Journal of Computers Vol. 35 No. 4, August 2024, pp. 59-73
doi: 10.53106/199115992024083504005

Research on Link Prediction Method Based on Information Fusion Graph Embedding

Yuhong Zhao[1], Xiangming Ni[1], Yue Yao[2*], and Peng Mei[1]

[1] School of Digital and Intelligent Industry, Inner Mongolia University of Science and Technology, Baotou 014010, China
[2] School of Urban Safety, Beijing Labor Security Vocational College, Beijing 100029, China
zhaoyuhong35@163.com, 18763358974@163.com, 1533630176@qq.com, 949860060@qq.com

Received 3 May 2023; Revised 1 June 2023; Accepted 1 June 2023

Abstract. To accurately and efficiently capture the topological and attribute information of nodes and apply them to the link prediction task, this paper proposes a Dual Channel Graph Convolution Link Prediction (DC-GCN). DC-GCN constructs a dual channel through the graph convolution network. DC-GCN can learn both topological embeddings and attribute embeddings of nodes; it introduces an attention mechanism to learn the weights of each embedding adaptively and then performs weighted fusion to obtain the final embedding representation of nodes. Finally, the Hadamard distance of nodes is used to construct the link representation between nodes, and the probability of linking between nodes is obtained by training a logistic regression function. By comparing and analyzing many different types of link prediction algorithms, the results show that this algorithm has greater advantages in both AUC and Precision evaluation metrics, so DC-GCN can effectively combine node attributes and structural information of the network to improve the accuracy of the link prediction algorithm.

Keywords: link prediction, graph embedding, graph convolutional network, attention mechanism, topological embedding, attribute embedding

1 Introduction

In network theory research, a complex network, as an abstract model of the real world, is a network composed of many nodes and complex relationships among nodes, and the nodes in the network represent individuals in the real world, and the connected edges in the network represent interactions among individuals. In recent years, link prediction (link prediction) [1], as one of the important hotspots in complex network research, has attracted the attention of scholars to explore and has shown great application value in recommendation systems, network reconstruction, and network evolution model evaluation.

Link prediction refers to inferring the possible (invisible) links in the network or the possible future links formed in the network based on the known information of network nodes and network topology information, i.e., the restoration of missing information of the network and the prediction of future information of the network. Nowadays link prediction is widely used in real life, such as active ingredient and mechanism of action prediction in the biological network [2], protein function prediction in protein networks [3], friend recommendation in the social network [4], etc.

Because of the rapid development of computer technology, deep learning has become a hot topic in the field of computer science, image processing, link prediction, and other fields, so deep learning has gradually become a major research hotspot in the field of link prediction. Among them, graph embedding [5] can convert graph data into a dense vector representation in low-dimensional space, while ensuring that certain characteristics of graph data still correspond in vector space, and in addition, the representation of graph data can contain a large amount of semantic information, thus providing better input characteristics for link prediction tasks. Most of the more popular graph embedding-based link prediction methods currently consider only the topological information of nodes, while ignoring the rich attribute information of nodes. In a network graph, topological information explicitly describes the pairwise and sequential relationships between nodes, while node attributes provide more fine-grained features of the nodes themselves, and both topology and node attributes are key to understanding the network formalism.

* Corresponding Author

Therefore, this paper proposes a GCN two-channel graph-embedded link prediction method (DC-GCN) that adaptively fuses network topology information and attribute information, and the contribution of this method is described as follows.

1. A network feature map generation method is proposed. Firstly, the attribute similarity of any two nodes is calculated, and the first k nodes with high attribute similarity of each node are selected and set as having edges, i.e., the attribute similarity of nodes is treated as having connected edges between nodes, and the k-nearest neighbor matrix of nodes is constructed to generate the network feature map, i.e., the attribute information of nodes is used for GCN convolution in the form of the network feature map.
2. A two-channel method for obtaining network embeddings and attribute embeddings is proposed. A two-channel model is constructed using GCN to obtain the topological embedding and attribute embedding of nodes in complex networks separately. It can perform convolutional operations on the network topology map and the feature map separately on the GCN dual channel, i.e., to maximize the information retention of complex networks and to improve the performance of prediction.
3. A method is proposed for adaptive weighting and fusing topological embeddings and attribute embeddings using an attention mechanism. The optimal weights can be obtained based on the training data without manually assigning the relevant weights, and the information of the nodes can be fully fused.
4. Validating the effectiveness of the method. Extensive experiments on three benchmark datasets, Ciseteer, Cora, and DBLP (Subgraph), show that DC-GCN has a large improvement in both AUC and Precision compared with a series of more representative link prediction methods, and can well extract the most relevant information from node attributes and topology to improve the accuracy of the link prediction task.

2 Introduction of Related Knowledge

2.1 Figure Embedding

Graph embedding (graph representation learning) [5] Iterative learning via neural networks provides an efficient way to represent nodes as low dimensional embeddings to solve complex network analysis and research problems. Specifically, based on the sparsity and high dimensionality of the network, graph embedding algorithms can transform the graph structure into low-dimensional vectors, which can preserve the data information of the initial graph and increase the accuracy of the link prediction task by using the embedding results for other tasks, such as link prediction and image classification [6], network recommendation [7], etc.

2.2 Graph Convolutional Network (GCN)

Bruna et al. [8] first proposed Graph Convolution Networks (GCN) in 2013 based on Spectral Graph Theory (SGT). GCN is the aggregation of neighborhood information for computation, which has the power to model dependencies between graph data and to feature extraction and representation of graph data. Previous CNNs (Convolution Neural Networks) deal with Euclidean structured data, which consists of pixel points, while GCNs deal with non-Euclidean structured data, which consists of nodes and edges, such as citation networks, social networks, etc. A typical GCN [10] and its variants [24-30] usually use a matrix form to iteratively calculate the features of each node, and then perform convolution operations by layer propagation, and finally, by updating the node features, the node embedding can be applied to data mining fields, such as link prediction.

2.3 Attention Mechanism

Attention mechanism [9] is widely applied in most fields of computers, including machine translation, natural language processing, image recognition, and other fields. For example, in the field of machine translation, scholars have found that context words have different influences on the target words, and introducing Attention Model (AM) can get information about the weights of different words relative to the target words to increase the accuracy of the translation.

With the introduction of graph neural networks, many scholars have combined attention mechanisms with graph neural networks and applied them to various domains [11], all of which have performed well relative to

traditional algorithms. The attention mechanism allows a neural network to center at a subset of features of its input, i.e., to focus on a specific input. There are various forms of combining attention mechanisms and graph neural networks, which include attention weight assignment to different neighbors to complete feature information aggregation, weighted aggregation of multiple models based on attention weights, using attention weights to guide random wandering, etc.

