Information Sciences 548 (2021) 1–17

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Link prediction based on feature representation and fusion

Yunpeng Xiao *, Rui Li, Xingyu Lu, Yanbing Liu

College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

ARTICLE INFO

Article history:
Received 6 January 2020
Received in revised form 12 August 2020
Accepted 17 September 2020
Available online 9 October 2020

Keywords:
Link prediction
Social networks
Network embedding
Convolutional neural network
Feature fusion

ABSTRACT

Link prediction is one of the core problems in social network analysis. Considering the complexity of features in social networks, we propose a link prediction method based on feature representation and fusion. Firstly, based on the sparseness and high-dimensionality of network structure, network embedding is applied to represent the network structure as low-dimensional vectors, which identifies the spatial relationships and discovers the relevance among users. Second, owing to the diversity and complexity of text semantics, the user text is converted into vectors by word embedding models. As user behaviors can reflect the dynamic change of links, a time decay function is introduced to process the text vector to quantify the impact of user text on link establishment. Meanwhile, to simplify the complexity, we choose the top-*k* relevant users for each user. Finally, due to the attention mechanism can improve the expression of user's interests in text information, a link prediction method with attention-based convolutional neural network is proposed. By fusing and mining structural and text features, the purpose of synthetically predict link is finally achieved. Experimental results show that the proposed model can effectively improve the performance of link prediction.

1. Introduction

With the rise of social networks such as YouTube, LinkedIn, Flickr and Sina Weibo, online social networks have gradually developed into global super networks. People can communicate and interact with others through these platforms, which brings great convenience to people's lives. Therefore, social networks are attracting not only an increasing number of users, but also the attention of scholars. In many social networks studies, link prediction is an effective method of information mining and prediction. Link prediction refers to inferring possible (invisible) links and future links from the observed user information [24]. Mining the potential relationships among users can be used for friend recommendation and community detection [7,11]. The description of network structure provides new ideas for the study of network evolution [14]. Furthermore, link prediction also plays a crucial role in other domains, such as protein detection [19], personalized recommendation [44], identifying abnormal communications in the security field [35], and so on.

In recent years, many scholars have done a lot of research on link prediction, including network structures [3,8,10,23,35], user attributes [1,15,17,27,37,38], and user behaviors [20,22,30]. The method based on network structures mainly studies the structural similarity. User attributes studies primarily consider user's interests, geographic information, text information, and so on. And in terms of user behaviors, scholars mainly focus on two fields: user individual behaviors and social behaviors. Although current research on link prediction has achieved remarkable results, there are still some challenges:

* Corresponding author.
E-mail address: xiaoyp@cqupt.edu.cn (Y. Xiao).

https://doi.org/10.1016/j.ins.2020.09.039
0020-0255/

- The high-dimensional and sparse structure. The real social networks are characterized by vast numbers of nodes and a high-dimensional structure. Therefore, manual features are extremely difficult to extract, and the constructed adjacency matrix is very sparse.
- The complexity of text semantics. Numerous messages pass through the daily communication of social networks. The complexity, diversity, and redundancy of messages can reduce the accuracy and efficiency of link prediction.
- The polymorphic input and the dynamic network. There are various data in social networks, such as network structures, text information, and other factors, which change over time. These factors can affect the link establishment [2,21]. Therefore, how to combine the above features together for link prediction has become the main challenge in link prediction research.

To meet the above challenges, a link prediction model based on feature representation and fusion is proposed in this paper, which combines network structures and text information, to achieve this task. Then, Exploiting the network embedding in feature representation, our method reduces the dimension and unifies the expression of network structure and user text respectively. And the network is mined by the attention-based convolutional neural network that finds the correlation among users. Our method is evaluated on real datasets. Our contributions can be summarized as follows:

- Multi-feature spaces are uniformly expressed through representation learning. The network structures and text information, which have different data structures, are expressed as vectors by different network embedding. This operation converts the data in different feature spaces into a unified low-dimensional vector format.
- Time decay function is introduced to improve the expression of user's interests. To make the influence of the text information on link establishment more realistic, the time decay function is used to dynamically fit the change of the user's interests. It not only further quantifies the user behaviors, but also solves the problem that the user's interests decline over time.
- A link prediction method based on convolutional neural network which integrates multiple features is proposed. To improve the performance of link prediction, the method takes into account the network structure and user text characteristics comprehensively, and fully explores the user relationships through the CNN.

The rest of this paper is organized as follows. Section 2 describes the related work of link prediction. Section 3 formulates the problems and gives the related definitions. Section 4 presents the proposed method and related learning algorithms. Section 5 analyzes the experimental results of the method. Section 6 summarizes the work of this paper.

2. Related work

For the three challenges mentioned in the previous part, this section mainly introduces the research and applications of link prediction in social networks analysis in recent years.

First of all, aiming at the problem of high dimensionality of network topology, it mainly focuses on that the network structure is difficult to extract and express. This requires that the algorithm can simply express the node and neighborhood relationship [9], so it can be used in many fields of social networks [4]. DeepWalk [28] used random walk strategy to obtain vector representation of network structure, and Grover [12] extended the search strategy of DeepWalk, which can capture homogeneous nodes and structural equivalence nodes simultaneously. LINE [31] considered both the direct similarity and 2-hop similarity of nodes, but there is lack of flexibility on exploring nodes at deeper networks. SDNE [34] proposed a deep learning model to capture the non-linear relationship among nodes, however its code implementation is complex. Tu et al. [32] considered the contextual information, but its context only includes the text information of the nodes connected by an edge. MMDW [33] combined 1-order and 2-order similarity into a unified matrix, which results in information loss in different neighborhoods.

Secondly, for the diversity and complexity of text information, it primarily considers that the link establishment is influenced by both the network structure and text content. However, there are few research on text content. Ref. After mapping the high-dimensional, sparse text to a low-dimensional, dense vector, the semantic correlations among texts can be directly computed. [25,18] Wang et al. [36] defined a topic inclusion degree (TID) score to measure the relationships among users. The link prediction problem is then solved by integrating the structural and content information using matrix decomposition technology. However, this method is not suitable for large-scale networks. An extensible deep learning link prediction framework, DeepLink, is proposed for the first time in [16], which considers both the node structural information and content information. Jiang et al. [15] proposed a social text topic modeling method based on domain dictionary. However, this method has a high time complexity due to its large-scale text content.

