspective. Heterogeneous Aggregation Network will focus on describing an entity's

characteristics from the perspective of entities of different types. Recommendation generation gathers all the information related to the entity and generates its high-quality embedding. This component also matches the embeddings of other entities to make the final recommendations.

As shown in Fig. 2, in the movie recommendation network, we connect different users/movies through the designed connection method (as indicated by the orange arrow). Then we use homogeneous and heterogeneous aggregation to combine users/movies its own characteristics and generate the final representation embedding. Finally, the user's embedding is compared with the movie's embedding to be evaluated, and the final rating score is generated. Next, we introduce the details of each component.

Connecting Potential Neighbors

We design this component to fully explore the influence between neighbors with potential connections. Note that in the common user-movie-rating network, there may not be a direct link among users and users, movies and movies. At this time, we can connect the users and movies through second-order neighbors. Suppose that $v_j \in B(v_i) \cap \overline{N(v_i)}$, $v_k \in C(v_i) \cap N(v_i)$, $v_k \in C(v_j) \cap N(v_j)$, where $B(v_i)$ is the set of entities that are the same type of v_i; $C(v_i)$ is the set of entities that are different types of v_i and $N(v_i)$ represents the set of neighbors of v_i. We can infer the influence
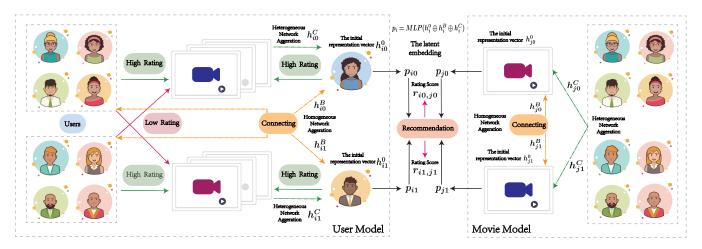
Figure 2: The architecture of HANRec. It contains four major components: connecting potential neighbors, homogeneous aggregation, heterogeneous aggregation and rating generation.

of v_j on v_i from the common neighbor v_k:

$$f_{i,j} = MLP(h_k^0 \oplus e_{i,k} \oplus h_j^0 \oplus e_{k,j}), \quad (1)$$

where $f_{i.j}$ is the opinion-aware interaction representation between entity v_i and v_j; MLP means Multilayer Perceptron (Pal and Mitra 1992); $\oplus$ represents the concatenate operator of two vectors; h_k^0 and h_j^0 represent the initial features of v_k and v_j, respectively; $e_{i,k}$ and $e_{k,j}$ represent the rating embedding for the rating level between v_i, v_k and v_k, v_j, respectively. Through this connecting way, we use $f_{i,j}$ to represent their previous potential relationship between v_i and v_j. Then v_j is added to $N'(v_i)$, which represents the set of entities that have potential connections with entity v_i. The connection method we designed can fully integrate the characteristics of entities and the relationships between entities in the path. For example, there may not be a direct connection between users in a common movie recommendation network. In this way, the two users can be connected by using the movies they have watched together. The movie genres and the user's rating of the movie can provide an essential reference for users' potential relationships.

Homogeneous Network Aggregation

Homogeneous Network Aggregation aims to learn the influence embedding h_i^B, representing the relationship among entity v_i and entities of the same type. Given the initial representation of entity v_j, and the rating embedding $e_{i,j}$, the opinion-ware interaction representation $f_{i,j}$ can be expressed as:

$$f_{i,j} = MLP(h_j^0 \oplus e_{i,j}). \quad (2)$$

The attention parameter $\alpha_{i,j}^*$ reflecting the influence of different entities v_i and v_j is designed as:

$$\alpha_{i,j}^* = \boldsymbol{w_2^T} \cdot \sigma(\boldsymbol{W_1} \cdot [h_i^0 \oplus h_j^0] + \boldsymbol{b_1}) + \boldsymbol{b_2}, \quad (3)$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ represents the weight and bias in neural networks. The normalized attention parameter $\alpha_{i,j}$ is as follows:

$$\alpha_{i,j} = \frac{exp(\alpha_{i,j}^*)}{\sum_{k \in B(v_i) \cap (N(v_i) \cup N'(v_i))} exp(\alpha_{i,k}^*)}. \quad (4)$$

The design of attention parameters is based on the assumption that entities with similar characteristics should have greater influence among them. And we use a neural network to learn this influence adaptively. In this way, we get the information gathered by entity v_i from its homogeneous graph:

$$h_i^B = \sum_{j \in B(v_i) \cap (N(v_i) \cup N'(v_i))} (\alpha_{i,j} * f_{i,j}) \quad (5)$$

It is worth mentioning that when we aggregate neighbor information, we not only consider neighbors with edge connections but also consider neighbors with potential relationships discovered through our connection method. We aggregate their influence on entity v_i in the meantime. Subsequent experiments proved the superiority of our design ideas.