2.4 Introduction to Classical Methods for Link Prediction

Link prediction Method Based on the Similarity Index. Traditional link prediction methods usually use the similarity between nodes as a metric to measure the likelihood of contiguous edges between node pairs <Vx, Vy> by measuring the similarity score S<Vx, Vy>, and the higher the similarity score, the higher the likelihood of contiguous edges. The related formula is described as follows.

(1) Method based on the local similarity index

CN (common neighbors) [12]: A first-order heuristic that calculates only the first-hop neighbor information of two target nodes. The number of common neighbors of the target nodes is calculated to determine the probability of generating concatenated edges, i.e., the more the number of common neighbors between the target nodes, the higher the possibility of concatenated edges exists.

AA (Adamic-Adar) [13]: AA belongs to the second-order heuristic, which calculates the information about the common neighbors of two target nodes at most two hops. In this algorithm, the smaller the degree the greater the contribution of the common neighbor nodes. For example, in a social network, two people who jointly like the same cold novel are more likely to create a connection between them than two people who jointly follow a more popular novel. Therefore, a weight value can be assigned to each node based on the degree of the common neighbor nodes.

(2) Method based on the path similarity index

LP (Local Path) [14]: Similar to the local similarity index, and based on this optimization is done, considering the second-order neighbors and at the same time considering the influence of the third-order neighbors on the target node, and controlling the third-order path by adding an adjustable parameter α.

Katz [15]: By calculating all paths between two nodes, weights are assigned to paths between two nodes according to the principle that the shorter the path, the greater the weight. Katz metric uses a weight decay factor β to calculate the similarity between two nodes.

(3) Method based on random wandering similarity

LRW (Local Random Walk) [16]: Local Random Walk indicator, which can limit the length of the motion path and reduce the amount of computation, and is more suitable for large networks with sparse nodes.

Link Prediction Method Based on Graph Embedding. In addition to link prediction research using similarity, methods of network representation have recently been increasingly applied to link prediction research, bringing a breakthrough in the development of link prediction [17]. Link prediction methods based on network representations can be divided into 3 categories: matrix decomposition-based methods, random wander-based methods, and deep learning-based methods.

(1) Matrix decomposition-based methods

At the early stage of research, matrix decomposition-based methods are popular studies in graph embedding. These algorithms can take into account all information of the network graph, decompose the information of the network graph into the context matrix and the embedding vector matrix, and then train accordingly to obtain the final embedding vector through the loss function. The matrix decomposition-based network embedding technique is mainly used to reduce matrix dimensions by linear and nonlinear transformations.

SVD (Singular Value Decomposition) [17] is a more efficient matrix decomposition method in machine learning. First, a transfer matrix of order 1-k is constructed, and then the singular value matrix decomposition is performed on it separately. This method can decompose a matrix of arbitrary form into the form of the orthogonal matrix, diagonal matrix, and product of orthogonal matrices, and finally, splice the obtained low-dimensional embedding vectors of multiple nodes to derive the final embedding vectors of nodes. However, the application of SVD makes the computational complexity extremely high, and this method cannot be applied to large-scale networks.

SPC [18] combines spectral clustering with unsupervised spectral clustering for similarity transformation, so that prior information can be extended to space level constraints. Research has found that these spatial level con-

strained (undirected) graphs are more meaningful and have better clustering results.

(2) Method based on random wandering

The methods based on the random walk utilizes a sequence of node random walks to calculate the similarity between nodes, if the vectorization between nodes is higher, the probability of generating edges between them is higher. Among them, DeepWalk and Node2vec are the classical algorithms of random walk algorithms.

DeepWalk [19] is a new social network embedding algorithm that uses the neural language model Skip-Gram to implement graph embedding, treating graph embedding as a natural language problem. The random wandering paths of nodes are obtained by mimicking the process of text generation to obtain a sequence of nodes, where the paths are equivalent to sentences in the text and the nodes are equivalent to words in the text. The objective function of DeepWalk is as follows:

$$\mathrm{L}(s)=\frac{1}{|s|}\sum_{x=1}^{|s|}\sum_{x-t\le y\le y+t,y\ne i}\log Pr(v_y\mid v_x). \tag{1}$$

Among them.

$$Pr(v_y\mid v_x)=\frac{\exp(v_{y'}\mid v_x)}{\sum_{v\in V}\exp(v'\cdot v_x)}. \tag{2}$$

The essence of Node2vec [20] is to embed the text, which is based on the DeepWalk algorithm and changes the way the random walk sequence is generated, in which the nodes are traversed using random walks with unequal probability (using a combination of depth-first and breadth-first algorithms). Its objective function is as follows:

$$\max_{\mathrm{f}}\sum_{u\in V}\log P(Ns(u)\mid f(u)). \tag{3}$$

Among them,

$$P(Ns(u)\mid f(u))=\sum_{n_x\in Ns(u)}\mathrm{P}(n_y\mid f(u)). \tag{4}$$

V is the set of nodes and f is a mapping function.

(3) Deep learning-based approach

Rapid development of deep learning, and as a result, researchers have designed various deep learning-based methods for embedding graph neural network nodes.

LINE (Large-scale Network Information Embedding) [21] defines two loss functions O_1 and O_2, which contain first-order proximity relations and second-order proximity relations, based on which the first-order and second-order objectives are then optimized. Although this method has a high learning efficiency, it does not show good performance when dealing with remote nodes. Its objective function is as follows:

$$O_1=-\sum_{(x,y)\in E}W_{x,y}\log P_1(v_x,v_y). \tag{5}$$

$$O_2=\sum_{x\in V}\lambda_x d(\hat{P}_2(\cdot\mid v_x),P_2(\cdot\mid v_x)). \tag{6}$$

LINE is a shallow model that often does not capture the nonlinear network structure well, giving rise to both the SDNE algorithms. SDNE is the most classical algorithm for deep self-encoder, which is widely used to downscale nonlinear data and learn the information of nonlinear data. SDNE algorithm takes the adjacency matrix of the network as input and learns the node information of each row and each column, where the algorithm preserves the nodes' first-order information by Laplace matrix and the nodes' second-order information by deep self-encoder, and finally outputs the nodes' embedding vector.

In addition to these methods of deep self-encoders, the remaining methods are based on graph neural networks, which are a combination of encoders and decoders. This type of method mainly aggregates the information of the node's neighbors and then updates the information of the current node, which results in the embedding vector representation of the node through multiple iterations. One of the most typical algorithms is the graph convolutional network GCN, which functions similarly to other convolutional neural networks as a feature extractor, but targets the graph data [22]. In general, a graph structure is very irregular, infinite-dimensional data, and therefore does not have translation invariance. The surrounding domain of each node in the graph structure may be different, and such a data structure leads to the uselessness of traditional CNN and RNN. GCN proposes a new method to extract features from graph data in a subtle way and use these features for downstream tasks, which can be used for a wide range of purposes [23].

