Applied Intelligence
https://doi.org/10.1007/s10489-022-03340-7

Connecting latent relationships over heterogeneous attributed network for recommendation

Ziheng Duan[1,2] · Yueyang Wang[1] · Weihao Ye[1] · Qilin Fan[1] · Xiuhua Li[1]

Accepted: 2 February 2022

Abstract

Recently, deep neural network models for graph-structured data have been demonstrated 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 among 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-art methods.

Keywords Deep learning · Recommendatino system · Heterogeneous attributed network · Graph neural network

1 Introduction

As it becomes more convenient for users to generate data than before, mass data that appeared on the Internet are no longer with simple structure but usually with more complex types and structures. The recommendation system, which helps users discover items of interest, is attracting increasing attention, but it is also facing more and more challenges [1, 2]. Traditional recommendation methods, such as matrix factorization [3], 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 [4].

As the data dusers and items in the recommendation system can be naturally organized as a graph structure, graph-based deep learning methods have been applied in recommendation fields [5] and alleviate the enormous amount and the sparsity problem to some extent. Meanwhile, heterogeneous attributed network (HAN) [6], 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. Therefore, designing a graph-based recommendation method to mine the information in HAN is a promising direction [7].

Recently, Graph Neural Network (GNN), 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. GNNs also have an additional huge benefit: they can be learning distributed - hence secure and safe, and at the same time they can be used for the upcoming explainability topic [8]. GNN based recommendation methods have proved to be successful to some extent because they

✉ Yueyang Wang
yueyangw@cqu.edu.cn

Ziheng Duan
duanziheng@zju.edu.cn

1 School of Big Data and Software Engineering, Chongqing University, Chongqing, 401331, China

2 College of Control Science and Engineering, Zhejiang University, Zhejiang, 310027, China

Published online: 21 March 2022

could simultaneously encode the graph structure and the attributes of nodes [9]. However, the advances of GNNs are primarily concentrated on homogenous graphs, so they still encounter limitations while utilizing 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 among users and items from various aspects [5]. On the one hand, different types of nodes have various attributes that cannot be directly applied due to the different dimensions of node attributes [6]. For example, a "user" is associated with attributes like interests and the number of watched movies in movie rating graphs, while a "movie" has attributes like genres and publication years. On the other hand, the influence of various relationships is different, which should not be treated equally [10]. 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 [11, 12]. However, the reality is that some potential relationships of entities are not directly connected but implicit [13]. For example, as shown in Fig. 1, both user one and user two score movie one as the same rating 4, which reflects that the interests of the two users are similar. Analogously, movie one and movie two have the same genres. There are some potential relationships between the two movies. Nevertheless, these implicit relationships are hardly captured by GNNs [14, 15]. Meanwhile, the degree of rating could reflect the preferences of users. For instance, user one rates movie two as 5, but movie three as 1, from which we may infer that user two prefers movie two to movie three. Therefore, explicitly modeling entities with potential relationships to provide references for each other's recommendations and distinguishing the rating weights are significant.

To better overcome these challenges mentioned above, we propose a neural network model to deal with <u>H</u>eterogeneous <u>A</u>ttributed <u>N</u>etwork for <u>Rec</u>ommendation, 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 among 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 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 [16] to measure the different impacts of heterogeneous nodes. The learned parameters in the two aggregations are different to model diverse patterns of the heterogeneous graph structure. Finally, we use the entities' high-quality embeddings to make corresponding recommendations through the

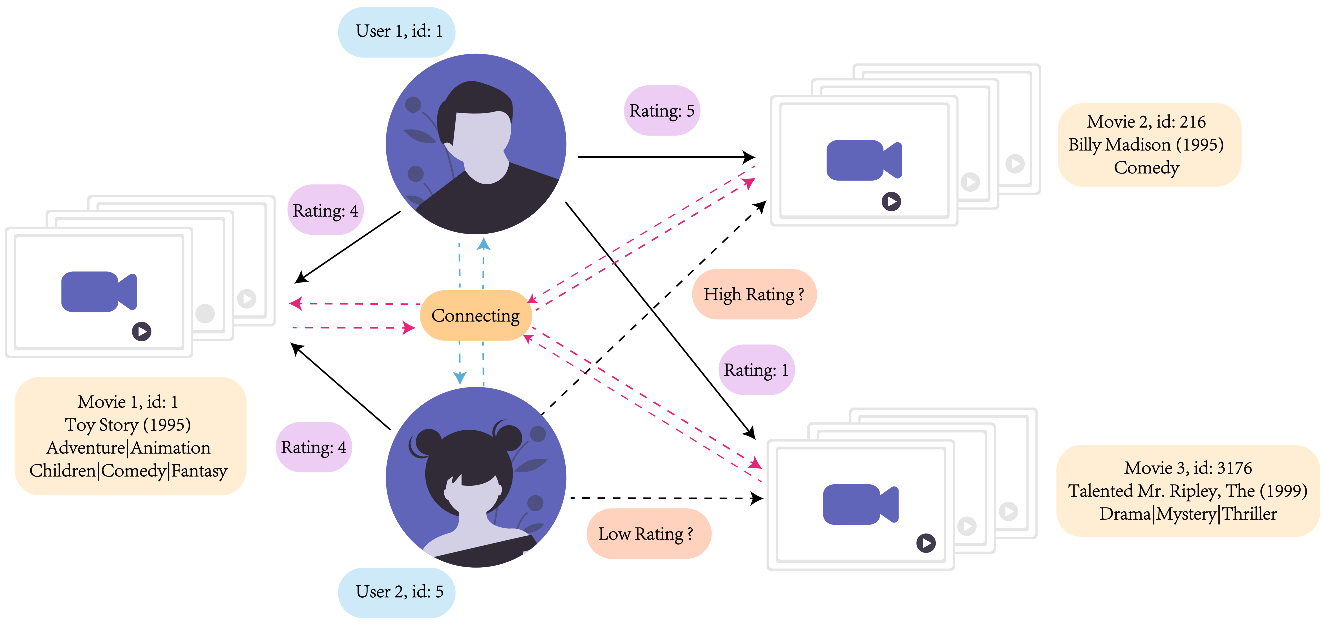


Fig. 1 An example of connecting the users and movies from the real-world dataset: MovieLens. The solid black line represents the rating scores of movies by users. We use the blue dashed line to indicate the potential relationships between users and the red dashed line to demonstrate the potential relationships 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

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, which is usually ignored by existing methods.
– We present a novel framework for the recommendation, which makes full use of the heterogeneity in graphs and utilizes two 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.

The rest of this paper is organized as follows. Section 2 introduces the related works on the recommendation algorithms. Section 3 describes the formulation expression of recommendation problem, notations used in the paper, and the heterogeneous attributed network embedding method HANRec we proposed. Experiments and detailed analysis are reported in Section 4. Finally, we conclude the paper in Section 5.

2 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 [17, 18]. Traditional methods to deal with recommendation problems mainly include content-based recommendation algorithms and collaborative filtering algorithms. The idea of Content-Based Recommendation (CB) [19] 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. Collaborative Filtering (CF) [20] aims to find some similarity through the behaviors of groups and make recommendations for users based on this similarity. CF methods, which are based on matrix decomposition, decompose the user rating matrix into user matrix U and product matrix V. U and V are recombined to get a new user rating matrix to predict unknown ratings. Matrix decomposition includes Funk-SVD model [21], PMF model [22], etc.