Finally, with the widely application of deep learning [29,42], neural network plays an increasingly important role in social networks. Ref. [40] proposed a new link prediction framework, SEAL, which simultaneously learns local closed subgraphs, embedding graphs and graph-based neural network attributes. But SEAL only considers the network structures. Zhang et al. [41] put forward a new attention-based deep neural network. This model combines the contents of tweets, user's interests and their similarity, author information, and other factors to predict the retweet behavior. Pan et al. [26] proposed a deep multi-label learning framework (iDeepM) that predicts RNA-binding proteins. Cao et al. [5] formed a self-social

attribute network that integrates user attributes, social links, and comments, and then input it into CNN. This model improves the accuracy of user's interests reasoning but ignores the overall structure of the network. Gu et al.[13] proposed DeepLinker, which extends the GAT architecture to predict missing links. Building on the previous work, this paper designs a link prediction method of CNN that combines multi-feature information to overcome the existing challenges.

3. Problem formulation

In order to formally formulate the problem of our research, let $G^t = (U, E^t)$ be the whole network at time t, where $U = \{u_1, u_2, \ldots, u_n\}$ represents the set of users and $E^t \subset U \times U$ refers to the set of links among users at moment t, where each edge $e \subseteq E^t$ is an unordered pair $e = (u_i, u_j)$. $|U| = N$ is the total number of users in the network, and $|E^t| = M$ is the total number of links in the network. And let $A = \{(a, u_i, t)|t \in \Psi, u_i \in U\}$ represents the historical text information of all users. First, the network structure and user text which are expressed in different forms, are represented in vectors by different network embedding algorithms, where R^d is a d-dimensional network structure vector and R^l is a l-dimensional text vector. Then, the correlation between each node is mined and the most relevant user group $G_p : \{u_i, u_{i+1}, u_{i+2}, \ldots u_{i+k+1}\}$ of each node is selected to simplify the computational complexity. Meanwhile, the time decay function $f(u_i)$ is introduced into user text matrix V_a to obtain V_a_decay. Finally, the network structure matrix V_u is fused with V_a_decay and constructing the CNN prediction model to get the final prediction result, which is the link at the next moment. Formally, the problem is defined as predicting the link in the network at time $t+1$ from the network structure at time t and the historical user text information:

$$\left.\begin{array}{c} G^t \rightarrow G_p^t \rightarrow V_u \\ A, G_p^t \rightarrow V_a \rightarrow V_a\_decay \end{array}\right\} \Rightarrow f : (V_u, V_a\_decay) \rightarrow E^{t+1}$$

3.1. Problem input

Based on the relevant definition, the input of this method is defined as:

1. User relationship network at time t $G^t = (U, E^t)$;
2. Historical text information of users $A = \{(a, u_i, t)|t \in \Psi, u_i \in U\}$, where (a, u_i, t) indicates that user u_i published or forwarded information a at time t, and Ψ represents the range of the historical information.

3.2. Problem output

According to the above-mentioned description, the problems to be solved are as follows.

1. How to transform the characteristics of different features into a unified vector expression? According to different features of network structure and user text, we use different network embedding algorithms to transform them into low-dimensional vectors with a unified form. And then the most relevant user group G_p is obtained by analyzing the correlations of nodes.
2. How to model and predict the future link among users? First, V_u and V_a are obtained by different network embedding algorithms. The time decay function $f(u_i)$ is introduced to get V_a_decay, which measures the influence of user's interests on link establishment. Second, V_a_decay is input to the attention layer of CNN model, then the result and V_u are concatenated to perform convolute operation and the final result E^{t+1} is obtained.

4. Proposed method

To solve the described problems above, we propose a link prediction model based on user relationships and text information. This section introduces the three stages of the model in detail: feature representation, feature fusion, and link prediction, as shown in Fig. 1. In the first stage, the main related factors are analyzed, feature representations are carried out for different features, and expresses them in a unified form. In the second stage, the change of user text information over time is analyzed, and the two feature vectors are fused to build a link prediction model based on CNN. In the third stage, the trained model is used for predicting the links that will generate at the next moment.

4.1. Feature representation

There are about three forms of data in social networks, network structure, user text and photos or videos published by users. The data of different forms are different in structure and are characterized by high-dimensionality and complexity. Traditional feature representation methods (such as adjacency matrix, one-hot vector, etc.) are difficult to deal with. Therefore, how to extract multiple features and uniformly express them have become a pressing problem to be solved. This paper selects two main features for link prediction: network structure and user text.

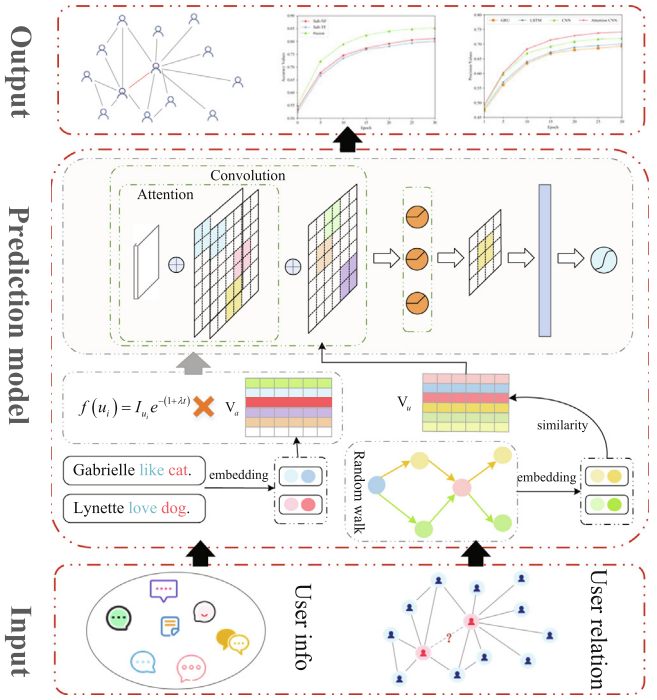


Fig. 1. Framework of link prediction model.

4.1.1. Feature representation of network structure

Network embedding is used to extract the features of network structure, so that we can reconstruct the network. In this way, the correlations among users can be found, and the disadvantages of manually constructing adjacency matrix can be avoided. Node2vec [12] can learn the homogeneity and structural equivalence of nodes. Therefore, it can capture the potential relations among users, and we can construct the feature vector of network structure.