Although there are currently many network-based link prediction algorithms, most of them only utilize the structural characteristics of the network itself and ignore the rich attribute information contained in the network. Due to the crucial role of link structural features in link prediction, utilizing attribute features can effectively improve link prediction accuracy. Therefore, we propose a GCN dual channel graph embedding method for link prediction.

3 DC-GCN Model

3.1 Problem Description

Define a complex network as $G = (V, E)$, where V denotes the set of nodes in the network and E denotes the set of connected edges between nodes. $A \in R^{n\times n}$ denotes the adjacency matrix of nodes in the network, and $A_{ij} = 1$ when there is a link between node i and node j, otherwise $A_{ij} = 0$. $X \in R^{n\times d}$ is the node feature matrix and d is the dimension of the node features. When a network has N nodes, then the network has at most $N(N\text{-}1)/2$ links, denoted as the full set U.

For any network, a link prediction method is designed to compute a similarity score value S_{ij} for any node pair $(i, j) \in U\text{-}E$, and to rank the node pairs according to this score value from the largest to the smallest, then, the higher the ranked node pair is the more likely to have a connection relationship between them. In the static link prediction shown in Fig. 1, the first half of the molecular map represents the sample network used for the study, which is also called the observation network, and the second half of the molecular map is derived by partitioning the data set, which is called the test network. Where the solid line represents the training set and the dashed line represents the test set, i.e., the prediction of whether unknown node pairs establish links based on the existing linking relationships between nodes.

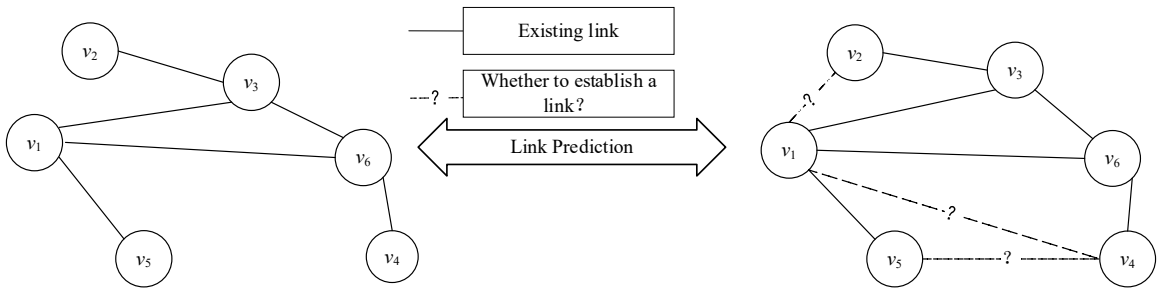

Fig. 1. Link prediction

3.2 DC-GCN Basic Ideas

To more comprehensively mine the network information, yet significantly enhance the ability to fuse topology and node attributes, a two-channel graph convolutional link prediction method (DC-GCN) is proposed in this paper. The model diagram of DC-GCN is shown in Fig. 2.