Deep learning, as an emerging method, also has many achievements of learning on graph structure data [23]. 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. [24] 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 a lower dimension 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. [25] 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 [26] (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 [27]. 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 [28]. The general recommendation considers users to have static interest preferences and models the degree of matching among 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) [29], based on RNN [30], and based on attention and self-attention mechanisms [31]. 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, (Graph Convolutional Matrix Completion) proposed by Berg, 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 have also developed 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 [32] models user preferences based on users' social relationships and historical behaviors, and it uses the GraphSAGE framework 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. [5] learns the low dimensional representation of users and products through graph neural networks. GraphRec uses 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 [33] 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 representations. IntentGC [34] 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

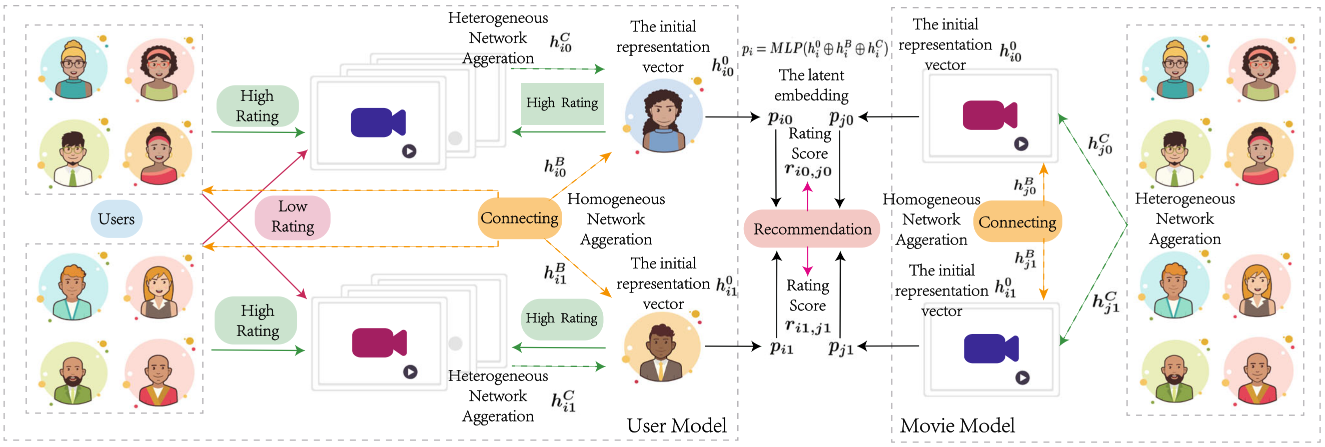

Users
High Rating
Low Rating
High Rating
Heterogeneous Network Aggeration
h_{i0}^{C}
High Rating
The initial representation vector h_{i0}^{0}
h_{i0}^{B}
Connecting
Homogeneous Network Aggeration
h_{i1}^{B}
The initial representation vector h_{i1}^{0}
High Rating
Heterogeneous Network Aggeration
h_{i1}^{C}
User Model
$p_i = MLP(h_i^0 \oplus h_i^B \oplus h_i^C)$
The latent embedding
p_{i0}
p_{j0}
Rating Score $r_{i0,j0}$
Recommendation
Rating Score $r_{i1,j1}$
p_{i1}
p_{j1}
The initial representation vector h_{i0}^{0}
h_{j0}^{C}
h_{j0}^{B}
Homogeneous Network Aggeration
Connecting
h_{j1}^{B}
The initial representation vector h_{i1}^{0}
h_{j1}^{C}
Heterogeneous Network Aggeration
Movie Model

Fig. 2 The architecture of HANRec. It contains four major components: connecting potential neighbors, homogeneous aggregation, heterogeneous aggregation and rating generation

this paper, we propose a graph neural network framework that can connect potential relationships in the network to fill this gap.

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

3.1 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 graph's heterogeneity is reflected as: the number of node types plus the number of edge types is bigger than two. On the other hand, when both the number of node types and the number of edge types are equal to one, we call it a homogeneous graph [35]. Attributed graphs mean that each node in graphs has corresponding attributes. Attributed entities 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 can be treated 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 $r_{i,j}$ 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.), which 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 $(r_{i,j}, v_i, v_j)$, $r_{i,j}$ represents the relationship

Table 1 Symbols' definition

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 entities which are neighbors of entity v_i.
$\overline{N(v_i)}$	The set of entities which are not the neighbors 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.
$\boldsymbol{p_i}$	The embedding vector of entity v_i.
d	The embedding dimension of $\boldsymbol{p_i}$.
$\boldsymbol{e_{i,j}}$	The rating embedding for the rating level (entity v_i to v_j).
$\boldsymbol{h_i^0}$	The initial representation vector of entity v_i.
$\boldsymbol{h_i^B}$	The influence embedding among entity v_i and entities of the same type.
$\boldsymbol{h_i^C}$	The influence embedding among entity v_i and entities of different types.
$\boldsymbol{f_{i,j}}$	The opinion-aware interaction representation between entity v_i and v_j.
$\alpha^*_{i,j}$	Attention parameter between entities of the same type (v_j to v_i).
$\beta^*_{i,j}$	Attention parameter between entities of different types (v_j to v_i).
$\alpha_{i,j}$	The normalized attention parameter between entities of the same type (v_j to v_i).
$\beta_{i,j}$	The normalized attention parameter between entities of different types (v_j to v_i).
$\boldsymbol{W}, \boldsymbol{b}$	The weight and bias in neural networks.

Vectors and matrices are bolded

between v_i and v_j. A link prediction model learns a score function, and outputs the relationship $r_{i,j}$ between v_i and v_j: $r_{i,j} = Link\_Prediction\_Model(v_i, v_j)$. In most cases, the value of $r_{i,j}$ 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.

3.2 An Overview of HANRec

HANRec consists of four components: Connecting Potential Neighbors, Homogeneous Network Aggregation, Heterogeneous Network Aggregation, and Recommendation Generation. Connecting Potential Neighbors is designed to fully explore the influence among 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 among users. However, the interaction among users is important, and there may be potential influences among 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 among 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 the user or movie features and generate the final representation embeddings. 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.

3.3 Connecting potential neighbors

We design this component to fully explore the influence among neighbors with potential connections, which are not directly connected in the original graph. 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; $N(v_i)$ represents the set of neighbors of v_i and $\overline{N(v_i)}$ represents the set of nodes which are not the neighbors of v_i. We can infer the influence of v_j on v_i from the common neighbor v_k:

$$\boldsymbol{f}_{i,j} = MLP\left(\boldsymbol{h}_k^0 \oplus \boldsymbol{e}_{i,k} \oplus \boldsymbol{h}_j^0 \oplus \boldsymbol{e}_{k,j}\right), \tag{1}$$

where $\boldsymbol{f}_{i,j}$ is the opinion-aware interaction representation between entity v_i and v_j; MLP means Multilayer Perceptron [36]; $\oplus$ represents the concatenate operator of two vectors; $\boldsymbol{h}_k^0$ and $\boldsymbol{h}_j^0$ represent the initial features of v_k and v_j, respectively; $\boldsymbol{e}_{i,k}$ and $\boldsymbol{e}_{k,j}$ represent the rating embeddings for the rating level between v_i, v_k and v_k, v_j, respectively.

The design to formula (1) is based on the assumption when evaluating the influence among entities, the entity features and the rating information are both critical, and they can reflect how significant the impact is. For instance, if two people have a similar taste for the same movie, then the two people should have a more significant influence on each other. Referred to [5], we use the dense learnable vector $e_{i,j}$, which can help us better characterize each rating score in the embedding space. For example, in a five-star rating system, we have five rating levels: {1, 2, 3, 4, 5}. Instead of directly using the rating level number, we first randomly initialize five rating embeddings: $\{e_1, e_2, e_3, e_4, e_5\}$. Then all $e_{i,j}$ will belong to one of those five rating embeddings and they will be jointly learned during the training stage.

Subsequent ablation study in Section 4.7 also proved the advantages of this design. Through this connecting way, we use $\boldsymbol{f}_{i,j}$ 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 among entities in the path. For example, there may not be a direct connection among 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.