Heterogeneous Network Aggregation

Heterogeneous Network Aggregation aims to learn the influence embedding h_i^C, representing the relationship among entity v_i and entities of different types. Given the initial representation of entity v_j, and the rating embedding $r_{i,j}$, the opinion-ware interaction representation $f_{i,j}$ can be expressed as:

$$f_{i,j} = MLP(h_j^0 \oplus e_{i,j}) \quad (6)$$

The attention parameter $\beta_{i,j}^*$ reflecting the influence of different entities v_i and v_j is designed as:

$$\beta_{i,j}^* = \boldsymbol{w_4^T} \cdot \sigma(\boldsymbol{W_3} \cdot [h_i^0 \oplus h_j^0] + \boldsymbol{b_3}) + \boldsymbol{b_4} \quad (7)$$

The normalized attention parameter $\beta_{i,j}$ is as follows:

$$\beta_{i,j} = \frac{exp(\beta_{i,j}^*)}{\sum_{k \in C(v_i) \cap (N(v_i) \cup N'(v_i))} exp(\beta_{i,k}^*)} \quad (8)$$

In this way, we get the information gathered by entity v_i from its heterogeneous graph:

$$h_i^C = \sum_{j \in C(v_i) \cap (N(v_i) \cup N'(v_i))} (\beta_{i,j} * f_{i,j}) \quad (9)$$

According to the different types of neighbors, we designed two aggregation strategies to highlight the impact of different types of entities on v_i. For example, in a common movie recommendation network, the influence of movies and other people on a person should be different, and mixing them cannot make full use of the graph's heterogeneity. The subsequent experimental part also proved our conjecture.

Recommendation Generation

After gathering information from entities of the same and different types, we can easily get the latent representation of entity v_i:

$$p_i = MLP(h_i^0 \oplus h_i^B \oplus h_i^C), \qquad (10)$$

where h_i^0 is the initial representation of entity v_i, such as the user's preferences in the user-movie-rating network, the movie genres, etc. h_i^B and h_i^C represent the influence embeddings indicating the relationships among entity v_i and entities of the same type and different types, respectively. For v_i and v_j, after getting the embedding that aggregates a variety of information (p_i and p_j), their relationship can be measured as:

$$r_{i,j} = MLP(p_i \oplus p_j). \qquad (11)$$

So far, the entire end-to-end recommendation prediction process has been completed.

Objective Function

To optimize the parameters involved in the model, we need to specify an objective function to optimize. Since the task we focus on in this work is rating prediction and link prediction, the following loss function is used referred to (Fan et al. 2019a):

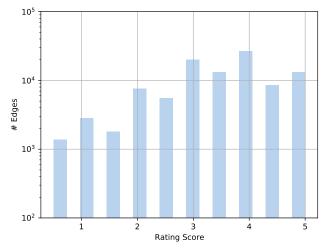$$Loss = \frac{1}{2|T|} \sum_{i,j \in T} (r'_{i,j} - r_{i,j})^2, \qquad (12)$$

where $|T|$ is the number used in the training dataset, $r'_{i,j}$ is the relationship between entity v_i and v_j predicted by the model and $r_{i,j}$ is the ground truth.

To optimize the objective function, we use Adam (Kingma and Ba 2014) as the optimizer in our implementation. Each time it randomly selects a training instance and updates each model parameter in its negative gradient direction. In optimizing deep neural network models, overfitting is an eternal problem. To alleviate this problem, we adopted a dropout strategy (Srivastava et al. 2014) in the model. The idea of dropout is to discard some neurons during training randomly. When updating parameters, only part of them will be updated. Besides, since the dropout function was disabled during the test, the entire network will be used for prediction. The whole algorithm framework is shown in algorithm 1.

Experiments

Datasets

In order to verify the effectiveness of HANRec, we performed two tasks: recommendation and link prediction.
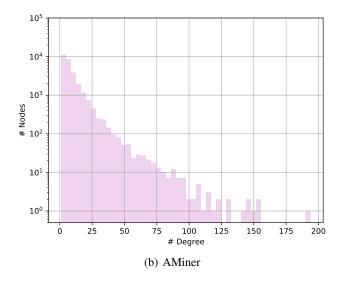

(a) MovieLen

(b) AMiner

Figure 3: Characteristics of the MovieLen and the AMiner dataset.

For the recommendation task, MovieLens 100k[2] was used, a common data set in the field of movie recommendation. It consists of 10,334 nodes, including 610 users and 9724 movies, and 100,836 edges. It is worth mentioning that the edges here all indicate the user's rating of the movie, and there is no edge connection between users or between movies. MovieLens also includes the movie's genre attributes and the timestamp of the rating. Movie categories include 18 categories such as Action, Drama, Fantasy. Each movie can have multiple unique category attributes. For example, Batman (1989) has three category attributes: Action, Crime, and Thriller. We initialize representation embeddings for these 18 genres. The initial embedding of each movie is the average of the genre's embeddings in the movie. For the MovieLen dataset, our model focuses on three kinds of em-