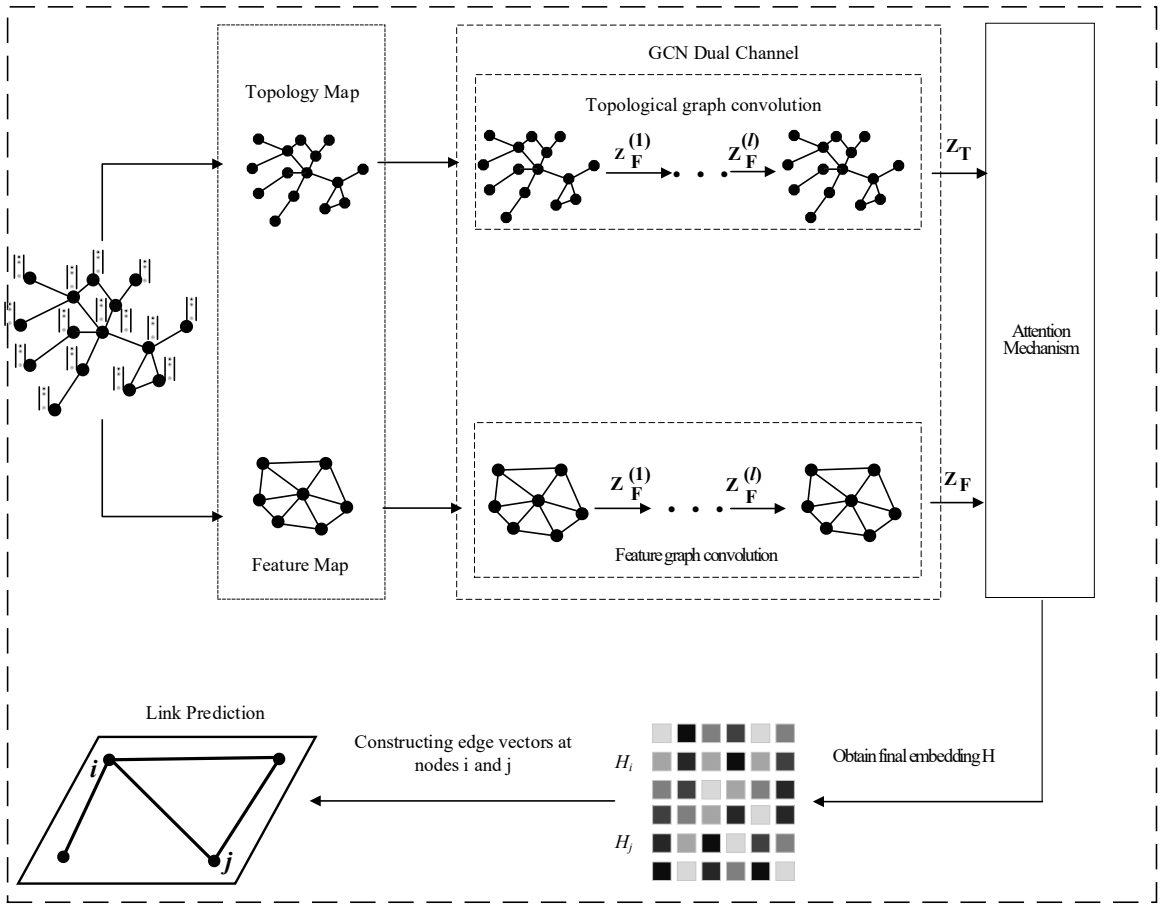

Fig. 2. GCN flow chart

DC-GCN first performs dataset processing to partition the information network graph into topological and attribute graphs. Then, using the feature that GCN can extract graph data to generate embedding, the topological embedding and attribute embedding of nodes are obtained by double-channel convolutional operations on topological and attribute graphs of nodes respectively. In the topological space, GCN is a multilayer graph convolutional neural network, and the input of each layer of GCN is the adjacency matrix A and the feature matrix X of nodes, and each convolutional layer is responsible for processing first-order neighborhood information, and the superposition of multiple convolutional layers can achieve the transmission of information from multiple neighboring regions, obtaining information from the neighbor summary of each node; similarly, the same is true in the feature space, except that the input of the feature space is the k-nearest neighbor matrix A_f and the feature matrix X of the node. After that, the attention mechanism model is introduced, and the attribute embedding and topological embedding of the nodes are weighted and fused to finally obtain the relational features of the nodes for link prediction. DC- GCN solves the problem that traditional GCN can only obtain information from node features or topology, but cannot fuse them adaptively, and improves the accuracy of link prediction.

3.3 DC-GCN Algorithm

Embedding Acquisition. DC-GCN constructs a dual channel of GCN based on topological space and feature space, where topological space convolution is used to learn the topological embedding of nodes H_F, and feature space convolution is used to learn the attribute embedding of nodes H_T.

(1) Topology embedding acquisition

In the topological space, the original input graph $G=(A, X)$ is fed into the GCN for convolution operation, and the output of the *lth* layer is shown in Equation (7).

$$H_T^l = \mathrm{ReLU}(\tilde{D}^{-\frac{1}{2}}\tilde{A}\ \tilde{D}^{-\frac{1}{2}}H^{l-1}W^l), \tag{7}$$

where ReLU is the activation function, the initial $H^0 = X$, the $\widetilde{D}$ is the diagonal matrix of A, also called the degree matrix, which represents the number of neighbors of each node, and W^l denotes the weight matrix of the *lth layer* in the GCN convolutional layer. Since the adjacency matrix A only obtains information about the neighbors of a node and ignores information about the node itself, the value of the diagonal in the adjacency matrix A is set to 1 to increase the self-connection relationship of each node, i.e. $\widetilde{A} = A+I$, I is the unit matrix. The aggregation operation and node feature update are the core operations of the graph convolutional network, as shown in Equation (7), where the characteristics of nodes are updated by continuously exchanging information about their neighbors until a stable equilibrium is reached, at which time the feature vectors of all nodes contain information about their neighbor nodes, which can then be used to obtain the final layer of output embedding, denoted as H_T.

(2) Attribute embedding acquisition

The input of GCN is generally the starting feature vector X and the adjacency matrix A that represents the connectivity between nodes. To make full use of the information in the feature space, first calculate the attribute similarity value S_{ij} for node pair (i,j), select the nodes with the top k most similar attributes to node i and set them as connected edges, generate a new adjacency matrix A_f by setting k nearest neighbor edges for each node, *and* obtain the k nearest neighbor graph $G_f =(A_f, X)$ as the feature structure graph. In the topological space, the adjacency matrix A in the original graph $G = (A, X)$ considers all the neighbors of the node and focuses on learning the topological information of the node; while in the feature space, the k-nearest neighbor matrix A_f in the feature structure graph $G_f = (A_f, X)$ focuses only on the first k nodes with similar attributes of the node and focuses on learning the attribute information of the node.

In the feature space, the attribute similarity matrix of the n nodes is first calculated using the cosine similarity formula as shown in Equation (8).

$$S_{ij} = \frac{x_i \cdot x_j}{|x_i| \cdot |x_j|}, \tag{8}$$

where $x_i, x_j \in R^{1\times d}$ are the *i-th* and *j-th* rows of the feature matrix X, representing the feature matrices of node i and node j.

Then, the first K nodes with similar attributes of each node are selected to set the edges according to the similarity matrix $S \in R^{n\times n}$, for example, if the first three nodes with high attribute values in node 1 are node 3, node 5, and node 6, then set the value between node 1 and the other three nodes to 1, and set the value between node 1 and the other nodes to 0. The same applies to other nodes. Finally, the modified N × N dimensional adjacency matrix A_f is obtained, and the structural characteristic graph is $G_f = (A_f, X)$.

As shown in Fig. 3, the similarity values of node V_3 and other nodes S_{V3Vi} , and S_{V3Vi} are calculated separately, and when k=2, the two nodes V_2 and V_8 with the largest similarity values to V_3 are selected as connected edges. Similarly, when k=m, the m nodes with the greatest similarity to V_3 are selected and set as connected edges, i.e., the k-nearest neighbor graph of the nodes is obtained.

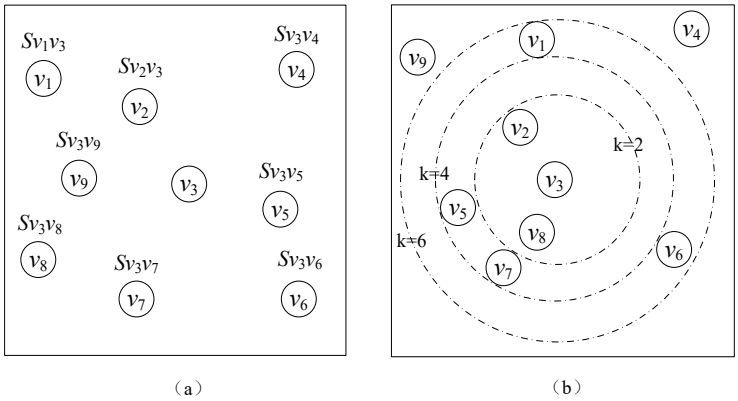

Fig. 3. k-nearest neighbor graph construction (with node V3 as an example)

Finally, the feature input map $G_f = (A_f, X)$ is input to the GCN for convolution operation, and the output of the *lth* layer is shown in Equation (9).

$$H_T^l = \mathrm{ReLU}(\tilde{D}_f^{-\frac{1}{2}} \tilde{A}_f \tilde{D}_f^{-\frac{1}{2}} H^{l-1} W^l), \tag{9}$$

where the initial $H_T^0 = X$. Specifically, there is $A_f = A_f + I_f$, I_f is the unit matrix, and the last level of output embedding is denoted as H_F.

Since then, the topological embedding H_T and the attribute embedding H_F of the node are obtained.