The purpose of network structure embedding is to obtain richer and deeper feature expression among social nodes and mines the potential correlations among nodes. To meet these requirements, a flexible neighborhood sampling strategy is proposed in node2vec. Node2vec improves the random walk strategy by introducing two parameters p (Return parameter) and q (In-out parameter) for smooth interpolation between the BFS and DFS when generating a sequence of nodes. Intuitively, the parameter p controls the region of random walk, and the parameter q determines the way of random walk (that is, whether the way of random walk is BFS or DFS). By combining these two parameters and performing several random walks, the homogeneity and structural equivalence of nodes in the network can be more comprehensively represented. This paper trains node2vec by three steps (as shown in Fig. 2). In the first step, any node U selected from network $G^t = (U, E^t)$ performs a biased random walk to obtain a node sequence $\{A, B, C, D, E, N\}$. Second, through repeated iterations and a resourceful set of global node sequence walks are obtained. In the third step, these walks are input to the skip-gram model of word2vec for training, and the d-dimensional global feature vectors of all nodes in G are output.

Different sampling strategies will result in different feature representations. The parameter p can control the random walk to favor the neighboring node of the current node, such as A, B, C and D in Fig. 2. While the parameter q pays more attention to structurally equivalent nodes, such as node N in Fig. 2. Therefore, different search strategies on the network are necessary. Similarly, the goal of node embedding is to maximize the log probability of the sampled node neighbors, i.e. to maximize the probability of the nearest neighbors of the given node. The formula is

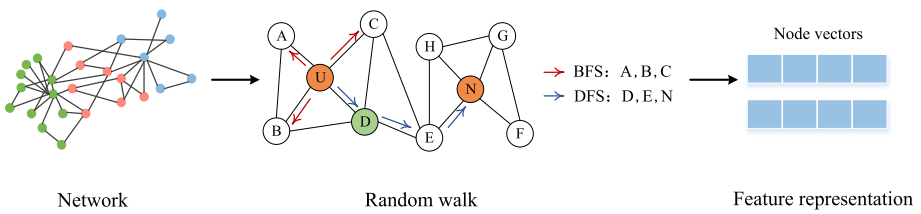


Fig. 2. Network Structure Feature Representation.

$$\max_{f} \sum_{u \in U} \log Pr(N_S(u)|f(u)) \tag{1}$$

where U is the set of nodes and $f : U \rightarrow R^d$ denotes a mapping function. $N_S(u) \subset U$ is the neighbor set of node u obtained by sampling strategy S.

4.1.2. Feature representation of user content

User text and network structure of a social network belong to different feature spaces. To obtain a unified form of both feature, the user text feature is extracted by the skip-gram model and converted into low-dimensional vectors. Traditional text embedding methods include VSM and one-hot vector. The former is prone to cause a dimension disaster when expressing a large number of texts, while the latter is simple but ignores the semantic correlations among words. In contrast, the skip-gram model can map high-dimensional, sparse words to low-dimensional, dense word vectors, and can directly calculate the semantic relevance between words.

The text information published by users will affect the link establishment among users. In general, active users that frequently interact with other users, and these users will more likely establish links than passive users. User text includes original tweets, retweets and comments on other people's tweets. Each tweet is a sequence of words. In this paper, the skip-gram model represents the complex tweet as low-dimensional vectors, and then mines user behavior rules. Specifically, we constructs a large vocabulary of text extracted from the data set, the words are then encoded as one-hot vectors to input the model. After the calculations in the hidden layer and the output layer, each output neuron generates a probability value between 0 and 1, which is the contextual vector of the input word.

To reduce the influence of the noise words unrelated to the input words and to simplify the computation, we optimize the calculation by negative sampling and use the following objective function,

$$\log \sigma\left(c = 1 \middle| v_w, v'_w\right) + \sum_{i=1}^{k} \mathrm{E}_{\omega_i \sim P_{noise}(\omega)} \left[\log \sigma\left(c = 0 \middle| v_w, v'_w\right)\right] \tag{2}$$

where v_w and v'_w are the input and output vectors of word w, respectively. $\sigma(\cdot)$ is the sigmoid function, $\sigma(c = 1|v_w, v'_w)$ represents the probability of predicting its context words based on the target word, while $\sigma(c = 0|v_w, v'_w)$ denotes the probability of predicting the noisy words according to the target word. $P_{noise}(w)$ is the distribution of noisy words.

4.2. Feature fusion

4.2.1. Optimization of model input

After the feature embedding in Section 4.1, the network structure space and user text space, which are in different forms, have been vectorized. Considering that the massive amounts of nodes and the redundancy of text information in social networks, this section will simplify the feature representation results of the above two sections.

First, the vectors of the nodes in the network are obtained by extracting and re-expressing the network structure in Section 4.1.1. Therefore, the correlations between a pair of nodes is transformed into the computation of semantic similarity between these vectors. In general, the higher the similarity between two nodes, the stronger is the correlation between the nodes; conversely, low similarity denotes a small correlation. Given any two nodes u_i and u_j, whose vector is $v(u_i)$ and $v(u_j)$ respectively, we use the cosine of the angle between two vectors to measure the similarity between these two nodes.

$$sim(u_i, u_j) = \cos\theta = \frac{\left\| v(u_i) \times v(u_j) \right\|}{\left\| v(u_i) \right\| \cdot \left\| v(u_j) \right\|} = \frac{\sum_i^d x_i \cdot y_i}{\sqrt{\sum_i^d x_i} \cdot \sqrt{\sum_i^d y_i}} \tag{3}$$

where d is the dimension of the vector, x_i and y_i represent the components of $v(u_i)$ and $v(u_j)$, respectively. For each node u_i, we calculate its similarity with other nodes by the above formula. Meanwhile, due to nodes with small similarity will be lowly correlated and will unlikely establish a link, and selecting all nodes will increase the complexity of the model. Thus, to simplify the calculation, we choose only top-k nodes with the highest similarity of the node topology, and form the most relevant user group $G_p : \{u_i, u_{i+1}, u_{i+2}, \ldots u_{i+k+1}\}$ of node u_i. Where each user group is a two-dimensional matrix of $R^{(k+1)\times d}$, as shown in formula 4. For each node, we predict links with nodes' most relevant user group, and the selection of the optimal k is given in Section 5.2.2.

$$\mathrm{V}_u = \left[\, v(u_0) \quad v(u_1) \quad \cdots \quad v(u_k) \,\right]^{\mathrm{T}} \in R^{(k+1)\times d} \tag{4}$$

where $v(\cdot) \in R^d$ denotes the node vector, $u_i \in [u_0, u_k]$ represents different nodes, and each row of V_u is the vector representation of a node in the social network.