[2]https://grouplens.org/datasets/movielens/100k/

Algorithm 1: HANRec algorithm framework

Input : The heterogeneous attributed network $G = (V, E, A)$; two entities v_1 and v_2 whose relationship needs to be evaluated

Output: The relationship $r_{1,2}$ between the two entities v_1 and v_2

---

1 $P = \{\}$;

    // P is used to record the representation embedding of each entity v

2 for v_i *in* $\{v_1, v_2\}$ **do**

3 $N'(v_i) = \{\}$;

      // $N'(v_i)$ records the entities that have potential connections with entity v_i

4   **for** v_m *in* $N(v_i)$ **do**

5 $f_{i,m} = MLP(h_m^0 \oplus e_{i,m})$;

         // The opinion-aware interaction representation between entity v_i and v_m

6   **for** v_n *in* $N(v_m)$ **do**

7   **if** $v_n \neq v_i$ **then**

8 $N'(v_i) \leftarrow v_n$;

               // Add v_n to $N'(v_i)$

9 $f_{i,n} = MLP(h_m^0 \oplus e_{i,m} \oplus h_n^0 \oplus e_{m,n})$;

               // The opinion-aware interaction representation between entity v_i and v_n

10   **end**

11   **end**

12   **end**

    // Connecting Potential Neighbors

13 $\alpha_{i,j}^* = \boldsymbol{w_2^T} \cdot \sigma(\boldsymbol{W_1} \cdot [h_i^0 \oplus h_j^0] + \boldsymbol{b_1}) + \boldsymbol{b_2}, \alpha_{i,j} = \frac{exp(\alpha_{i,j}^*)}{\sum_{k \in B(v_i) \cap (N(v_i) \cup N'(v_i))} exp(\alpha_{i,k}^*)}$;

    // The attention parameters in homogeneous aggregation

14 $h_i^B = \sum_{v_k \in B(v_i) \cap (N(v_i) \cup N'(v_i))} (\alpha_{i,k} * f_{i,k})$;

    // Homogeneous Aggregation

15 $\beta_{i,j}^* = \boldsymbol{w_4^T} \cdot \sigma(\boldsymbol{W_3} \cdot [h_i^0 \oplus h_j^0] + \boldsymbol{b_3}) + \boldsymbol{b_4}, \beta_{i,j} = \frac{exp(\beta_{i,j}^*)}{\sum_{k \in C(v_i) \cap (N(v_i) \cup N'(v_i))} exp(\beta_{i,k}^*)}$;

    // The attention parameters in heterogeneous aggregation

16 $h_i^C = \sum_{v_k \in C(v_i) \cap (N(v_i) \cup N'(v_i))} (\beta_{i,k} * f_{i,k})$;

    // Heterogeneous Aggregation

17 $p_i = MLP(h_i^0 \oplus h_i^B \oplus h_i^C)$;

    // Fuse v_i's initial feature and the information aggregated from homogeneous and heterogeneous neighbors to obtain the final representation embedding p_i

18 $P \leftarrow p_i$;

    // Add p to P

19 end

20 $r_{1,2} = MLP(p_1 \oplus p_2)$;

    // Compute the relationship $r_{1,2}$ between two entity v_1 and v_2

21 return $r_{1,2}$

beddings, including user embedding h_i^0, where i belongs to the user index, movie embedding h_j^0, which is composed of the embedding of movie genres, and j belongs to the movie index, and opinion embedding $e_{i,j}$. They are initialized randomly and learn together during the training phase. Since the original features are extensive and sparse, we do not use one-key vectors to represent each user and item. By embedding high-dimensional sparse features into the low-dimensional latent space, the model can be easily trained (He et al. 2017). The opinion embedding matrix $e_{i,j}$ depends on the system's rating range. For example, for the MovieLen dataset, which is a 5-star rating system and 0.5 as an interval, the opinion embedding matrix e contains nine different embedding vectors to represent $\{0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5\}$ in the score. Fig. 3(a) shows the rating score distribution of edges in the MovieLen dataset. We can clearly see fewer edges with lower rating scores, and there are more edges with rating scores from 3 to 4.

For the link prediction task, we use the AMiner dataset[3] (Tang et al. 2008). We construct the relationship between an author and one paper when the author published this paper. We also generate the relationships between authors when they are co-authors and generate the relationships between papers when there are citation relationships. The authors' initial attribute contains research interests, published papers, and the total number of citations. (Stallings et al. 2013). The title and abstract can represent the initial attribute of papers. In this author-paper-network, we treat the weight of edges as binary. In the experiment part, we selected all papers from the famous venue[4] in eight research topics (Dong, Chawla, and Swami 2017) and all the relative authors who published these papers. Based on this, we derive a heterogeneous attributed network from the academic network of the AMiner dataset. It consists of 29,059 nodes, including 16,604 authors and 12,455 papers with eight labels, and 124,626 edges representing 62,115 coauthor, 31,251 citation, and 31,263 author-paper relationships. We treat authors' and papers' text descriptions as node attributes and transform them into vectors by Doc2vec in the experiments. Fig. 3(b) shows the node distribution of the AMiner dataset. We can find that most nodes have few neighbors, but there are still some super nodes whose number of neighbors exceeds 100. In this case, it is more important to distinguish the influence of different neighbors effectively.