Embedding Fusion. The attention mechanism allows neural networks to focus on a subset of their input features, now in this paper, there are two specific embeddings H_T and H_F that use the attention mechanism α_{tt} (H_T, H_F) to learn their corresponding importance (α_t, α_f) as shown in equation (10).

$$(\alpha_t, \alpha_t) = \alpha_{tt}(H_T, H_F). \tag{10}$$

Where α_t, $\alpha_f \in \mathrm{R}^{n\times 1}$, represent the attention values of the topological embedding H_T and feature embedding H_F of the node, respectively.

For node *i*, it's embedding in H_T $H_T^i \in R^{1\times d}$, representing row *i* of H_T. This embedding uses a nonlinear transformation followed by a shared attention vector $q \in \mathrm{R}^{h'\times 1}$ to obtain the attention value *w*, as shown in Equation (11).

$$W_T^i = q^T \cdot \tanh\left(W \cdot (H_T^i)^T + b\right), \tag{11}$$

where $W \in R^{h'\times h}$ is the weight matrix and $b \in \mathrm{R}^{h'\times 1}$ is the bias vector. Similarly, the attention values of node *i* in the embedding matrix H_F can be obtained separately w_F^i .

The attention values w_T^i , w_F^i are normalized using the softmax function to obtain the final weights as shown in Equation (12).

$$\alpha_T^i = \mathrm{softmax}(w_T^i) = \frac{\exp(\mathrm{W}_T^i)}{\exp(\mathrm{W}_T^i) + \exp(\mathrm{W}_F^i)}, \tag{12}$$

where the larger α *is*, the more important the corresponding embedding is. Similarly, $\alpha_F^i = \text{softmax}\,(w_F^i)$ is obtained.

For each of the n nodes, their weights are learned, $\alpha_t = \left[\alpha_T^i\right]$ $\alpha_f = \left[\alpha_F^i\right] \in R^{n\times 1}$, and then the weight information of the n nodes is aggregated within the same matrix using the diag function $\alpha_T = \text{diag}(\alpha_t)$ $\alpha_F = \text{diag}(\alpha_f)$.

Finally, the topology embedding and attribute embedding are combined according to the weight coefficient to obtain the final embedding H, as shown in Equation (13).

$$H = \alpha_T \cdot H_T + \alpha_F \cdot H_F. \tag{13}$$

After learning of the attention model, the final embedding H of the nodes, i.e., the relational feature embedding of all target type nodes, is obtained.

Constructing Edge Vectors for Link Prediction. After learning the relational embedding of all nodes, any two nodes i and j are randomly selected for link prediction by constructing connected edge representations. Their Hadamard distances are first computed as their edge vectors, and then the probability of edge existence is evaluated by training a logistic regression function. The edge vectors are defined as shown in equation (14).

$$e_{ij} = H_i \circ H_j. \tag{14}$$

Assume that the probability of having a contiguous edge between node *i* and node *j* is P_{ij}, calculated as shown in Equation (15).

$$P_{ij} = \frac{1}{1+\exp(-e_{ij}^T \phi)}, \tag{15}$$

where φ is a parameter vector with the same dimension as *H*. $e_{ij}^T \varphi$ is the dot product between the vector e_{ij}^T and the vector φ.

After obtaining the probability of connecting edges between two nodes, its loss is calculated by a logistic regression function with the formula shown in (16).

$$L = -\frac{1}{N}\sum_{1}^{N} y_{ij} \log P_{ij} + (1-y_{ij})(1-P_{ij}), \tag{16}$$

where y_{ij} indicates whether there is a connection between node *i* and node *j*. If there is, $y_{ij} = 1$, otherwise $y_{ij} = 0$. *N* is the total number of samples, and *1/N* indicates that the mean value is taken.

4 Experiments and Analysis of Results

In this paper, three datasets with different node information are selected for extensive experiments, different embedding representations of nodes are constructed by the DC-GCN model for two-channel learning, and weighted learning of target type nodes is performed by using GCN combined with attention mechanism, to capture the feature vector of the relationship between them, and finally, link prediction is completed by using logistic regression function. To prove the performance of the model, DC-GCN compares with the similarity-based traditional link prediction methods LP, Katz, LRW, and the neural network-based link prediction methods Node2vec, LINE, GCN, KNN-GCN to prove the superiority of DC-GCN.

4.1 Data Set

To verify the validity of DC-GCN, experiments were conducted on three real data sets in this paper.

Citeseer [25]: This is a paper citation network. It consists of 3327 nodes and 4732 edges, where nodes and edges represent publication and citation relationships, and node attributes represent keywords in the paper.

Cora [26]: This is the well-known citation network. Represent machine learning papers as nodes, with citation relationships between papers as edges, and keywords in the paper representing node attributes.

DBLP (Subgraph) [27]: It represent machine learning papers as nodes, with citation relationships between papers as edges, and keywords in the paper representing node attributes.

The parameters of the three data sets are shown in Table 1.

Table 1. Parameters of networks datasets

Dataset	Number of nodes	Number of consecutive sides	Number of characteristics
Citeseer	3327	4732	3703
Cora	2708	5249	1433
DBLP (Subgraph)	18448	45611	5959

4.2 Evaluation Indicators

The AUC metric is a general link prediction method that provides a comprehensive evaluation of the accuracy of link prediction, then the formula for calculating AUC is defined as shown in Equation (17).

$$\mathrm{AUC} = \frac{\mathrm{n}' + 0.5n''}{n}. \tag{17}$$

The Precision index does not evaluate the accuracy of the algorithm from a macroscopic point of view like AUC, but only considers whether the top L edges are predicted accurately. Training the link prediction algorithm to capture the similarity values between two nodes, remove the edges in the training set, rank only the similarity values of the edges in the test set and the set of nonexistent edges, and take the top L (here L is 100) after the ranking. Assuming that N out of L belongs to the test set, the Precision algorithm is derived as shown in Equation (18).

$$\mathrm{Precision} = \frac{N}{L}, \tag{18}$$

where N denotes the number of test sets in L that belong to the test set.

4.3 Experimental Results and Analysis

This article randomly divides article randomly divides into a training set and a test set in the ratio of 9:1, and then 100 independent experiments are conducted, and for each independent experiment, the AUC and Precision values are obtained, and finally, all the obtained AUC and Precision values are averaged as the final experimental results. For each independent experiment, 10,000 random samples are taken to compare the scores of predicted edges and nonexistent edges to calculate AUC and Precision.

Parameter Analysis. In this model, two-layer GCNs with identical hidden layer dimensions and identical output dimensions are trained simultaneously, with a hidden layer dimension of 512 and an output dimension of 128. For the DC-GCN model, this paper sets the random seed to 123, the learning rate to 0.005, the regularization parameter to 0.001, the dropout to 0.5, and 100 iterations. Under the same environment, 5 experiments were conducted on all studies, and the experimental results were analyzed on an average basis. In this paper, we evaluated this model using both AUC and Precision methods. The parameters in the DC-GCN model are analyzed below, including k values, epoch values, and parameters α_t and α_f.