3.4 Homogeneous network aggregation

Homogeneous Network Aggregation aims to learn the influence embedding $\boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{B}}$, 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 $\boldsymbol{e}_{i,j}$, the opinion-aware interaction representation $\boldsymbol{f}_{i,j}$ can be expressed as:

$$\boldsymbol{f}_{i,j} = MLP\left(\boldsymbol{h}_{\boldsymbol{j}}^{\boldsymbol{0}} \oplus \boldsymbol{e}_{i,j}\right). \tag{2}$$

In this way, we can get a vector that combines entity features and rating information. So we call $\boldsymbol{f}_{i,j}$ the "opinion-aware" interaction representation. The attention parameter $\alpha^*_{i,j}$ reflecting the influence of different entities v_i and v_j is designed as:

$$\alpha^*_{i,j} = \boldsymbol{w}_{\boldsymbol{2}}^{\boldsymbol{T}} \cdot \sigma\left(\boldsymbol{W}_{\boldsymbol{1}} \cdot \left[\boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{0}} \oplus \boldsymbol{h}_{\boldsymbol{j}}^{\boldsymbol{0}}\right] + \boldsymbol{b}_{\boldsymbol{1}}\right) + \boldsymbol{b}_{\boldsymbol{2}}, \tag{3}$$

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

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

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:

$$\boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{B}} = \sum_{j \in B(v_i) \cap (N(v_i) \cup N'(v_i))} \left(\alpha_{i,j} * \boldsymbol{f}_{i,j}\right) \tag{5}$$

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.

3.5 Heterogeneous network aggregation

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

$$\boldsymbol{f}_{i,j} = MLP\left(\boldsymbol{h}_{\boldsymbol{j}}^{\boldsymbol{0}} \oplus \boldsymbol{e}_{i,j}\right) \tag{6}$$

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

$$\beta^*_{i,j} = \boldsymbol{w}_{\boldsymbol{4}}^{\boldsymbol{T}} \cdot \sigma\left(\boldsymbol{W}_{\boldsymbol{3}} \cdot \left[\boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{0}} \oplus \boldsymbol{h}_{\boldsymbol{j}}^{\boldsymbol{0}}\right] + \boldsymbol{b}_{\boldsymbol{3}}\right) + \boldsymbol{b}_{\boldsymbol{4}} \tag{7}$$

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

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

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

$$\boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{C}} = \sum_{j \in C(v_i) \cap (N(v_i) \cup N'(v_i))} \left(\beta_{i,j} * \boldsymbol{f}_{i,j}\right) \tag{9}$$

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.

3.6 Recommendation generation

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

$$\boldsymbol{p}_{\boldsymbol{i}} = MLP\left(\boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{0}} \oplus \boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{B}} \oplus \boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{C}}\right), \tag{10}$$

where $\boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{0}}$ is the initial representation of entity v_i, such as the user's preferences in the user-movie-rating network, the movie genres, etc. $\boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{B}}$ and $\boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{C}}$ 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 embeddings that aggregate a variety of information ($\boldsymbol{p}_{\boldsymbol{i}}$ and $\boldsymbol{p}_{\boldsymbol{j}}$), their relationship can be measured as:

$$r_{i,j} = MLP(\boldsymbol{p}_{\boldsymbol{i}} \oplus \boldsymbol{p}_{\boldsymbol{j}}). \tag{11}$$