Metrics

We apply two conventional evaluation metrics to evaluate the performance of different models for recommendation: Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE):

$$MAE = \frac{1}{m} \sum_{i=1}^{m} (p_i - a_i) \qquad (13)$$

$$RMSE = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (p_i - a_i)^2} \qquad (14)$$

a = actual target, p = predict target

Smaller values of MAE and RMSE indicate better predictive accuracy. For the link prediction task, Accuracy and AUC (Area Under the Curve) are used to quantify the accuracy. Higher values of Accuracy and AUC indicate better predictive accuracy.

Baselines

The methods in our comparative evaluation are as follows:

- **Doc2Vec** (Le and Mikolov 2014) is the Paragraph Vectors algorithm that embeds the text describing objects in a distributed vector using neural network models. Here we use Doc2Vec to process the text describing authors and papers' research interests in the link prediction task to obtain the initialization embedding of authors and papers.

- **DeepWalk** (Perozzi, Al-Rfou, and Skiena 2014b) uses random walk to sample nodes in the graph to get the node embedding. As for the relevant parameters, we refer to the original paper. We set num-walks as 80, walk-length as 40, and window-size as 10.

- **LINE** (Tang et al. 2015) minimizes a loss function to learn embedding while preserving the first and the second-order neighbors' proximity among nodes. We use LINE (1st+2nd) as overall embeddings. The number of negative samples is set as 5, just the same as the original paper.

- **Node2Vec** (Grover and Leskovec 2016) adopts a biased random walk strategy and applies Skip-Gram to learn node embedding. We set p is 0.25 and q as 4 here.

- **SoRec** (Ma et al. 2008) performs co-factorization on the user-item rating matrix and user-user social relations matrix. We set the parameters as the same as the original paper. $\lambda_C = 10$, and $\lambda_U = \lambda_V = \lambda_Z = 0.001$.

- **GATNE** (Cen et al. 2019) provides an embedding method for large heterogeneous network. In the two datasets used here, the edge type is 1, so the edge embedding type in GATNE is set to 1.

- **GraphRec** (Fan et al. 2019c) jointly captures interactions and opinions in the user-item graph. For the MovieLen dataset here, there are no user-to-user and movie-to-movie interactions, so only item aggregation and user aggregation of the original paper are used.

- **HANRec** is our proposed framework, which makes full use of the heterogeneity and attribute of the network and uses the attention mechanism to provide better recommendations.

Training Details

For the task of recommendation, we use the MovieLen dataset. The recommendation task's goal is to predict the

Table 2: Results comparison of the recommendation. MAE/RMSE are evaluation metrics.

Train Edges	DeepWalk	LINE	Node2Vec	SoRec	GATNE	GraphRec	HANRec
30%	0.8514/1.0578	0.8614/1.0687	0.8519/1.0317	0.8278/1.0324	0.7923/0.9848	0.7824/0.9748	**0.7751/0.9624**
40%	0.8258/1.0014	0.8413/1.0342	0.8202/0.9832	0.7891/0.9936	0.7774/0.9607	0.7528/0.9555	**0.7419/0.9355**
50%	0.7869/0.9817	0.8017/0.9959	0.7928/0.9707	0.7417/0.9615	0.7319/0.9487	0.7347/0.9379	**0.7128/0.9132**
60%	0.7521/0.9678	0.7758/0.9816	0.7498/0.9601	0.7314/0.9504	0.7217/0.9215	0.7191/0.9189	**0.6981/0.9047**
70%	0.7331/0.9497	0.7525/0.9607	0.7345/0.9427	0.7257/0.9407	0.7047/0.9158	0.7055/0.9107	**0.6823/0.8925**
80%	0.7250/0.9418	0.7332/0.9511	0.7273/0.9395	0.7192/0.9339	0.6954/0.9057	0.6966/0.9071	**0.6753/0.8855**
90%	0.7148/0.9375	0.7241/0.9403	0.7068/0.9189	0.6957/0.9048	0.6824/0.8907	0.6807/0.8827	**0.6673/0.8681**

Table 3: Results comparison of the link prediction. AUC/Accuracy are evaluation metrics.