(1) k-epoch analysis

To check the number of iterations (epoch) of the experiment and the effect of different k values in the k-nearest neighbor graph on the AUC, this paper investigates the effect of k chosen from 2 to 9, respectively, with 100 iterations, on the AUC values on the three datasets. As shown in Fig. 4, in general, regardless of k chosen any one of 2-9, for the three datasets of Citesser, Cora, and DBLP, the value of AUC changes roughly as the number of iterations increases and then levels off or decreases slightly. This is because the number of epochs is different for different models and different datasets for training. In this paper, we first use the training set to train the model and then evaluate the performance of the model on the validation set. As the epoch increases, the effectiveness of the model will be better improved, but if the number of trainings is too large, it will cause overfitting of the model training data and the effect of the validation set tends to be flat or decreasing. Ideally, the inflection points at which the model changes from good to bad needs to be found. From Fig. 4, it is observed that most of the folds reach the maximum value of AUC at the number of iterations equal to 90. Although there are a few cases where the value of AUC does not result in the maximum at the number of iterations of 90, this experiment selects the number of iterations of 90 by most principles and considers that the results reach the optimum at this time.

To examine the effect of the selection of row k values in the kNN graph on the performance of DC-GCN, this paper investigates the performance of DC-GCN with k values in the range of 2-9. As shown in Fig. 5, for three datasets with 90 iterations, the accuracy first increases and then starts to decrease as the k value increases. This may be because if the graph becomes denser, the features are more likely to be smoothed, and the larger k is, the more noisy edges may be introduced. When the number of iterations is 90, this paper finds that the AUC value peaks at k=5 on the dataset Citeseer by comparing and analyzing the change of AUC when different k values are chosen for the Citeseer, Cora and DBLP datasets, respectively. Therefore, for the Citeseer dataset, this paper selects epoch=90 and k=5 and considers that DC-GCN has the best prediction effect on the Citeseer dataset at this time. Similarly, for the Cora dataset, epoch=90 and k=5 are also chosen, while for the DBLP dataset, epoch=90 and k=4 are chosen. In this paper, the optimal number of iterations is determined by comparing the changes of AUC values, the optimal k value is obtained by the control variables method, and finally, the best model is used for experiments on the test set.

(2) Note the value distribution

In this paper, the proportion of topological and attribute information of the node that plays a role in the experimental results are measured by the attention values, which are represented by the parameters αt, αf where αt denotes the attention value of topological information of the node and αf denotes the attention value of attribute information of the node. When the number of iterations is 90 and k is chosen as 5 for the Citeseer and Cora datasets and 4 for DBLP, the attention mechanism adaptively generates the optimal attention values, i.e., the weights of attributes and topology, by learning. In Fig. 6, it can be seen that for the dataset Citesser, the attention value accounted for by the topological information of the nodes is 0.61 and the attention value of the attribute information is 0.39, which means that the topological information of the nodes plays a greater role than the attribute information of the nodes on the Citeseer dataset. Of course, both the topological information of nodes and the attribute information of nodes have a corresponding role in the results and cannot be ignored. Similarly, in the data set Cora, the attention values of node topology information and attribute information are 0.78 and 0.22, respectively, and the topology information plays a greater role. In contrast, for the DBLP dataset, the attention values of these two are 0.47 and 0.53, respectively, and the attention value of attribute information is slightly higher. Therefore, it can be inferred that in the link prediction task, topological information in both datasets Cora and Citeseer contribute more to the analysis and study of the network, while in the dataset DBLP, the contribution of attribute information is greater than that of topological information.

To verify the inference of this paper, the k-nearest neighbor graph $G_f = (A_f, X)$ generated from the node attribute matrix *X is* used as the input graph of GCN in Table 2 instead of the traditional network graph and represented as KNN-GCN, when KNN-GCN performs link prediction based on the node attribute information only. It can be seen that GCN outperforms KNN-GCN on the datasets Cora and Citeseer, and the attention values of topological embedding in Fig. 6 are also greater than those of attribute embedding. On the contrary, for DBLP it can be found that GCN outperforms KNN-GCN and the attention value of attribute embedding is also larger than that of topological embedding. In summary, the above inference is proven to be correct.