Then, for the word vector obtained in Section 4.1.2, according to the G_p of node, each word in a tweet is converted into a l-dimensional vector. We create the text vector matrix $V_a \in R^{((k+1)\times m)\times l}$ of $k + 1$ users, as shown in formula (5), where m is the (zero-padded) user's text length.

$$V_a = [\, v(p_{01}) \quad \cdots \quad v(p_{0m}) \quad \cdots \quad v(p_{k1}) \quad \cdots \quad v(p_{km}) \,]^T \in R^{((k+1)\times m)\times l} \tag{5}$$

where $v(\cdot) \in R^l$ is the l-dimensional word vector of text, $p_{rj}, r \in [0, k+1], j \in [1, m]$ denotes the j-th word of the r-th user's sentence.

Second, since user's interests in social networks often change with time, this feature leads to a time-decaying feature of user's interests. Therefore, we introduce the time weight coefficient based on the user's published tweet, which boosts the importance of recently published content in the prediction process. Specifically,

$$f(u_i) = I_{u_i} e^{-(1+\lambda t)} \tag{6}$$

where I is an indicator function with value of 0 or 1, which indicating whether user text features will decay over time. t represents the difference between the current time and the time at which the user published the tweet. λ is weight growth coefficient, which is an adjustable parameter.

Finally, V_a_decay is obtained by calculating the user text vector V_a and the time decay function, which is still a matrix, as shown in formula (7),

$$V_a\_decay = f(u_i) \times V_a \tag{7}$$

4.2.2. Fusion model construction

According to the two features in the previous stage, the problem to be solved is how to integrate these two features into a unified form for the later modeling and link prediction tasks. Associative with attention mechanism can help the model assign different weights to each word, extract more critical information from the text, and make more accurate prediction. Moreover, considering that CNN has the ability to capture local features, and can also effectively reduce the computational complexity and the over-fitting problem through the weight sharing and pooling. In this stage, we use the model based on attention to solve this problem, as shown in Fig. 3.

As described in Section 4.1, this paper respectively constructs different feature representations of the network structure and the tweets published by users, and mines the correlations among nodes. In order to integrate the structural and textual features of the network, reduce the impact of single feature on the prediction results and enhance the accuracy of prediction, these features will be combined to link prediction.

The proposed method is shown in Fig. 3, which is added an attention layer between the input layer and the convolution layer. The attention layer selectively considers the more important information about the user's interests in the input vector, informing which words should be given more weight. The model consists of one input layer, one attention layer, one feature fusion layer, two convolution and pooling layers, one fully-connected layer and one output layer, in which the convolution layer and pooling layer appear alternately. In Fig. 3, the dotted line is used as the boundary and the first step on the left is attention layer. Then the processed text features and network structure features are integrated. On the right side of the dotted line are the convolution and pooling operations. This section focuses on feature fusion operations.

The input of the model is V_a_decay. First, the attention layer creates a contextual vector c_i for each word by formula (8) and concatenates these with the original word vectors as the subsequent input of the model. When mining user text, attention mechanism determines which words are more concerned than others.

$$c_i = \sum_{j \neq i} \mu_{i,j} \cdot v(p_{rj}) \tag{8}$$

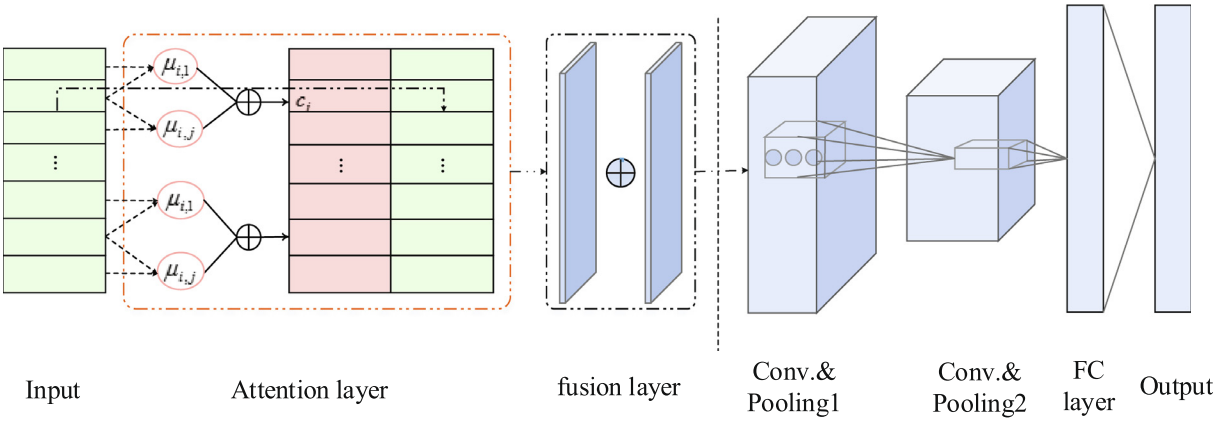


Fig. 3. Details of the model.

where $u_{i,j}$ denotes the attention weight and by softmax regularization to make $u_{i,j} \geqslant 0$ and $\sum_j u_{i,j} = 1$. $u_{i,j}$ and the scoring function is calculated by formula (9) and (10) in [43]. Where v_a and W_a are the parameters of the attention module, which are jointly trained with other parts of the model.

$$\mu_{i,j} = \frac{\exp\left(score\left(v(p_{ri}), v(p_{rj})\right)\right)}{\sum_{j'} \exp\left(score\left(v(p_{ri}), v(p_{rj'})\right)\right)} \tag{9}$$

$$score\left(v(p_{ri}), v(p_{rj})\right) = v_a^{\mathrm{T}} \tanh\left(v(p_{ri}) W_a v(p_{rj})\right) \tag{10}$$

Next, the contextual vector c_i and $v(p_{ri})$ are concatenated into a new word vector $v(p_{ri})'$, as shown in formula 11. After that, the sentence matrix V_s is constructed by concatenating words from users' each tweet. The fusion operation is then carried out by formula 12.

$$v\prime(p_{ri}) = v(p_{ri}) \oplus c_i \tag{11}$$

$$\mathrm{V}_s = \begin{bmatrix} v'(p_{01}) \oplus v'(p_{02}) \oplus \cdots \oplus v'(p_{0m}) \\ v'(p_{11}) \oplus v'(p_{12}) \oplus \cdots \oplus v'(p_{1m}) \\ \vdots \\ v'(p_{k1}) \oplus v'(p_{k2}) \oplus \cdots \oplus v'(p_{km}) \end{bmatrix} = \begin{bmatrix} v'(p_0) \\ v'(p_1) \\ \vdots \\ v'(p_k) \end{bmatrix} \in R^{(k+1)\times(m\times 2l)} \tag{12}$$