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

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          $\boldsymbol{f_{i,m}} = MLP(\boldsymbol{h_m^0} \oplus \boldsymbol{e_{i,m}})$;
           // 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                  $\boldsymbol{f_{i,n}} = MLP(\boldsymbol{h_m^0} \oplus \boldsymbol{e_{i,m}} \oplus \boldsymbol{h_n^0} \oplus \boldsymbol{e_{m,n}})$;
                   // The opinion-aware interaction representation between entity $v_i$
                      and $v_n$
10             end
11         end
12     end
       // Connecting Potential Neighbors
13     $\alpha^*_{i,j} = \boldsymbol{w_2^T} \cdot \sigma(\boldsymbol{W_1} \cdot [\boldsymbol{h_i^0} \oplus \boldsymbol{h_j^0}] + \boldsymbol{b_1}) + \boldsymbol{b_2}, \alpha_{i,j} = \frac{exp(\alpha^*_{i,j})}{\sum_{k \in B(v_i) \cap (N(v_i) \cup N'(v_i))} exp(\alpha^*_{i,k})}$;
       // The attention parameters in homogeneous aggregation
14     $\boldsymbol{h_i^B} = \sum_{v_k \in B(v_i) \cap (N(v_i) \cup N'(v_i))} (\alpha_{i,k} * \boldsymbol{f_{i,k}})$;
       // Homogeneous Aggregation
15     $\beta^*_{i,j} = \boldsymbol{w_4^T} \cdot \sigma(\boldsymbol{W_3} \cdot [\boldsymbol{h_i^0} \oplus \boldsymbol{h_j^0}] + \boldsymbol{b_3}) + \boldsymbol{b_4}, \beta_{i,j} = \frac{exp(\beta^*_{i,j})}{\sum_{k \in C(v_i) \cap (N(v_i) \cup N'(v_i))} exp(\beta^*_{i,k})}$;
       // The attention parameters in heterogeneous aggregation
16     $\boldsymbol{h_i^C} = \sum_{v_k \in C(v_i) \cap (N(v_i) \cup N'(v_i))} (\beta_{i,k} * \boldsymbol{f_{i,k}})$;
       // Heterogeneous Aggregation
17     $\boldsymbol{p_i} = MLP(\boldsymbol{h_i^0} \oplus \boldsymbol{h_i^B} \oplus \boldsymbol{h_i^C})$;
       // Fuse $v_i$'s initial feature and the information aggregated from
          homogeneous and heterogeneous neighbors to obtain the final
          representation embedding $\boldsymbol{p_i}$
18     $P \leftarrow \boldsymbol{p_i}$;
       // Add $\boldsymbol{p}$ to P
19 end
20 $r_{1,2} = MLP(\boldsymbol{p_1} \oplus \boldsymbol{p_2})$;
   // Compute the relationship $r_{1,2}$ between two entity $v_1$ and $v_2$
21 return $r_{1,2}$
```

3.7 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 [5]:

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

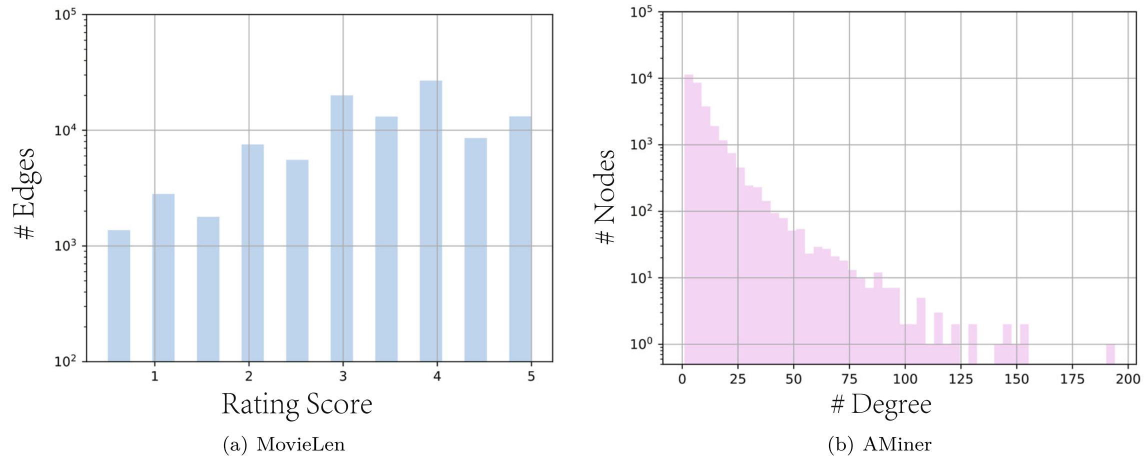


Fig. 3 Characteristics of the MovieLens and the AMiner dataset

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 [37] 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 adopt a dropout strategy [38] 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 is disabled during the test, the entire network will be used for prediction. The whole algorithm framework is shown in algorithm 1.

4 Experiments

4.1 Datasets

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

We use MovieLens Latest Dataset [1], a common data set in the field of movie recommendation, for the recommendation task. 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 users' ratings of movies, and there is no edge connection among users or among movies. So in this case, the original edges only connect users and movies, that is, different types of nodes. While connecting potential neighbors, we better characterize the impact of users on users and movies on movies, that is, the effect among nodes of the same type to generate high-quality node embeddings. MovieLens also includes the movie's genre attributes and timestamps of ratings. 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 MovieLens dataset, our model focuses on three kinds of embeddings, including user embedding $\boldsymbol{h}_i^{\mathbf{0}}$, where i belongs to the user index, movie embedding $\boldsymbol{h}_j^{\mathbf{0}}$, which is composed of the embedding of movie genres, and j belongs to the movie index, and opinion embedding $\boldsymbol{e}_{i,j}$. They are initialized randomly and learned 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 [39]. The opinion embedding matrix $\boldsymbol{e}_{i,j}$ 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 $\boldsymbol{e}$ 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. Figure 3a shows the rating score distribution of edges in the MovieLens 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[2] [40]. We construct the relationship between an author and one paper when the author published this paper. We also generate the relationships among authors when they are co-authors and generate the relationships among papers when there are citation relationships. The authors' initial

[1]https://grouplens.org/datasets/movielens/latest/

[2]https://www.aminer.cn/aminernetwork

attribute contains research interests, published papers, and the total number of citations. [41]. The title and abstract can represent the initial attribute of papers. In this author-paper-network, we treat the weights of edges as binary. In the experiment part, we selected all papers from the famous venues[3] in eight research topics [42] 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. Figure 3b shows the node distribution of the AMiner dataset. We can find that most nodes have a small number of 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.

4.2 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 ($RMSE$):

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|p_i - a_i|, \tag{13}$$

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

where a is the actual target and p is the 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 predictive performance:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{15}$$

$$AUC = \frac{n' - 0.5n''}{n}, \tag{16}$$

where TP, TN, FP and FN respectively stand for true positive, true negative, false positive and false negative. n is the number of independent comparisons. n' is the number of times the missing link having a higher score and n'' is the number of times they have the same score. In general, the value of AUC will be between 0.5 and 1. Higher values of accuracy and AUC indicate better predictive performance.

[3] 1. IEEE Trans. Parallel Distrib. Syst; 2. STOC; 3. IEEE Communications Magazine; 4. ACM Trans. Graph; 5.CHI; 6. ACL; 7. CVPR; 8. WWW

4.3 Baselines

The methods in our comparative evaluation are as follows:

- **Doc2Vec** [43] 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 embeddings of authors and papers.
- **DeepWalk** [24] uses random walk to sample nodes in the graph to get the node embeddings. 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** [44] 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 5, just the same as the original paper.
- **SoRec** [3] 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. λ_C = 10, and $\lambda_U = \lambda_V = \lambda_Z = 0.001$.