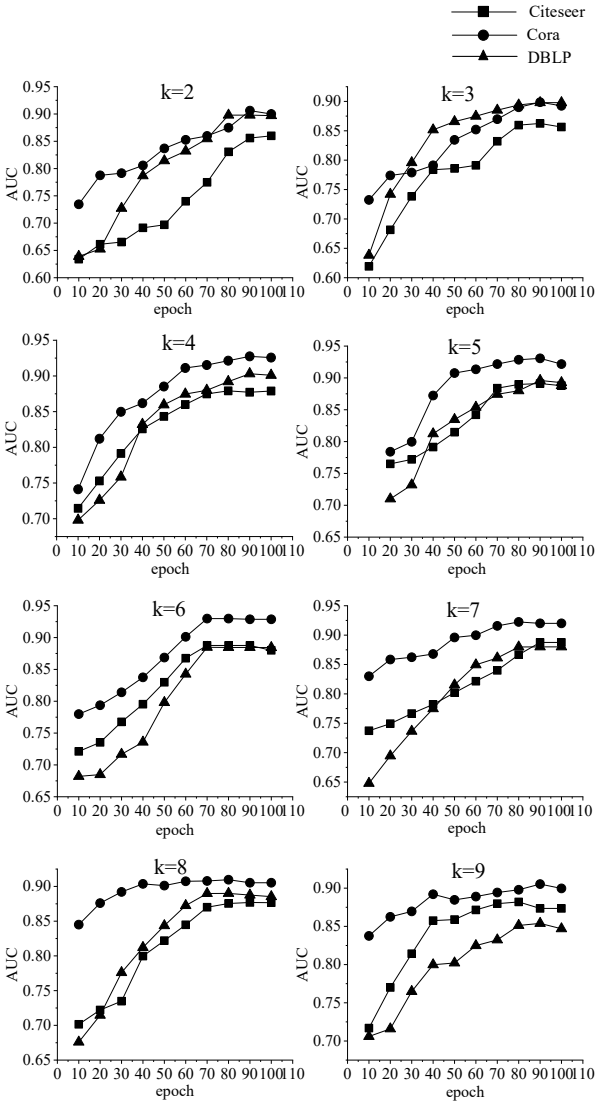

Fig. 4. k-epoch analysis

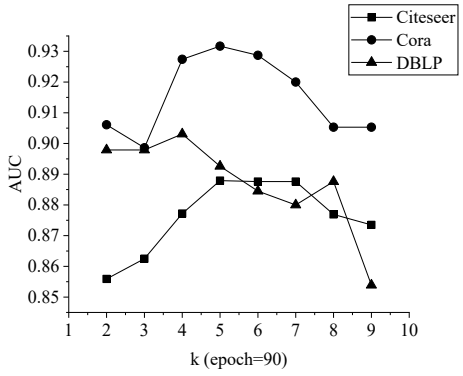

Fig. 5. k analysis

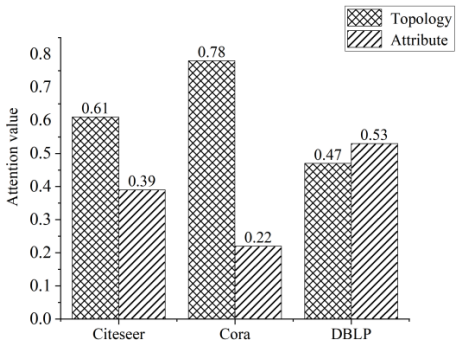

Fig. 6. Attention value distribution

Experimental Results. This paper uses two metrics, AUC and Precision, to measure the effectiveness of DC-GCN, and demonstrate the superiority of the proposed DC-GCN method by comparing it with the similarity-based traditional link prediction methods LP, Katz, LRW, and the graph embedding-based link prediction methods Node2vec, LINE, GCN, KNN-GCN, etc. Their comparison results are shown in Table 2 and Table 3.

The model DC-GCN proposed in this paper experimented on three real datasets, and the AUC values of DC-GCN reached over 89% on all three datasets, which is a large improvement compared with baseline LP, Katz, LRW, Node2vec, LINE, GCN, and KNN-GCN, indicating that DC-GCN can effectively predict the links in information networks. As shown in Table 2.

Table 2. AUC values of each method under different data sets

	Citeseer	Cora	DBLP
PA	0.7153	0.6338	0.7312
Katz	0.7861	0.8821	0.6174
LRW	0.8024	0.8195	0.8744
Node2vec	0.8391	0.7493	0.8013
LINE	0.7786	0.7591	0.8846
GCN	0.8733	0.9231	0.8379
KNN-GCN	0.8500	0.8974	0.8863
DC-GCN	0.8912	0.9307	0.9031

For Citeseer, the paper's cited network, PA performs relatively the worst among the similarity-based traditional link prediction methods, with an AUC value of only 0.7153, which is because the number of Citeseer nodes is large and the number of connected edges is small, and PA only considers the degree of nodes, so it is not effective; Katz is suitable for small networks, so it achieves better results among the traditional algorithms; LRW algorithm is more suitable for large networks with sparse nodes, so the AUC values on the Citeseer dataset are lower than those on the Cora dataset, and the AUC values on both datasets are average because both datasets are small. The neural network-based link prediction method Node2Vec improves the generation of random wandering, so that the generated random wandering can reflect the characteristics of both depth-first and breadth-first sampling, thus increasing the accuracy of embedding and improving the effect of network embedding, thus achieving a good score of 0.8391; LINE is more suitable for large-scale networks, and the accuracy on the Citeseer and Cora datasets The accuracy on the Citeseer and Cora datasets is somewhat lower compared to that on the DBLP dataset; the link prediction methods of GCN and KNN-GCN are more effective, with AUC values above 0.85, but the AUC value of GCN is 2.33% higher than that of KNN-GCN because the topological information is more valuable than the attribute information in the link prediction task on the Citeseer dataset. The method in this paper integrates the topological and attribute information of nodes and improves the AUC metric by 1.79% over GCN, which is the best performer in the traditional method.

For Cora, a paper citation network with fewer node types and edge types, the PA method considering node degrees performs worse than on the Citeseer dataset, with an AUC value of only 0.6338; the Cora dataset has fewer nodes than the Citeseer dataset, but more connected edges than the Citeseer dataset and Katz considers all paths between nodes. Therefore, the results are better on the Cora dataset than on the Citeseer dataset. The best performance among the traditional methods is the neural network-based link prediction method GCN. Similar to the Citeseer data and the Cora data set, the topological information is more valuable than the attribute information in the link prediction task, so the AUC value of GCN is higher than that of KNN-GCN. And the proposed method in this paper even works better than GCN with 0.76 improvements in the AUC metric.

Since the number of connected edges and nodes of the bibliographic information network DBLP is larger, the network size becomes larger and the PA effect decreases compared to the Citeseer and Cora datasets, while Katz, LRW, and LINE effects all improve. Among the traditional methods, the Katz algorithm, which considers global paths and uses the entire topological information of the network to calculate them, is the least effective for large-scale networks, with an AUC value of only 0.6174. In comparison, KNN-GCN, a neural network-based link prediction method, makes the best accuracy performance among the traditional methods due to the high number of features in the DBLP dataset. However, DC-GCN considers both the topological information of the network and the attribute information of the nodes through the two-channel network, and there is a 1.68% improvement compared with KNN-GCN, which proves that the DC-GCN algorithm can get better prediction results by considering

both the topological information of the network and the attribute information of the nodes.

Although the AUC value is the most important metric to evaluate the link prediction performance, the Precision value plays an important role when the AUC values do not differ much, and a higher Precision value indicates a higher accuracy of the algorithm. Among the baseline PA, Katz, LRW, Node2vec, LINE, GCN, and KNN-GCN, the GCN method has the highest Precision values of 0.81, 0.71, and 0.71 on the three datasets, respectively, while the DC-GCN has Precision values of 0.84, 0.74, and 0.87, respectively, which are higher than the comparison methods. The best-performing GCN method improved by 3%, 3%, and 7%, respectively, proving that the DC-GCN method is more stable with improved performance. The specific values are shown in Table 3.

Table 3. Precision values of various methods under different data sets

	Citeseer	Cora	DBLP
PA	0.67	0.58	0.58
Katz	0.75	0.62	0.62
LRW	0.76	0.66	0.66
Node2vec	0.70	0.65	0.73
LINE	0.69	0.55	0.68
GCN	0.81	0.71	0.71
KNN-GCN	0.60	0.53	0.57
DC-GCN	0.84	0.74	0.87

By comparing with the traditional GCN and KNN-GCN, it can be seen that there are indeed structural differences between the topological and feature maps, and neither the link prediction on the topological or attribute maps alone can yield satisfactory results. In comparison, the DC-GCN proposed in this paper has a large improvement in both AUC and Precision, which further confirms the necessity of using GCN to fuse topological and attribute information.