Finally, the result V_s of attention layer is combined with the network structure vector V_u to obtain a network node matrix $V \in R^{(k+1)\times(d+(m\times 2l))}$, which is as the input of the first convolution layer.

$$\mathrm{V} = \mathrm{V}_s \oplus \mathrm{V}_u = \begin{bmatrix} v'(p_0) \\ v'(p_1) \\ \vdots \\ v'(p_k) \end{bmatrix} \oplus \begin{bmatrix} v(u_0) \\ v(u_1) \\ \vdots \\ v(u_k) \end{bmatrix} \in R^{(k+1)\times(d+(m\times 2l))} \tag{13}$$

4.3. Prediction model

4.3.1. Link prediction

This section mainly discusses the operation on the right of the dashed line in Fig. 3, and learns the local features and the overall spatial features of nodes. The convolutional layer generates a feature map by convolution using filters of window size f, and extracts local features from the matrix, as shown in formula (14),

$$y^{conv} = f(W * \mathrm{V} + b_i) \tag{14}$$

where y^{conv} represents the output of convolution operation, $f(\cdot)$ represents a nonlinear activation function ReLU, $W \in R^{f\times f}$ is the filter matrix, and b_i is the bias term.

To find the most representative local features from the feature map, to further aggregate the values in the feature map, and to reduce the number of parameters, we perform a sampling operation on the feature map. In this paper, the maximum pooling is adopted to realize the sampling operation. Denoting y^{pool} as the output of the pooling layer, and $max(\cdot)$ as the maximum pooling operation, we have

$$y^{pool} = \max\left(y^{conv}\right) \tag{15}$$

After multi-layer convolution and pooling learning, each network node is transformed into a local feature vector, and these vectors are then combined through the fully-connected layer. The functional relationship between input and output is established, and a global feature vector is obtained, as shown in Formula (16). Finally, the output layer is connected in the fully connected way for classification. The output result is a k-dimensional vector, in which each number represents the probability that the node will establish a link with the central node of its relevant user group at the next moment.

$$y^{FC} = flatten\left(y^{pool}\right) \tag{16}$$

$$\mathrm{P}\left(\mathrm{Y} = i | y^{FC}, W, b\right) = softmax\left(W \cdot y^{FC} + b\right) \tag{17}$$

where y^{FC} represents the output of the fully-connected layer, $P(Y = i|y^{FC}, W, b)$ is the probability distribution of linking two types of tags Y, and i equals 0 or 1. The links are deemed to be existing when the value of a category of tags is greater. Assuming e^{t+1} denotes the existence of the each new link in E^{t+1} at time $t+1$, the definition is as follows:

$$e^{t+1} = \begin{cases} 1, & \mathrm{P}(\mathrm{Y}=1|y^{FC}, W, b) > \mathrm{P}(\mathrm{Y}=0|y^{FC}, W, b) \\ 0, & \mathrm{P}(\mathrm{Y}=1|y^{FC}, W, b) < \mathrm{P}(\mathrm{Y}=0|y^{FC}, W, b) \end{cases} \tag{18}$$

Suppose $e^{t+1} = 1$, the link between a pair of users is established; otherwise, there is no link between them. And the loss function is defined as:

$$L = -\sum_{x}^{k+1} \sum_{y \in Y} P'_{xy} \log P_{xy} \tag{19}$$

where $k+1$ is the number of samples, that is the number of users in the current batch, and Y is the number of classes. P'_{xy} represents the real value of the current output, and P_{xy} represents the current predicted value.

4.3.2. Learning algorithm

The input of the algorithm includes the whole network $G^t = (U, E^t)$ and the historical text information $A = \{(a, u_i, t)|t \in \Psi, u_i \in U\}$ published by users. As described in Section 3, the output of the algorithm is the link at time $t+1$ predicted from the information at time t. The experiment was divided into two parts. One is the feature representation of social networks (Algorithm 1); the other is the feature fusion and link prediction (Algorithm 2). In Algorithm 1, the network structure feature V_u and user text feature V_a are extracted by different network embedding algorithms, and the correlation among nodes is found. Meanwhile, to simplify the complexity of the model, k users that are most correlated with u_i are gathered into G_p. Then V_a and time decay function are calculated to obtain V_a_decay.

Algorithm 1. Feature representation

Input: whole network $G^t = (U, E^t)$; User history behavior $A = \{(a, u_i, t)|t \in \Psi, u_i \in U\}$;
Output: Network structure vector V_u; User text vector V_a_decay;
1: //Step1. Feature representation of network structure based on NRL
2: // Initialization
3: Set p, q, number of related users k;
4: // Training
5: Extract network structure feature and obtain node embedding $\{v(u_i)|u_i \in U\}$ from Eq. (1);
6: Calculate the correlations between $\{u_1, u_2, \ldots, u_n\}$ from Eq. (3) based on node embedding;
7: Select top-k nodes with the highest relevance for each node, and form a maximal relevant user group G_p;
8: //Step2. Feature representation of user text based on NRL
9: // Training
10: Extract user published content information and obtain user text vector $\{v(p_i)|p_i \in A\}$ from Eq. (2);
11: The maximal relevant user group is transformed into a two-dimensional matrix $V_u = [v_{u_0}, v_{u_1}, \ldots, v_{u_k}]^\top$ from Eq. (4);
12: Select G_p of each user and constitute user text matrix V_a from Eq. (5);
13: Calculate V_a_decay from Eq. (6), Eq. (7);

Then we introduce Algorithm 2. In Algorithm 2, the V_a_decay obtained in Algorithm 1 is used as the input of prediction model. Extracting crucial information from the attention layer and concatenating the word vectors into sentence vectors, and then the fusion feature V is obtained. Finally, V serves as the input to the convolution layer to predict links at the next moment.