Train Edges	Doc2Vec	DeepWalk	LINE	Node2Vec	SoRec	GraphRec	HANRec
30%	0.8074/0.7335	0.8693/0.8206	0.7911/0.7214	0.8808/0.8341	0.9228/0.8547	0.9241/0.8613	**0.9612/0.8982**
40%	0.8158/0.7407	0.8873/0.8345	0.8017/0.7275	0.8879/0.8410	0.9317/0.8602	0.9333/0.8752	**0.9647/0.9032**
50%	0.8243/0.7505	0.8948/0.8415	0.8155/0.7357	0.9023/0.8531	0.9394/0.8739	0.9517/0.8907	**0.9679/0.9101**
60%	0.8395/0.7612	0.9105/0.8631	0.8209/0.7398	0.9201/0.8617	0.9534/0.8907	0.9571/0.8988	**0.9758/0.9223**
70%	0.8548/0.7765	0.9358/0.8758	0.8288/0.7451	0.9393/0.8826	0.9627/0.9015	0.9673/0.9056	**0.9807/0.9409**
80%	0.8702/0.7893	0.9501/0.8921	0.8371/0.7473	0.9513/0.8901	0.9708/0.9077	0.9728/0.9099	**0.9881/0.9514**
90%	0.8771/0.7973	0.9573/0.8967	0.8429/0.7506	0.9584/0.8972	0.9782/0.9103	0.9792/0.9156	**0.9912/0.9611**

rating score of one user to one movie. $x\%$ of the scoring records are used for training, and the remaining $(100\text{-}x)\%$ are used for testing. In this task, HANRec will output a value between 0.5 and 5.0, indicating the user's possible rating for the movie. The closer the score is to the ground truth, the better the performance of the model. For the task of link prediction, we use the AMiner dataset. The link prediction task's goal is to predict whether there is an edge between two given nodes. First, we randomly hide $(100\text{-}x)\%$ of the edges in the original graph to form positive samples in the test set. The test set also has an equal number of randomly selected disconnected links that servers as negative samples. We then use the remaining $x\%$ connected links and randomly selected disconnected ones to form the training set. HANRec outputs a value from 0 to 1, indicating the probability that there are edges between entities. For these two tasks, the value of x ranges from 30 to 90, and the interval is 10. It is worth noting that when x is low, we call it a cold start problem (Schafer et al. 2007), which is discussed in detail in the section 21.
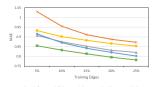
The proposed HANRec is implemented on the basis of Pytorch 1.4.0[5]. All multilayer perceptrons have a three-layer linear network and prelu activation function by default. The embedding size d used in the model and the batch size are all set to 128. The learning rate are searched in [0.001, 0.0001, 0.00001], and the Adam algorithm (Kingma and Ba 2014) is used to optimize the parameters of HANRec. The performance results were recorded on the test set after 2000 iterations on the MovieLen dataset and 20000 iterations on the AMiner dataset. The parameters for the baseline algorithms were initialized as in the corresponding papers and were then carefully tuned to achieve optimal performance.
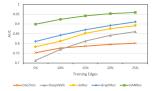
Main Results

We can first see that no matter which task it is, as the proportion of the training data set increases, the effect of

HANRec on the test set gets better and better. Then we focus on the recommendation performance of all methods. Table 2 shows the overall rating prediction results (MAE and RMSE) among different recommendation methods on MovieLen datasets. We find that no matter how much x is, the performance of DeepWalk, LINE, and Node2Vec, which are based on the walking in the graph, is relatively low. As a matrix factorization method that leverages both the rating and social network information, SoRec is slightly better than the previous three. GATNE's embedding of edge heterogeneity and GraphRec's social aggregation cannot be fully utilized in this case. Nevertheless, these two methods' performance is generally better than the previous methods, which proves the effectiveness of deep neural networks on the recommendation task. The model we proposed, HANRec, can fully connect users, thus providing more guidance for the recommendation, thus showing the best performance on the recommendation task.

Then we evaluate HANRec's performance on the link prediction accuracy and compare HANRec with other link prediction algorithms mentioned above. Table 3 demonstrates that the performance of Doc2Vec, which does not use graph information and only uses the initial information of each node, is the worst. This shows that the graph structure is essential in the link prediction task, and the initial features of the nodes are challenging to provide enough reference for the link prediction task. Methods that use graph structure information, such as DeepWalk, LINE, Node2Vec, and SoRec, have higher performance than Doc2Vec, indicating that in this case, the graph structure information is more important than the initial information of nodes. The effect of GATNE, which uses graph heterogeneity, is better than the previously mentioned methods, which shows that graph heterogeneity can also provide some reference for the link prediction task. The performance of our method HANRec in the link prediction task far exceeds other methods, which further shows that the framework can fully consider the ho-

(a) Cold-Start: MovieLen-MAE   (b) Cold-Start: AMiner-AUC

Figure 4: Cold Start Problem analysis on the MovieLen and AMiner datasets.

mogeneity and heterogeneity of the graph and use the attention mechanism to generate high-quality embeddings for nodes. Further investigations are also conducted to better understand each component's contributions in HANRec in the following subsection.
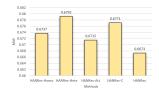
In conclusion, we can know from these results: (1) graph neural networks can boost the performance in the recommendation and the link prediction tasks; (2) using the heterogeneity of graphs can make full use of the information of the network, thereby improving the performance on these two tasks; (3) our proposed HANRec achieves the state-of-art performance in the MovieLen and the AMiner datasets for both tasks.
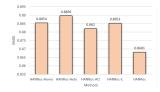
Cold Start Problem