- **LightGCN** [45] simplifies the design of GCN, which removes the feature transformation and nonlinear activation, to make it more concise and appropriate for recommendation. We use the same parameters introduced in the original paper: the default learning rate is 0.001 and the L_2 regularization coefficient λ is 0.0001.
- **GATNE** [35] 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** [5] jointly captures interactions and opinions in the user-item graph. For the MovieLens 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.

4.4 Training details

For the task of recommendation, we use the MovieLens dataset. The recommendation task's goal is to predict the rating score of one user to one movie. $x\%$ of the scoring

Table 2 Results comparison of the recommendation

Train Edges	DeepWalk	LINE	SoRec	LightGCN	GATNE	GraphRec	HANRec
30%	0.8514/1.0578	0.8614/1.0687	0.8278/1.0324	0.8307/1.0498	0.7923/0.9848	0.7824/0.9748	**0.7751/0.9624**
40%	0.8258/1.0014	0.8413/1.0342	0.7891/0.9936	0.8125/1.0123	0.7774/0.9607	0.7528/0.9555	**0.7419/0.9355**
50%	0.7869/0.9817	0.8017/0.9959	0.7417/0.9615	0.7828/0.9825	0.7319/0.9487	0.7347/0.9379	**0.7128/0.9132**
60%	0.7521/0.9678	0.7758/0.9816	0.7314/0.9504	0.7599/0.9707	0.7217/0.9215	0.7191/0.9189	**0.6981/0.9047**
70%	0.7331/0.9497	0.7525/0.9607	0.7257/0.9407	0.7409/0.9534	0.7047/0.9158	0.7055/0.9107	**0.6823/0.8925**
80%	0.7250/0.9418	0.7332/0.9511	0.7192/0.9339	0.7287/0.9438	0.6954/0.9057	0.6966/0.9071	**0.6753/0.8855**
90%	0.7148/0.9375	0.7241/0.9403	0.6957/0.9048	0.7173/0.9340	0.6824/0.8907	0.6807/0.8827	**0.6673/0.8681**

$MAE/RMSE$ are evaluation metrics. The best results are bolded

records are used for training, and the remaining (100-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 truch, 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-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 among 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 [46], which is discussed in detail in the Section 4.6.

The proposed HANRec is implemented on the basis of Pytorch 1.4.0.[4] 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 is used to optimize the parameters of HANRec. The performance results are recorded on the test set after 2000 iterations on the MovieLens dataset and 20000 iterations on the AMiner dataset. The parameters for the baseline algorithms are initialized as in the corresponding papers and are then carefully tuned to achieve optimal performance.

4.5 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 the MovieLen dataset. We find that no matter how much x is, the performance of DeepWalk and LINE, 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. Although LightGCN has proved successful in simple collaborative filtering, its performance in more complex situations is weaker than deep learning methods. For example, the MovieLens dataset contains entity attributes of different types and rating information. It is difficult for LightGCN to fully capture information of this dataset using only linear transformation. GATNE's embedding of edge heterogeneity and GraphRec's social aggregation cannot be fully utilized in this case. In this dataset, there is no connection among users. Without a well-designed connecting strategy, GraphRec cannot explore the potential influence among users, which is essential in providing accurate recommendations. Nevertheless, these two methods' performance is generally better than the previous methods, proving 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 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 and SoRec, have higher performance than Doc2Vec, indicating that in this case, the graph structure information

[4]https://pytorch.org/

Table 3 Results comparison of the link prediction

Train Edges	Doc2Vec	DeepWalk	LINE	SoRec	LightGCN	GraphRec	GATNE	HANRec
30%	0.8074/0.7335	0.8693/0.8206	0.7911/0.7214	0.9228/0.8547	0.9038/0.8287	0.9241/0.8613	0.9341/0.8705	**0.9612/0.8982**
40%	0.8158/0.7407	0.8873/0.8345	0.8017/0.7275	0.9317/0.8602	0.9126/0.8311	0.9333/0.8752	0.9419/0.8787	**0.9647/0.9032**
50%	0.8243/0.7505	0.8948/0.8415	0.8155/0.7357	0.9394/0.8739	0.9157/0.8389	0.9517/0.8907	0.9553/0.8871	**0.9679/0.9101**
60%	0.8395/0.7612	0.9105/0.8631	0.8209/0.7398	0.9534/0.8907	0.9332/0.8541	0.9571/0.8988	0.9702/0.9015	**0.9758/0.9223**
70%	0.8548/0.7765	0.9358/0.8758	0.8288/0.7451	0.9627/0.9015	0.9529/0.8607	0.9673/0.9056	0.9723/0.9108	**0.9807/0.9409**
80%	0.8702/0.7893	0.9501/0.8921	0.8371/0.7473	0.9708/0.9077	0.9610/0.8725	0.9728/0.9099	0.9759/0.9155	**0.9881/0.9514**
90%	0.8771/0.7973	0.9573/0.8967	0.8429/0.7506	0.9782/0.9103	0.9677/0.8831	0.9792/0.9156	0.9806/0.9227	**0.9912/0.9611**

$AUC/Accuracy/c_{\Delta}$ are evaluation metrics. The best results are bolded

is more important than the initial information of nodes. In the Aminer dataset, there are three kinds of relationships: the coauthor relationship among authors, the publication relationship among authors and papers, and the citation relationship among papers. In addition, authors and papers also have their initial features. It is difficult for LightGCN to learn the high-quality representation of each entity by using simple linear transformations, so its performance is weaker than deep learning methods such as GATNE. 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 homogeneity and heterogeneity of the graph and uses 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 MovieLens and the AMiner datasets for both tasks.

4.6 Cold-start problem

The cold-start problem [46] refers to the difficulty of the recommendation system to give high-quality recommendations when there is insufficient data. According to [47], 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, we study these three cold-start problems by modifying the data distribution in the training dataset. (1) For the user cold-start problem, it solves the problem of how to make personalized recommendations for new users. Nowadays, many methods [48] require users to give feedback on items when users first log in to the system. They collect information about the user's interest in these items and then recommend items similar to these items to the user. We can replace the information provided when they log in by adding a tiny amount of information about the

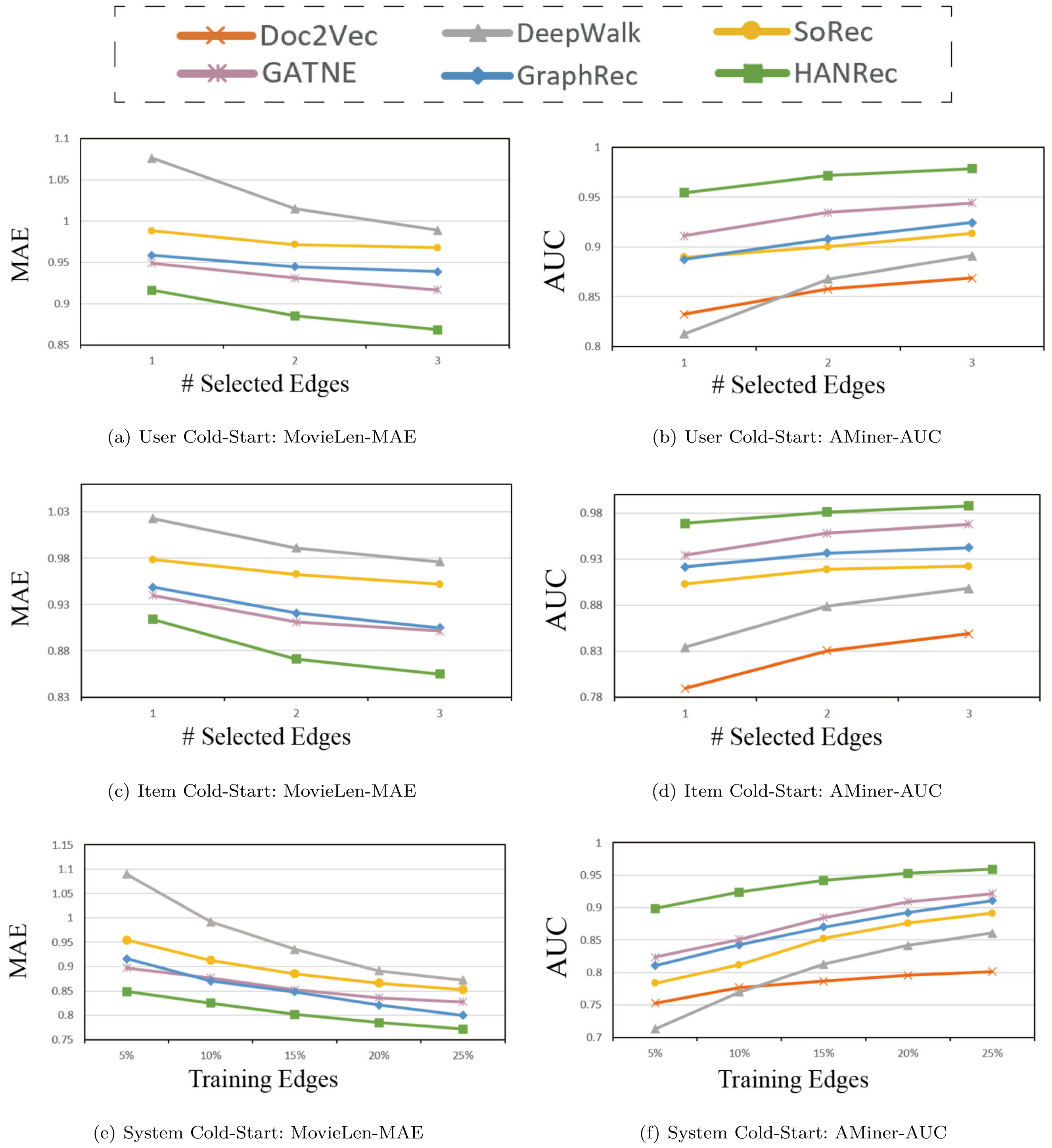


Fig. 4 Cold-start problem analysis on the MovieLens and AMiner datasets