Algorithm 2. Link prediction algorithm

Input: Network structure vector V_u; User text vector V_a_decay;
Output: Future link E^{t+1};
1: //Step1. Feature fusion
2: **for** each user text vector **do**
3: Calculate the attention weight $\mu_{i,j}$ from Eq. (9), Eq. (10);
4: Calculate the contextual vector c_i for each word by $\mu_{i,j}$ from Eq. (8);
5: Concatenate the word vector and c_i to form new word vector from Eq. (11);
6: Construct sentence matrix V_s from Eq. (12);
7: **end for**
8: Concatenate V_s with network structure vector V_u as the input of first convolution layer from Eq. (13);
9: //Step2. Prediction model

10: // Initialization
11: Randomly initialize W, b;
12: Set epoch, batchsize;
13: Choose kernel size of filter, kernel size of pooling;
14: // Training
15: **for** training data
16: Feed the sample batch of V into CNN for forward propagation;
17: Compute results y^{conv}, y^{pool}, y^{FC}, $P(Y = i|y^{FC}, W, b)$ from Eq. (14), Eq. (15), Eq. (16), Eq. (17);
18: **end for**
19: **repeat**
Update W, b by gradient descent algorithm to calculate $\frac{\partial L}{\partial W}$ and $\frac{\partial L}{\partial b}$ from Eq. (19);
20: **until** convergence;
21: **for** test data
22: Get future link E^{t+1} from Eq. (18);
23: **end for**

In this paper, the features of network structure and user text are extracted by different network embedding algorithms. Then, the correlations among users are calculated and the most relevant user group is selected. The time complexity is $O(m)$, where m is the number of users selected from data set and $m \subseteq N$. Next, the user text feature processed by time decay function is input to CNN prediction model, which extracts the crucial information. Finally, the user text matrix and network structure matrix are concatenated and input to the convolution layer for link prediction, and the time complexity is $O(m^2)$. Accordingly, the time complexity of the whole algorithm is $O(m) + O(m^2) \sim O(m^2)$.

5. Experiments and analysis

5.1. Experimental settings

In this section, experimental settings are described in detail. First of all, the experimental data is described. Next, the baseline methods compared in the experiments are introduced. Moreover, several evaluation metrics are proposed to evaluate the performance of the model and the experimental results are analyzed. Finally, the division of data sets is introduced.

5.1.1. Dataset

The data sets in this paper were extracted from Twitter and Sina Weibo. The Twitter data set is from SNAP (http://snap.stanford.edu/data/index.html), which has been built after monitoring the spreading processes on Twitter before, during and after the announcement of the discovery of a new particle with the features of the elusive Higgs boson on July 4, 2012. The messages about this discovery between July 1 to 7, 2012 and have been updated on Mar 31, 2015. For the experimental evaluation, we selected 97,632 users and 6,883,144 user relationships from this data set. Sina weibo is one of the most popular online social network platforms in China, similar to Twitter. The data set is a public data set downloaded from AMiner (https://www.aminer.cn). It randomly selected 100 users as seed users, and then collected their followees and followees' followees. To evaluate the link prediction method, we collated user relationship data and users' tweets and retweets from the data set with 28 timestamps within one month. After this process, we obtained 6,309 users, 689,739 user relationships, 1,039,478 original microblogs and 1,734,282 retweets.

5.1.2. Baseline methods

In order to verify the performance of the proposed method, we compare our approach with some baseline methods.

- E-LSTM-D [6]: The E-LSTM-D deep learning model is the first attempt to apply LSTM and encoder-decoder architecture in link prediction. E-LSTM-D automatically learns the structure and time characteristics of the network and predicts its future links.
- 3-HBP [38]: Based on the latent Dirichlet allocation topic model, this link prediction model combines the user relationship and user behavior, and mines the users' potential interests distribution to effectively improve the performance of link prediction.
- NetworkGAN [39]: This model combines the advantages of GCN, TMF, LSTM, and GAN. The spatial and temporal characteristics of the dynamic network are modeled by a unified depth generation framework.
- Other methods: For the comparison study, we also choose several traditional link prediction algorithms resource allocation (RA), Adamic-Adar (AA), preferential attachment (PA), and Jaccard coefficient (JC), and two classical binary classifiers for link prediction: Logistic regression (LR) and support vector machine (SVM).

Next, we introduce the parameter settings of the baseline methods described above. For the E-LSTM-D model, according to the settings of the original, we set the number of encoder/decoder layers K and number of LSTM cells L as 2, and the learning rate $\lambda = 0.001$. Since the two data sets in this paper are relatively large networks, the number of encoder units is set to 512 | 256, the number of LSTM units is set to 384 | 384. The number of units in the first decoder layer should be equal to the number of units in the last encoder layer, i.e. 256, and we set the number of units in the output layer in decoder equals to the number of nodes in the network, i.e. 6,309 and 97,632.

For 3-HBP, the Dirichlet priors are set as $\alpha = 50/K,\ \beta = 0.01$, and the number of interests is $K = 10$.

For NetworkGAN, the framework consists of two parts: a generative network G and a discriminative network D. The generative model G is composed of an attentive GCN model, a TMF-LSTM network and a FC layer, and the discriminative model D is a binary classifier. We introduce network G network and D respectively. Attentive GCN is the first: in both data sets, we set the number of the hidden units for GCN as 64, the number of output features as $F = 128$, and the number of neighboring nodes as $k = 10$. Next is the LSTM: Similarly, in both data sets, the number of hidden units is set to 64, and the dropout rate is set to 0.2 to avoid overfitting. TMF: we set the time-dependent matrix $d = 2$. The parameter of exponential decay function ϖ is set to 0.3, the potential dimension ι is 10, the regularizer weight $\varepsilon = \omega = 0.01$, and the maximum iteration is 200. Finally, in GAN, we set the loss balance parameter $\gamma = 0.3$, the threshold parameter as $\zeta = 0.0001$, and we apply the L2 regularization to generative model G.

5.1.3. Evaluation metrics

In this paper, we introduce accuracy, precision, recall, F1-Measure and AUC (area under receiver operating characteristic curve) to evaluate the prediction results. F1-Measure is the harmonic average of precision and recall. AUC measures the precision of the algorithm as a whole. The closer to 1, the better the predicting effect of the algorithm will be.

5.1.4. Dataset settings

In the part of data set setting, the division of the training set and the testing set of both data sets are described respectively. We assume that there is a link between a pair of users is a positive sample, and no link is a negative sample. For Twitter, we randomly select user pairs and divide them into training sets and testing sets in a proportion of 7:3, and supposed that there is no user link in the testing set. That is, we need to use the training results to predict the links in the testing set. For Sina weibo, because the data set has the time information of link establishment, the data set is divided into a training set and a testing set according to the time information. Considering the time cost of processing the large amount of data, we select the data with timestamp of 1 to 27 as the training set, and the data with timestamp of 28 as the testing set for link prediction. Then, we evaluate the performance of the algorithm by comparing the predicted results with the real network.