The cold start problem (Schafer et al. 2007) refers to the difficulty of the recommendation system to give high quality recommendations when there is insufficient data. According to (Bobadilla et al. 2012), the cold start problem can be divided into three categories: (1) User cold start: how to make personalized recommendations for new users; (2) Item cold start: how to recommend a new item to users who may be interested in it; (3) System cold start: how to design a personalized recommendation system on a newly developed website (no users, user behavior, only partial item information), so that users can experience personalized recommendations when the website is released.
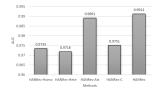
In this paper, the cold start problem belongs to the third category, which can be defined as how to make high-quality recommendations based on the information of a small number of edges (rating scores or relationships). We continuously adjusted the proportion of available information in the two datasets, that is, the proportion of training set x, from 5% to 25%, and obtained the performance results of a series of methods. When the ratio of the training edges is from 5% to 25%, Fig. 4(a) and 4(b) show the performance of different models in the MoveLen and the AMiner datasets. Since the methods based on the random walk, such as Node2Vec and LINE, have a similar performance with DeepWalk, we only report the results of DeepWalk here.
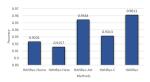
Both Fig. 4(a) and Fig. 4(b) reflect that HANRec performs the best in various cases. Specifically, when the proportion of training edges is 5%, HANRec outperforms other methods the most. This reflects the superiority of the connect method we designed: when there are very few known edges,

(a) Ablation Study: MovieLen-MAE   (b) Ablation Study: MovieLen-RMSE

(c) Ablation Study: AMiner-AUC   (d) Ablation Study: AMiner-Accuracy

Figure 5: Effect of each component on the MovieLen and the AMiner dataset.

the entity's neighbor information is incomplete. The embedding obtained by other methods by aggregating neighbor information does not fully integrate this neighbor information. By connecting potential neighbors, HANRec overcomes the disadvantage of few known edges to a certain extent and is suitable for alleviating the difficulty of generating high-quality recommendations during cold start. It is worth noticing that as shown in Fig. 4(b), Doc2Vec, the recommended method based on entity features, is better than DeepWalk as the first, which is based on the random walk. As the proportion of the training edges increases, DeepWalk gradually overtakes Doc2Vec. This is intuitive: because DocVec does not use the graph's structural information, it only makes recommendations based on each entity's feature. The increase in the number of training edges has relatively little effect on the performance improvement of Doc2Vec. While DeepWalk can capture more neighbor information by random walking, improve the generated embedding quality, and make better recommendations.

Ablation Study

We evaluated how each of the components of HANRec affects the results. We report the results on two datasets after removing a specific part. HANRec-C, HANRec-Homo, HANRec-Hete, HANRec-Att respectively represent our model without connecting potential neighbors, homogeneous aggregation, heterogeneous aggregation, and the attention mechanism. HANRec is the complete model we proposed. It can be seen from Fig. 5 that regardless of removing the connecting component, homogeneous aggregation, heterogeneous aggregation, or the attention mechanism, the performance of HANRec shows attenuation, and removing the attention mechanism has less impact on performance than the former two. This makes intuitive sense. The other three cases will lose much information in the graph, which is not conducive to the entity's high-quality embedding. It is worth noting that after removing the connecting compo-

(a) Connecting Strategies: MovieLen-MAE

(b) Connecting Strategies: MovieLen-RMSE

(c) Connecting Strategies: AMiner-AUC
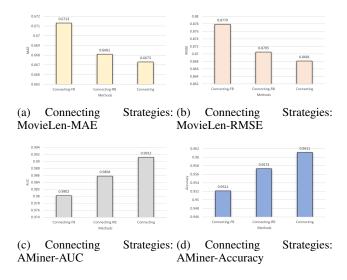
(d) Connecting Strategies: AMiner-Accuracy

Figure 6: The performance of different connecting methods on the MovieLen and the AMiner dataset.

nent, the performance of the model has dropped a lot, indicating that the connection method we designed can provide an essential reference for recommendations. This also further proves that the connecting component, aggregation methods, and attention mechanism are practical. With their joint contribution, HANRec can perform well in the recommendation task.

We also explored the performance of different connection strategies. Connecting-FB represents the feature-based connecting method. When we explore neighbors with potential relationships, Connecting-FB only uses the feature information of entities on the path, not the relationship information between entities. In this case, formula 1 can be rewritten as: $f_{i,j} = MLP(h_k^0 \oplus h_j^0)$. Connecting-RB means the relation-based connecting method. Connecting-RB only uses the relationship information between entities on the path, not the feature information of entities. So formula 1 can be rewritten as: $f_{i,j} = MLP(e_{i,k} \oplus e_{k,j})$. Connecting is the complete connecting method we proposed. As shown in Fig. 6, whether it is a feature-based connection or a relationship-based connection, the model's performance is not as good as the connection method we use. This is intuitive: for example, in a movie recommendation network, if two users have watched a movie, then both the rating scores of the movie and the characteristics of the movie and users can provide a vital reference for the relationship between users.