5 Conclusion

In this paper, we introduce the idea of the k-nearest neighbor graph, and consider the first k nodes with similar attributes to nodes as having connected edges between them to construct the feature graph of nodes; use GCN to construct dual channels to obtain topological embedding and attribute embedding of nodes; combine the attention mechanism to complete the adaptive fusion of topological embedding and attribute embedding; calculate the Hadamard distance between nodes to represent the edge vector between nodes, and train the logistic regression function to The link prediction is performed by training a logistic regression function to obtain the probability of generating connected edges between nodes. This method fuses more semantic and structural information of nodes and reflects the original graph information more comprehensively in the vector space so that the vector represents the node information more completely. Through experimental validation, DC-GCN shows good performance on real data sets.

However, the method proposed in this paper is only applicable to homogeneous static networks and does not take into account the heterogeneity and dynamics of the networks, and future research will proceed to the analysis of the heterogeneity and dynamics of the networks.

6 Acknowledgement

This work was supported by the project fund: Inner Mongolia Natural Science Foundation “Research on Heterogeneous Network Link Prediction Method Based on Deep Learning of Graph Representation” (2022MS06006).

References

[1] A. Kumar, S.-S. Singh, K. Singh, B. Biswas, Link prediction techniques, applications, and performance: a survey, Physica A: Statistical Mechanics and its Applications 553(2020) 124289.

[2] Y. Pan, X.-J. Lei, Y.-C. Zhang, Association predictions of genomics, proteinomics, transcriptomics, microbiome, metabolomics, pathomics, radiomics, drug, symptoms, environment factor, and disease networks: A comprehensive approach, Medicinal Research Reviews 42(1)(2022) 441-461.

[3] J. Singh, T. Litfin, J. Singh, K. Paliwal, Y.Q. Zhou, SPOT-Contact-LM: improving single-sequence-based prediction of protein contact map using a transformer language model, Bioinformatics 38(7)(2022) 1888-1894.

[4] E. Nasiri, K. Berahmand, Z. Samei, Y. Li, Impact of centrality measures on the common neighbors in link prediction for multiplex networks, Big Data 10(2)(2022) 138-150.

[5] X. Yue, Z. Wang, J. Huang, S. Parthasarathy, S. Moosavinasab, Y. Huang, S.M. Lin, W. Zhang, P. Zhang, H. Sun, Graph embedding on biomedical networks: methods, applications and evaluations, Bioinformatics 36(4)(2020) 1241-1251.

[6] T. Zhao, X. Zhang, S. Wang, Graphsmote: Imbalanced node classification on graphs with graph neural networks, in: Proc. ACM International Conference on Web Search and Data Mining (WSDM), 2021.

[7] R. Xie, C. Ling, Y. Wang, R. Wang, F. Xia, L. Lin, Deep feedback network for recommendation, in: Proc. International Joint Conferences on Artificial Intelligence, 2021.

[8] J. Bruna, W. Zaremba, A. Szlam, Y. Lecun, in: Proc. Spectral networks and locally connected networks on graphs, International Conference on Learning Representations, 2014.

[9] Y. Dong, Q. Liu, B. Du, L. Zhang, Weighted feature fusion of convolutional neural network and graph attention network for hyperspectral image classification, IEEE Transactions on Image Processing 31(2022) 1559-1572.

[10] B.-B. Xu, K.T. Cen, J.-J. Huang, H.-W. Shen, X.-Q. Cheng, A survey on graph convolutional neural network, Chinese Journal of Computers 43(5)(2020) 755-780.

[11] H.-H. Chen, G.-D. Wu, J.-X. Li, J.-Y. Wang, H. Tao, Research advances on deep learning recommendation based on attention mechanism, Computer Engineering & Science 43(2)(2021) 370-380.

[12] F. Lorrain, H.-C. White, Structural equivalence of individuals in social networks, The Journal of mathematical sociology 1(1)(1971) 49-80.

[13] T. Zhou, L. Lü, Y.-C. Zhang, Predicting missing links via local information, The European Physical Journal B 71(4)(2009) 623-630.

[14] L. Lü, C.-H. Jin, T. Zhou, Similarity index based on local paths for link prediction of complex networks, Physical Review E, 80(4)(2009) 046122.

[15] L. Katz, A new status index derived from sociometric analysis, Psychometrika 18(1)(1953) 39-43.

[16] P.-M. Chuan, L.-H. Son, M. Ali, T.D. Khang, L.T. Huong, N. Dey, Link prediction in co-authorship networks based on hybrid content similarity metric, Applied Intelligence 48(8)(2018) 2470-2486.

[17] D. Kalman, A singularly valuable decomposition: the SVD of a matrix, The college mathematics journal 27(1)(1996) 2-23.

[18] W. Chen, G. Feng, Spectral clustering: a semi-supervised approach, Neurocomputing 77(1)(2012) 229-242.

[19] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: online learning of social representations, in: Proc. ACM SIGKDD international conference on Knowledge discovery and data mining, 2014.

[20] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proc. ACM SIGKDD international conference on Knowledge discovery and data mining, 2016.

[21] J. Tang, M. Qu, M.-Z. Wang, M. Zhang, J. Yan, Q.-Z. Mei, Line: Large-scale information network embedding, in: Proc. International Conference on World Wide Web, 2015.

[22] P. Velikovi, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: Proc. International Conference on Learning Representations, 2018.

[23] H. Wang, J. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, M. Guo, Graphgan: Graph representation learning with generative adversarial nets, in: Proc. AAAI conference on artificial intelligence, 2018.

[24] O. Du, Y. Li, Academic Collaborator Recommendation Based on Attributed Network Embedding, Journal of Data and Information Science 7(1)(2022) 37-56.

[25] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, H. Liu, Attributed network embedding for learning in a dynamic environment, in: Proc. ACM Conference on Information and Knowledge Management (CIKM), 2017.

[26] M. Cokun, M. Koyutürk, Node similarity-based graph convolution for link prediction in biological networks, Bioinformatics 37(23)(2021) 4501-4508.

[27] J. Ma, P. Cui, K. Kuang, X. Wang, W. Zhu, Disentangled graph convolutional networks, in: Proc. International conference on machine learning (ICML), 2019.

[28] F. Wu, T. Zhang, A. Souza, C. Fifty, T. Yu, K.Q. Weinberger, Simplifying graph convolutional networks, in: Proc. International conference on machine learning (ICML), 2019.

[29] J.-X. You, R. Ying, J. Leskovec, Position-aware graph neural networks, in: Proc. International conference on machine learning (ICML), 2019.

[30] Z. Yang, W.-W. Cohen, R. Salakhutdinov, Revisiting semi-supervised learning with graph embeddings, in: Proc. International conference on machine learning (ICML), 2016.