5.2. Performance analysis

In this section, the performance of the proposed method was evaluated in five sets of comparative experiments. First, the attention-based CNN prediction model is validated. Second, the main parameters of the proposed method are compared and

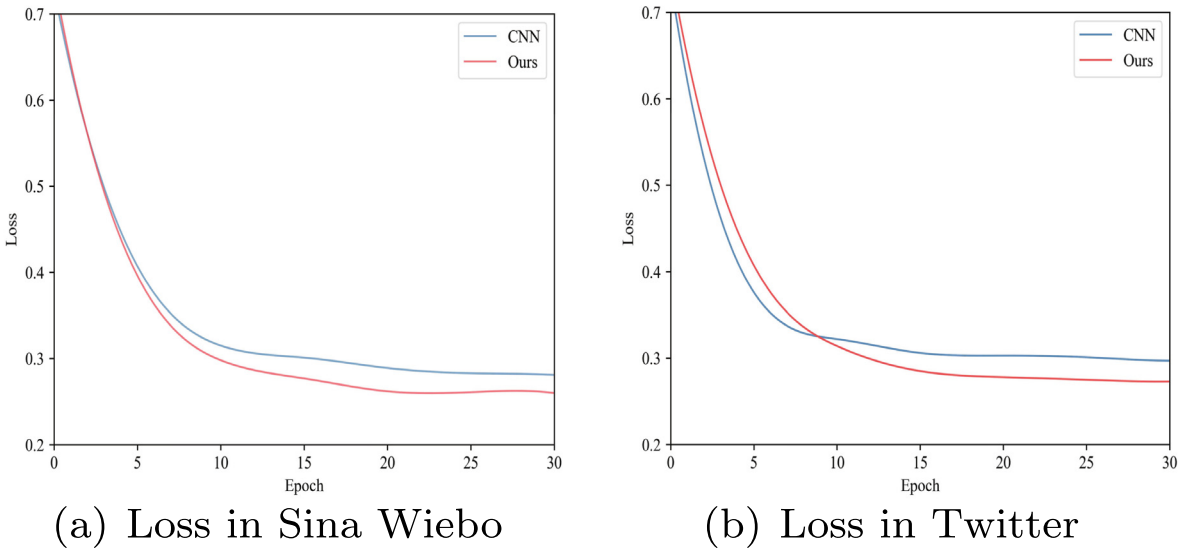


Fig. 4. Comparison of loss values of attention mechanism.

selected. Next, the performances of different representation learning algorithms are compared. Furthermore, the results of single and multiple features are presented. Finally, the effectiveness of our method is verified by comparing with other baseline methods, and computational complexity is also compared.

5.2.1. Analysis of Attention Mechanism

First, the effectiveness of attention mechanism is analyzed. In this experiment, the attention layer was inserted after the input layer of the CNN prediction model. Its function is to capture the contextual relationships among texts, and hence improve the prediction performance. To verify the effectiveness of the proposed attention mechanism, we first compare the performances in the absence and presence of the attention mechanism. The model with 128-dimensional embedding and 30 epochs, as shown in Fig. 4. Next, the NetworkGAN and E-LSTM-D models are compared with our method, the effect is shown in Fig. 5.

Fig. 4 shows the performance changing curves of loss function in two data sets. The *x*-axis represents the number of iterations, and the *y*-axis represents cross entropy loss values. As shown in Fig. 4(a), the loss values of Sina Weibo descend rapidly first, then slowly to convergence. As shown in Fig. 4(b), the result is the same in Twitter, but our method declines