Parameter Analysis

In this part, we analyze the effect of embedding dimension d of the entity's latent representation p_i on the performance of HANRec. Fig. 7 presents the performance comparison of the embedding dimension on the MovieLen and the AMiner dataset. In general, with the increase of the embedding dimension, the performance first increases and then decreases. When expanding the embedding dimension from 32 to 128 can improve the performance significantly. However, with the embedding dimension of 256, HANRec degrades the
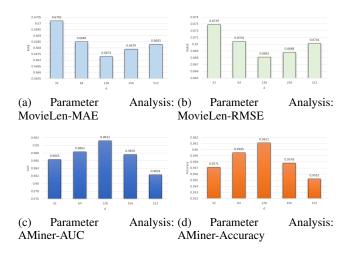

(a) Parameter Analysis: MovieLen-MAE

(b) Parameter Analysis: MovieLen-RMSE

(c) Parameter Analysis: AMiner-AUC

(d) Parameter Analysis: AMiner-Accuracy

Figure 7: Effect of dimension d on the MovieLen and the AMiner dataset.

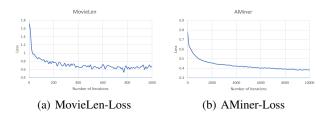(a) MovieLen-Loss

(b) AMiner-Loss

Figure 8: Performance with respect to the number of iterations on the MovieLen and the AMiner dataset.

performance. It demonstrates that using a large number of the embedding dimension has powerful representation. Nevertheless, if the embedding dimension is too large, the complexity of our model will significantly increase. Therefore, we need to find a proper length of embedding to balance the trade-off between the performance and the complexity.

We also study the performance change w.r.t. the number of iterations when the learning rate is 0.001, and the training ratio is 90%. As shown in Fig. 8, we can see that the proposed model has a fast convergence rate, and about 400 iterations are required for the MovieLen dataset, while about 8000 iterations are required for the AMiner dataset.

Conclusion

In this paper, we propose a heterogeneous attributed network framework with a connecting method, called HANRec, to address the recommendation task in heterogeneous graph. By gathering multiple types of neighbor information and using the attention mechanism, HANRec efficiently generates embeddings of each entity for downstream tasks. The experiment results on two real-world dataset can prove HANRec outperform state-of-the art models for the task of recommendation and link prediction.

References

Adomavicius, G.; and Tuzhilin, A. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering* 17(6): 734–749.

Baesens, B.; Viaene, S.; Van Gestel, T.; Suykens, J. A. K.; Dedene, G.; De Moor, B.; and Vanthienen, J. 2000. An empirical assessment of kernel type performance for least squares support vector machine classifiers. In *KES'2000. Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies. Proceedings (Cat. No.00TH8516)*, volume 1, 313–316 vol.1.

Berg, R. v. d.; Kipf, T. N.; and Welling, M. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* .

Bobadilla, J.; Ortega, F.; Hernando, A.; and Bernal, J. 2012. A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-based systems* 26: 225–238.

Bronstein, M. M.; Bruna, J.; LeCun, Y.; Szlam, A.; and Vandergheynst, P. 2017. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine* 34(4): 18–42.

Cen, Y.; Zou, X.; Zhang, J.; Yang, H.; Zhou, J.; and Tang, J. 2019. Representation learning for attributed multiplex heterogeneous network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1358–1368.

Chen, X.; Liu, D.; Xiong, Z.; and Zha, Z.-J. 2020. Learning and fusing multiple user interest representations for micro-video and movie recommendations. *IEEE Transactions on Multimedia* 23: 484–496.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *arXiv preprint arXiv:1606.09375* .

Derr, T.; Ma, Y.; and Tang, J. 2018. Signed graph convolutional networks. In *2018 IEEE International Conference on Data Mining (ICDM)*, 929–934. IEEE.

Dong, Y.; Chawla, N. V.; and Swami, A. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 135–144.

Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, E.; Tang, J.; and Yin, D. 2019a. Graph Neural Networks for Social Recommendation. 417–426. doi:10.1145/3308558.3313488.

Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, E.; Tang, J.; and Yin, D. 2019b. Graph Neural Networks for Social Recommendation. In *The World Wide Web Conference*, 417–426. New York, NY, USA: Association for Computing Machinery.

Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, E.; Tang, J.; and Yin, D. 2019c. Graph neural networks for social recommendation. In *The World Wide Web Conference*, 417–426.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 855–864.

He, R.; and McAuley, J. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 191–200. IEEE.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, 173–182.

Herlocker, J. L.; Konstan, J. A.; Terveen, L. G.; and Riedl, J. T. 2004. Evaluating Collaborative Filtering Recommender Systems 22(1): 5–53. ISSN 1046-8188.

Hidasi, B.; and Karatzoglou, A. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM international conference on information and knowledge management*, 843–852.

Kang, W.-C.; and McAuley, J. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, 197–206. IEEE.

Kefalas, P.; Symeonidis, P.; and Manolopoulos, Y. 2018. Recommendations based on a heterogeneous spatio-temporal social network. *World Wide Web* 21(2): 345–371.