user in the training dataset. For example, for the MovieLens dataset, we randomly select 5% from 610 users, that is, 35 users as new users. Then we use all the information of the other 95% of users (rating scores of the movie and attributes of the corresponding movie's subject matter) for training. Next, for each of these 35 users, we randomly select 1, 2, and 3 movie rating records and add them to the training dataset to simulate their feedback when they first logged in to the system. Finally, we use the other rating score data of these 35 users as the test dataset. As for the AMiner dataset, we use the same preprocessing method. We randomly select 5% from 16,604 authors, that is, 830 authors as new users, and use the rest as the training dataset. Next, for each of these 830 authors, we select 1, 2, and 3 edges connected to it and add them to the training dataset. After this, we use rest edges and negative samples related to these new users as the test dataset. (2) The item cold-start problem mainly solves the problem of how to recommend a new item to users who may be interested in it [49]. Like user cold-start, in the MovieLens dataset, we filter out 5% of 9724, that is, 486 movies, and then select 1, 2, and 3 rating records for each item. Then we add them to the training set and treat the rest as the test dataset. For the AMiner dataset, we select 5% of 12455, that is, 623 papers, and add 1, 2, and 3 edges to the training dataset for each paper. We also use negative sampling to generate the test dataset. (3) And for the system cold-start problem, we define it as how to make high-quality recommendations based on the information of a small number of edges (rating scores or relationships). We continuously adjust the proportion of available information in the two datasets, that is, the proportion of training dataset x, from 5% to 25%, and obtain the performance results of a series of methods.

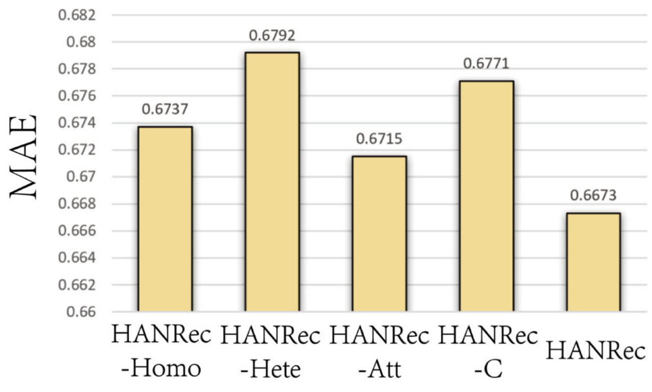
Figure 4 shows the corresponding cold-start analysis on the MovieLens and AMiner datasets. Since the random walk-based methods (DeepWalk and LINE) have similar performance, we only report the results of DeepWalk here. As a GCN-based method, LightGCN only uses the information whether the edge exists and does not use the weight information of the edge (such as a rating score, etc.). It is difficult for LightGCN to give a high-quality recommendation when we need to get a specific rating score. When only a few edges (rating scores or relationships) are given, it is harder for LightGCN to provide effective recommendations. So for graph deep learning methods, we only discuss GATNE and GraphRec here, which can make full use of data information, as comparison algorithms.

As shown in Fig. 4a and b, even if we only add one edge related to each test user (new user) in the training set, HANRec can still show excellent performance. For example, HANRec reaches more than 0.95 for the AUC index in the AMiner dataset, indicating that HANRec can fully describe the characteristics of new users with only a few tags. It is helpful for the system to predict the preferences of new users in the future. For the item cold-start problem, HANRec still achieves the highest performance in various cases. When the ratio of the training edges is from 5% to 25%, both Fig. 4e and f reflect that HANRec performs the best. 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, 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. Whether it is a user/item/system cold-start problem, we can find that the graph-based deep learning method is better than the random walk method and the matrix decomposition method; and in the graph deep learning method, HANRec has achieved the best. It is worth noticing that as shown in Fig. 4f, 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.

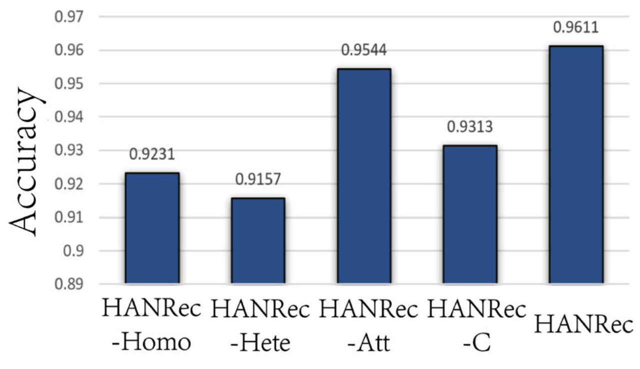
4.7 Ablation study

We evaluate 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 three. 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 component, 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 explore 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 among entities. In this case, formula (1) can be rewritten as: $\boldsymbol{f}_{i,j} = MLP(\boldsymbol{h}_k^{\mathbf{0}} \oplus \boldsymbol{h}_j^{\mathbf{0}})$. Connecting-RB means the relation-based connecting method. Connecting-RB only uses the relationship information among entities

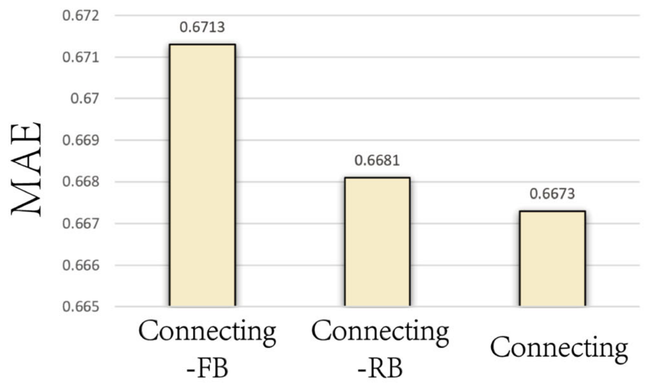


(a) Ablation Study: MovieLen-MAE

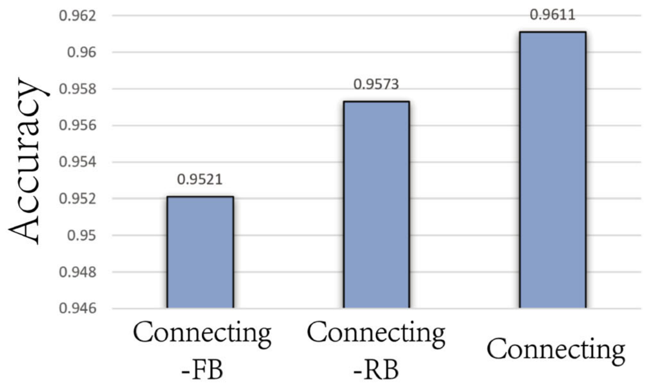


(b) Ablation Study: AMiner-Accuracy

Fig. 5 Effect of each component on the MovieLens and the AMiner dataset

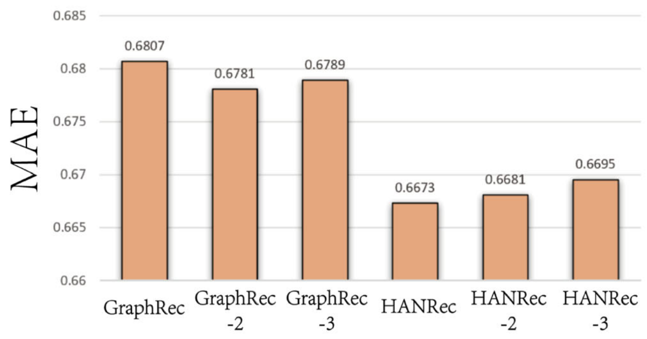


(a) Connecting Strategies: MovieLen-MAE

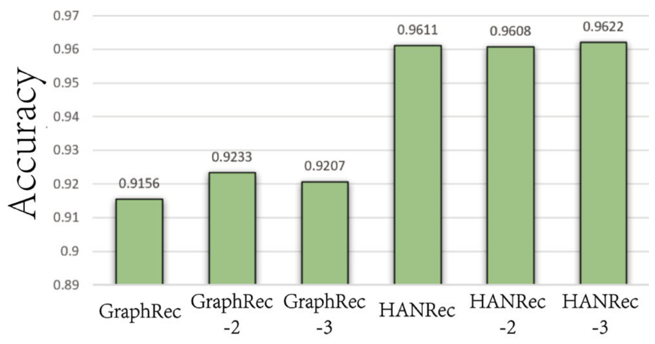


(b) Connecting Strategies: AMiner-Accuracy

Fig. 6 The performance of different connecting methods on the MovieLens and the AMiner dataset

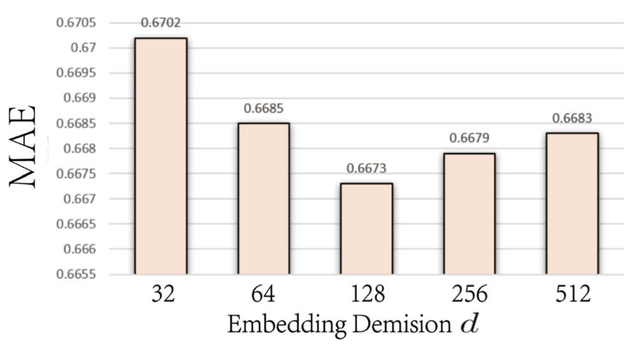


(a) Aggregation Layers: MovieLen-MAE

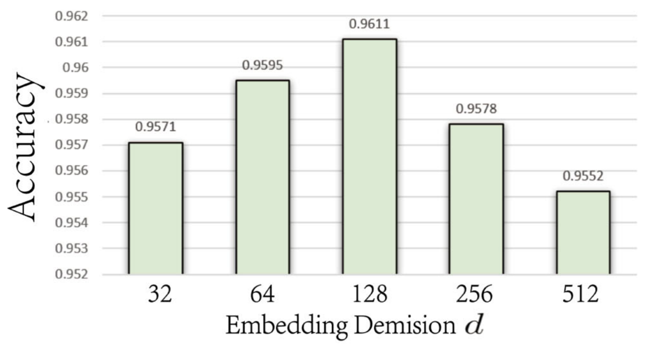


(b) Aggregation Layers: AMiner-Accuracy

Fig. 7 The performance of GraphRec and HANRec with different aggregation layers on the MovieLens and the AMiner dataset. GraphRec, GraphRec-2, and GraphRec-3 contain one, two, and three aggregation layers, respectively, and the same goes for HANRec, HANRec-2, and HANRec-3



(a) Parameter Analysis: MovieLen-MAE



(b) Parameter Analysis: AMiner-Accuracy

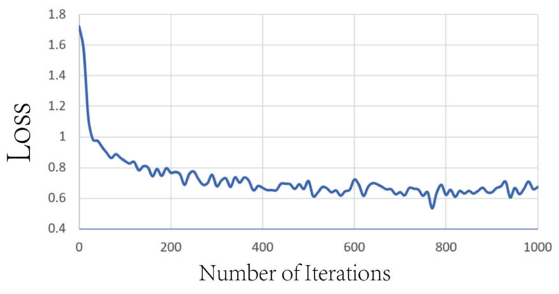
Fig. 8 Effect of dimension d on the MovieLens and the AMiner dataset

on the path, not the feature information of entities. So formula (1) can be rewritten as: $\boldsymbol{f}_{i,j} = MLP(\boldsymbol{e}_{i,k} \oplus \boldsymbol{e}_{k,j})$. 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 among users.

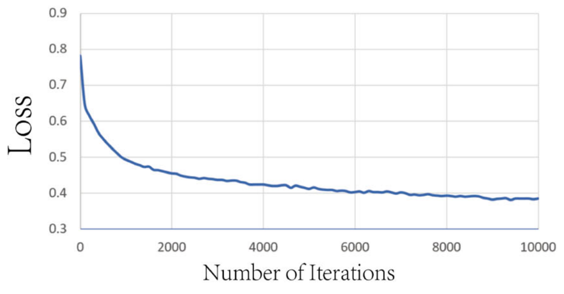
4.8 Parameter analysis

In this part, we discuss the impact of the number of aggregation layers on the performance of GraphRec and HANRec. GraphRec, GraphRec-2, and GraphRec-3 contain one, two, and three aggregation layers, respectively, and the same goes for HANRec, HANRec-2, and HANRec-3. Figure 7 shows that for both GraphRec and HANRec, increasing the number of aggregation layers does not significantly improve the performance. For the MovieLens Lateset Datasets, when the number of aggregation layers of HANRec is increased to three, the performance will decrease to a certain extent, which may be due to too many parameters, and the model is challenging to train adequately. When the aggregation layer of GraphRec is two, the model has learned the influence of entities with potential relationships through implicit aggregation. However, the performance is still weaker than that of HANRec, which shows that in this task, explicit mining of neighbors with potential relationships is better than implicit mining by stacking multiple aggregation layers, which is also in line with our design philosophy. For the AMiner dataset, HANRec-3 is slightly better, while HANRec-2 is a little bit worse than HANRec. We infer that the difference in the random initialization embedding leads to slight fluctuations in the results when the number of aggregation layers is different. But no matter which number of aggregation layers is used, the effect of HANRec is significantly higher than that of GraphRec, which shows the effectiveness of HANRec.

Then, we analyze the effect of embedding dimension d of the entity's latent representation $\boldsymbol{p}_i$ on the performance of HANRec. Figure 8 presents the performance comparison of the embedding dimension on the MovieLens and the AMiner datasets. 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 performance. It demonstrates that using a large number of the embedding dimension has powerful representation. Nevertheless, if the embedding



(a) MovieLen-Loss



(b) AMiner-Loss

Fig. 9 Performance with respect to the number of iterations on the MovieLens and the AMiner dataset

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. 9, we can see that the proposed model has a fast convergence rate, and about 400 iterations are required for the MovieLens dataset, while about 8000 iterations are required for the AMiner dataset.

5 Conclusion

In this paper, we propose a heterogeneous attributed network framework with a connecting method, called HANRec, to address the recommendation task in the 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 datasets can prove that HANRec outperforms state-of-the-art models for the recommendation and link prediction tasks.

Acknowledgements This work is supported in part by the National Key Research and Development Program of China (No.2019YFB2102600), the National Natural Science Foundation of China (No.62002035, No.61902044), the Natural Science Foundation of Chongqing(No.cstc2020jcyj-bshX0034).

References

1. Ricci F, Rokach L, Shapira B (2011) Introduction to recommender systems handbook. In: Recommender systems handbook. Springer, pp 1–35. https://doi.org/10.1007/978-0-387-85820-3_1
2. Wen P, Yuan W, Qin Q, Sang S, Zhang Z (2021) Neural attention model for recommendation based on factorization machines. Appl Intell 51(4):1829–1844. https://doi.org/10.1007/s10489-020-01921-y
3. Ma H, Yang H, Lyu MR, King I (2008) Sorec: social recommendation using probabilistic matrix factorization. In: Proceedings of the 17th ACM conference on information and knowledge management, pp 931–940. https://doi.org/10.1145/1458082.1458205
4. Wu J, Chen L, Yu Q, Han P, Wu Z (2015) Trust-aware media recommendation in heterogeneous social networks. World Wide Web 18(1):139–157. https://doi.org/10.1007/s11280-013-0243-3
5. Fan W, Ma Y, Li Q, He Y, Zhao E, Tang J, Yin D (2019) Graph neural networks for social recommendation. In: The World Wide Web Conference, pp 417–426. https://doi.org/10.1145/3308558.3313488
6. Wang Y, Duan Z, Liao B, Wu F, Zhuang Y (2019) Heterogeneous attributed network embedding with graph convolutional networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol 33, pp 10061–10062. https://doi.org/10.1609/aaai.v33i01.330110061
7. Zhong T, Zhang S, Zhou F, Zhang K, Trajcevski G, Wu J (2020) Hybrid graph convolutional networks with multi-head attention for location recommendation. World Wide Web 23(6):3125–3151. https://doi.org/10.1007/s11280ndash 020–00824–9
8. Holzinger A, Malle B, Saranti A, Pfeifer B (2021) Towards multi-modal causability with graph neural networks enabling information fusion for explainable ai. Inf Fusion 71:28–37
9. Derr T, Ma Y, Tang J (2018) Signed graph convolutional networks. In: 2018 IEEE International Conference on Data Mining (ICDM). IEEE, pp 929–934. https://doi.org/10.1109/ICDM.2018.00113
10. Chen X, Liu D, Xiong Z, Zha Z-J (2020) Learning and fusing multiple user interest representations for micro-video and movie recommendations. IEEE Trans Multimed 23:484–496. https://doi.org/10.1109/TMM.2020.2978618
11. Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, Wang L, Li C, Sun M (2020) Graph neural networks: A review of methods and applications. AI Open 1:57–81. https://doi.org/10.1016/j.aiopen.2021.01.001
12. Ji S, Yang W, Guo S, Chiu DicksonKW, Zhang C, Yuan X (2020) Asymmetric response aggregation heuristics for rating prediction and recommendation. Appl Intell 50(5):1416–1436. https://doi.org/10.1007/s10489-019-01594-2
13. Wu Z, Pan S, Long G, Jiang J, Chang X, 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, pp 753–763. https://doi.org/10.1145/3394486.3403118
14. Mandal S, Maiti A (2020) Explicit feedback meet with implicit feedback in gpmf: a generalized probabilistic matrix factorization model for recommendation. Appl Intell:1–24. https://doi.org/10.1007/s10489-019-01594-2
15. Zhang X, Luo H, Chen B, Guo G (2020) Multi-view visual bayesian personalized ranking for restaurant recommendation. Appl Intell 50(9):2901–2915. https://doi.org/10.1007/s10489-020-01703-6
16. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems, pp 5998–6008. https://doi.org/10.5555/3295222.3295349
17. Tang J, Aggarwal C, Liu H (2016) Recommendations in signed social networks. In: Proceedings of the 25th International Conference on World Wide Web, pp 31–40. https://doi.org/10.1145/2872427.2882971
18. Yang B, Lei Y, Liu J, Li W (2016) Social collaborative filtering by trust. IEEE Trans Pattern Anal Mach Intell 39(8):1633–1647. https://doi.org/10.1109/TPAMI.2016.2605085
19. Pazzani MJ, Billsus D (2007) Content-based recommendation systems. In: The adaptive web. Springer, pp 325–341. https://doi.org/10.1007/978-3-540-72079-9_10
20. Herlocker JL, Konstan JA, Terveen LG, Riedl JT (2004) Evaluating collaborative filtering recommender systems. ACM Trans Inf Syst (TOIS) 22(1):5–53. https://doi.org/10.1145/963770.963772
21. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 42(8):30–37. https://doi.org/10.1109/MC.2009.263
22. Mnih A, Salakhutdinov RR (2007) Probabilistic matrix factorization. Adv Neural Inf Process Syst 20:1257–1264. https://doi.org/10.5555/2981562.2981720
23. Bronstein MM, Bruna J, LeCun Y, Szlam A, Vandergheynst P (2017) Geometric deep learning: going beyond euclidean data. IEEE Signal Proc Mag 34(4):18–42. https://doi.org/10.1109/MSP.2017.2693418
24. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 701–710. https://doi.org/10.1145/2623330.2623732

25. Wang H, Wang N, Yeung D-Y (2015) Collaborative deep learning for recommender systems. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1235–1244. https://doi.org/10.1145/2783258.2783273