Keller, J. M.; Gray, M. R.; and Givens, J. A. 1985. A fuzzy K-nearest neighbor algorithm. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-15(4): 580–585.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* .

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

Le, Q.; and Mikolov, T. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, 1188–1196. PMLR.

Lekakos, G.; and Caravelas, P. 2008. A hybrid approach for movie recommendation. *Multimedia tools and applications* 36(1): 55–70.

Li, J.; Ren, P.; Chen, Z.; Ren, Z.; Lian, T.; and Ma, J. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 1419–1428.

Liu, Q.; Zeng, Y.; Mokhosi, R.; and Zhang, H. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1831–1839.

Ma, H.; Yang, H.; Lyu, M. R.; and King, I. 2008. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, 931–940.

Mnih, A.; and Salakhutdinov, R. R. 2007. Probabilistic matrix factorization. *Advances in neural information processing systems* 20: 1257–1264.

Pal, S. K.; and Mitra, S. 1992. Multilayer perceptron, fuzzy sets, classifiaction .

Pazzani, M. J.; and Billsus, D. 2007. *Content-Based Recommendation Systems*, 325–341. Berlin, Heidelberg: Springer-Verlag. ISBN 9783540720782.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014a. DeepWalk: Online Learning of Social Representations. 701–710. Association for Computing Machinery.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014b. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710.

Ramos, J. 2003. Using TF-IDF to Determine Word Relevance in Document Queries.

Rendle, S.; Freudenthaler, C.; and Schmidt-Thieme, L. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, 811–820.

Ricci, F.; Rokach, L.; and Shapira, B. 2011. Introduction to recommender systems handbook. In *Recommender systems handbook*, 1–35. Springer.

Safavian, S. R.; and Landgrebe, D. 1991. A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics* 21(3): 660–674.

Schafer, J. B.; Frankowski, D.; Herlocker, J.; and Sen, S. 2007. Collaborative filtering recommender systems. In *The adaptive web*, 291–324. Springer.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15(1): 1929–1958.

Stallings, J.; Vance, E.; Yang, J.; Vannier, M. W.; Liang, J.; Pang, L.; Dai, L.; Ye, I.; and Wang, G. 2013. Determining scientific impact using a collaboration index. *Proceedings of the National Academy of Sciences* 110(24): 9680–9685.

Tan, Y. K.; Xu, X.; and Liu, Y. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st workshop on deep learning for recommender systems*, 17–22.

Tang, J.; Aggarwal, C.; and Liu, H. 2016. Recommendations in Signed Social Networks. In *Proceedings of the 25th International Conference on World Wide Web*, 31–40. International World Wide Web Conferences Steering Committee.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, 1067–1077.

Tang, J.; Zhang, J.; Yao, L.; Li, J.; Zhang, L.; and Su, Z. 2008. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 990–998.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* .

Vazquez, A.; Flammini, A.; Maritan, A.; and Vespignani, A. 2003. A Global protein function prediction in protein-protein interaction networks. *Nature biotechnology* 21: 697–700. doi:10.1038/nbt825.

Wang, H.; Wang, N.; and Yeung, D.-Y. 2015. Collaborative Deep Learning for Recommender Systems. 1235–1244. Association for Computing Machinery.

Wang, H.; Zhao, M.; Xie, X.; Li, W.; and Guo, M. 2019a. Knowledge graph convolutional networks for recommender systems. In *The world wide web conference*, 3307–3313.

Wang, Y.; Duan, Z.; Liao, B.; Wu, F.; and Zhuang, Y. 2019b. Heterogeneous attributed network embedding with graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 10061–10062.

Wu, J.; Chen, L.; Yu, Q.; Han, P.; and Wu, Z. 2015. Trust-aware media recommendation in heterogeneous social networks. *World Wide Web* 18(1): 139–157.

Wu, L.; Sun, P.; Fu, Y.; Hong, R.; Wang, X.; and Wang, M. 2019. A neural influence diffusion model for social recommendation. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, 235–244.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Chang, X.; and Zhang, C. 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 753–763.

Yang, B.; Lei, Y.; Liu, J.; and Li, W. 2017. Social Collaborative Filtering by Trust. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39(8): 1633–1647.

Zhao, J.; Zhou, Z.; Guan, Z.; Zhao, W.; Ning, W.; Qiu, G.; and He, X. 2019. Intentgc: a scalable graph convolution framework fusing heterogeneous information for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2347–2357.

Zhong, T.; Zhang, S.; Zhou, F.; Zhang, K.; Trajcevski, G.; and Wu, J. 2020. Hybrid graph convolutional networks with multi-head attention for location recommendation. *World Wide Web* 23(6): 3125–3151.

Zhou, J.; Cui, G.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M. 2018. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434* .

Zhu, Y.; Lin, Q.; Lu, H.; Shi, K.; Qiu, P.; and Niu, Z. 2021. Recommending scientific paper via heterogeneous knowledge embedding based attentive recurrent neural networks. *Knowledge-Based Systems* 215: 106744.