26. Kipf TN, Welling M (2017) Semi-Supervised Classification with Graph Convolutional Networks. In: Proceedings of the 5th International Conference on Learning Representations, ICLR '17
27. Xu H, Duan Z, Wang Y, Feng J, Chen R, Zhang Q, Xu Z (2021) Graph partitioning and graph neural network based hierarchical graph matching for graph similarity computation. Neurocomputing 439:348–362. https://doi.org/10.1016/j.neucom.2021.01.068
28. Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17(6):734–749. https://doi.org/10.1109/TKDE.2005.99
29. He R, McAuley J (2016) Fusing similarity models with markov chains for sparse sequential recommendation. In: 2016 IEEE 16th international conference on data mining (ICDM). IEEE, pp 191–200. https://doi.org/10.1145/3383313.3412247
30. Hidasi B, 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, pp 843–852. https://doi.org/10.1145/3269206.3271761
31. Liu Q, Zeng Y, Mokhosi R, 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, pp 1831–1839. https://doi.org/10.1145/3219819.3219950
32. Wu L, Sun P, Fu Y, Hong R, Wang X, 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, pp 235–244. https://doi.org/10.1145/3331184.3331214
33. Wang H, Zhao M, Xie X, Li W, Guo M (2019) Knowledge graph convolutional networks for recommender systems. In: The world wide web conference, pp 3307–3313. https://doi.org/10.1145/3308558.3313417
34. Zhao J, Zhou Z, Guan Z, Zhao W, Ning W, Qiu G, 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, pp 2347–2357. https://doi.org/10.1145/3292500.3330686
35. Cen Y, Zou X, Zhang J, Yang H, Zhou J, Tang J (2019) Representation learning for attributed multiplex heterogeneous network. In: proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, pp 1358–1368. https://doi.org/10.1145/3292500.3330964
36. Gardner MW, Dorling SR (1998) Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmosph Environ 32(14-15):2627–2636. https://doi.org/10.1016/S1352-2310(97)00447-0
37. Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. In: Bengio Y, LeCun Y (eds) 3rd international conference on learning representations, ICLR 2015. Conference Track Proceedings, San Diego
38. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958. https://doi.org/10.5555/2627435.2670313
39. He X, Liao L, Zhang H, Nie L, Hu X, Chua T-S (2017) Neural collaborative filtering. In: Proceedings of the 26th international conference on world wide web, pp 173–182. https://doi.org/10.1145/3038912.3052569
40. Tang J, Zhang J, Yao L, Li J, Zhang L, 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, pp 990–998. https://doi.org/10.1145/1401890.1402008
41. Stallings J, Vance E, Yang J, Vannier MW, Liang J, Pang L, Dai L, Ye I, Wang G (2013) Determining scientific impact using a collaboration index. Proc Natl Acad Sci 110(24):9680–9685. https://doi.org/10.1073/pnas.1220184110
42. Dong Y, Chawla NV, 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, pp 135–144. https://doi.org/10.1145/3097983.3098036
43. Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In: International conference on machine learning. PMLR, pp 1188–1196. https://doi.org/10.5555/3044805.3045025
44. Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q (2015) Line: Large-scale information network embedding. In: Proceedings of the 24th international conference on world wide web, pp 1067–1077. https://doi.org/10.1145/2736277.2741093
45. He X, Deng K, Wang X, Li Y, Zhang Y, Wang M (2020) Lightgcn: Simplifying and powering graph convolution network for recommendation. In: Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, pp 639–648. https://doi.org/10.1145/3397271.3401063
46. Schafer JB, Frankowski D, Herlocker J, Sen S (2007) Collaborative filtering recommender systems. In: The adaptive web. Springer, pp 291–324. https://doi.org/10.1007/978-3-540-72079-9_9
47. Bobadilla J, Ortega F, Hernando A, Bernal J (2012) A collaborative filtering approach to mitigate the new user cold start problem. Knowl-based Syst 26:225–238. https://doi.org/10.1016/j.knosys.2011.07.021
48. Wang J, Huang P, Zhao H, Zhang Z, Zhao B, Lee DL (2018) Billion-scale commodity embedding for e-commerce recommendation in alibaba. In: proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp 839–848
49. Grbovic M, Cheng H (2018) Real-time personalization using embeddings for search ranking at airbnb. In: proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp 311–320

Ziheng Duan received the BEng degree from Zhejiang University, China in 2020. Now he is a research intern at Chongqing University. His research interests lie in the area of machine learning, computational biology, graph representation learning, time series analysis, espeically the interaction of them.

Yueyang Wang received the BEng degree from Nanjing University, Nanjing, China in 2013, and the PhD degree from Zhejiang University, Hangzhou, China in 2019. She is currently a lecturer at the School of Big Data and Software Engineering, Chongqing University. Her research interests include social network analysis and data mining.

Weihao Ye received the BEng degree from Chongqing University, China in 2020. Now he is a graduate student at Carnegie Mellon University. His research interests include graph representation learning and pattern recognition.

Qilin Fan (M'19) is currently a Lecturer in the School of Big Data and Software Engineering, Chongqing University, Chongqing, China. She received the B.E. degree in the College of Software Engineering, Sichuan University, Chengdu, China, in 2011, and the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2017. Her research interests include network optimization, mobile edge computing and caching, network virtualization and machine learning.

Xiuhua Li (S'12, M'19) received the B.S. degree from the Honors School, Harbin Institute of Technology, Harbin, China, in 2011, the M.S. degree from the School of Electronics and Information Engineering, Harbin Institute of Technology, in 2013, and the Ph.D. degree from the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC, Canada, in 2018. He joined Chongqing University through One-Hundred Talents Plan of Chongqing University in 2019. He is currently a tenure-track Assistant Professor with the School of Big Data & Software Engineering, and the Director of the Institute of Intelligent Software and Services Computing associated with Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Education Ministry, China. He is also leading a research team in State Key Laboratory of Power Transmission Equipment and System Security and New Technology, Chongqing University, Chongqing, China. His current research interests are 5G/B5G mobile Internet, mobile edge computing and caching, big data analytics and machine learning.