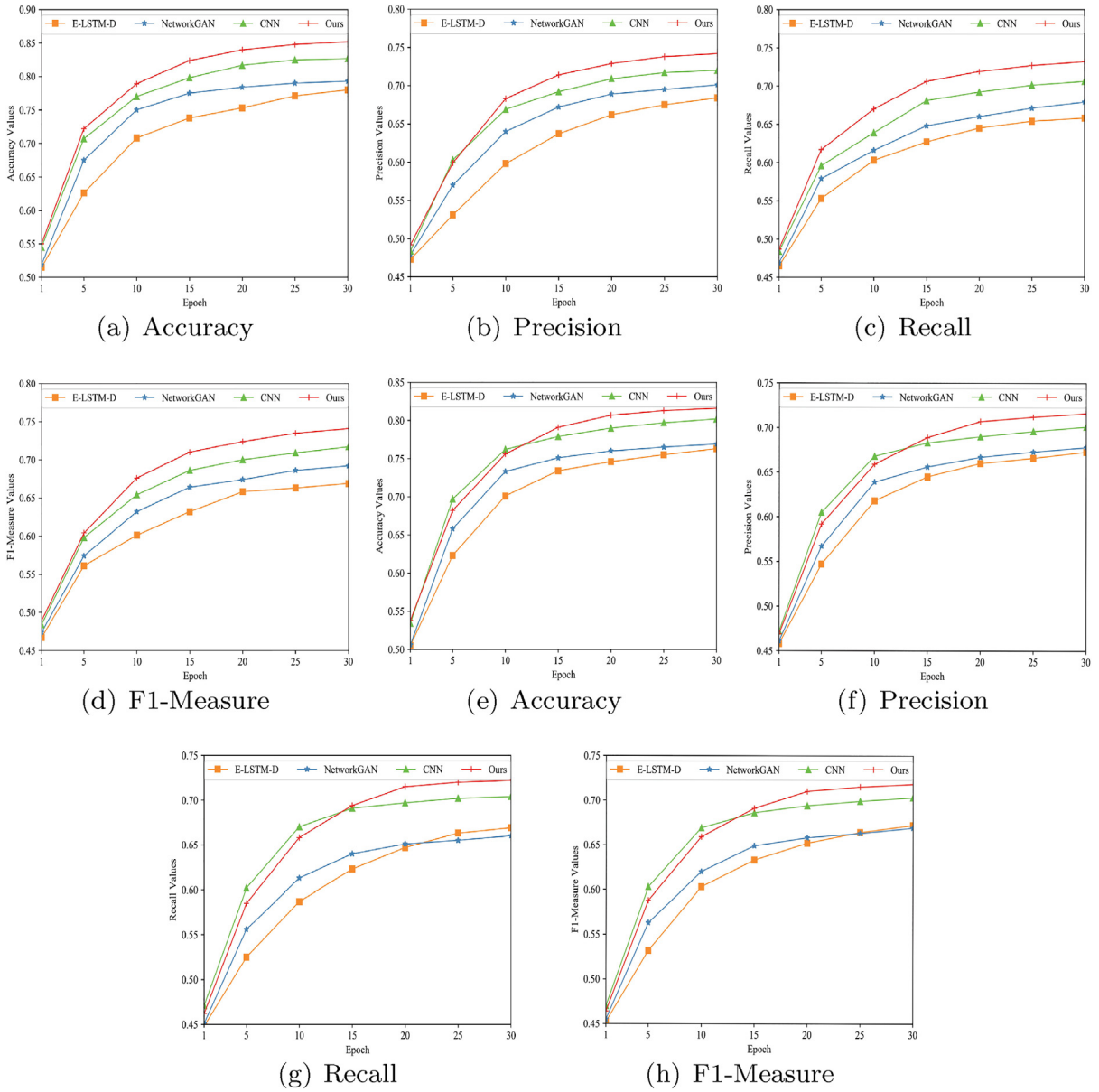


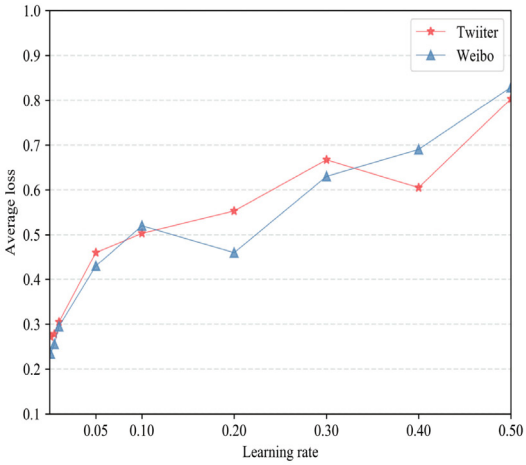
Fig. 5. Effects of attention mechanism. (a)–(d) are the results in Sina Weibo; (e)–(h) are the results in Twitter.

more slowly than the original CNN model at the beginning of training. In other words, the learning rate of CNN model is faster. The cross entropy loss of our method has a relatively large decline than CNN, indicating that the proposed method has a certain optimization effect. Fig. 5 shows the performance of different models in two data sets, including NetworkGAN, E-LSTM-D, CNN, and our method. The horizontal coordinate represents the number of epoch, and the vertical coordinate represents the values of four metrics respectively: accuracy, precision, recall and F1-Measure. Fig. 5 confirms the superiority of our method over the other methods in two data sets. In Fig. 5(e)–(h), we can also see that the learning rate of CNN model is faster. However, after multiple iterations, the performance of our method finally exceeds CNN.

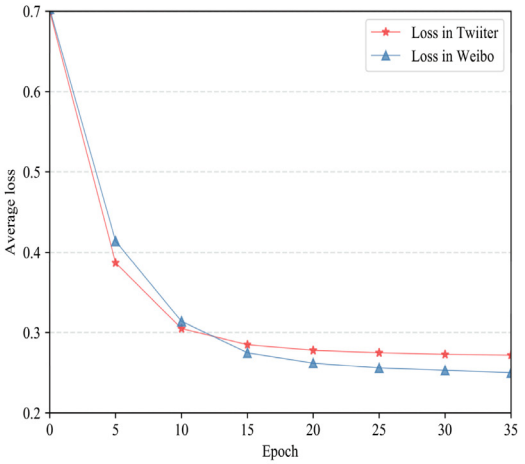
5.2.2. Comparison of parameter settings

Second, the impact of different parameters in the model is analyzed. The experiment compared: the learning rate of CNN, the epoch of CNN, and the number k of the most relevant user group.

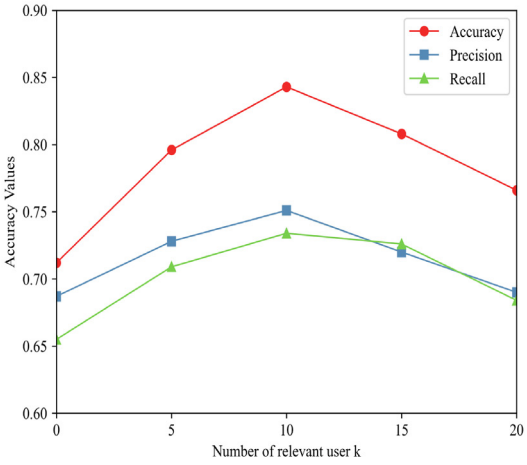
With the same setting of other parameters, the average error values of each epoch with different learning rates between 0 and 0.5 was calculated, as shown in Fig. 6(a). The x-axis represents different learning rates, and the y-axis represents the average error values. The experiment shows that when the learning rate exceeded 0.01, the average error of the model increased sharply. Therefore, when the learning rate is small, the model is stabler. In further experiments, the learning rate



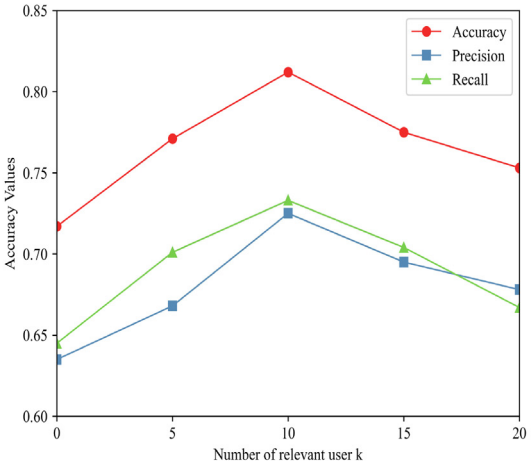
(a) The selection of learning rate



(b) The selection of epoch



(c) The selection of k in Sina Weibo



(d) The selection of k in Twitter

Fig. 6. Selection of parameters.

is set to 0.001. In Fig. 6(b), the *x*-axis represents different epochs, and the *y*-axis represents the average error values. The model tends to be flat after 20 epochs, and the model has converged at 30 epochs. Thus, the epoch is set as 30 in subsequent experiments.

In addition, this paper also compares the selection of *k*. In this paper, the value *k* represents the number of users with the most similar structure to the central user, and its optimal value is determined according to the experimental results, as shown in Fig. 6(c) and (d). With the increase of the number of relevant users, the values of evaluation indices increased first and then decreased in two data sets. We also found the peak value of the evaluation metrics, $k = 10$, in other words, the optimal value of the number of relevant users *k* is 10. If *k* is small, most of the selected relevant users are neighbors of the central user, although the evaluation metrics are relatively high, the significance of prediction is lost. If *k* is too large, the model becomes sensitive to noise nodes among the relevant users, and the complexity of the model will increase accordingly.

5.2.3. Analysis of feature representation

Then, the impact of representation learning algorithms is verified. The influence of different representation learning algorithms on link prediction is shown in Fig. 7. The *x*-axis represents the embedding dimensions of four algorithms, and

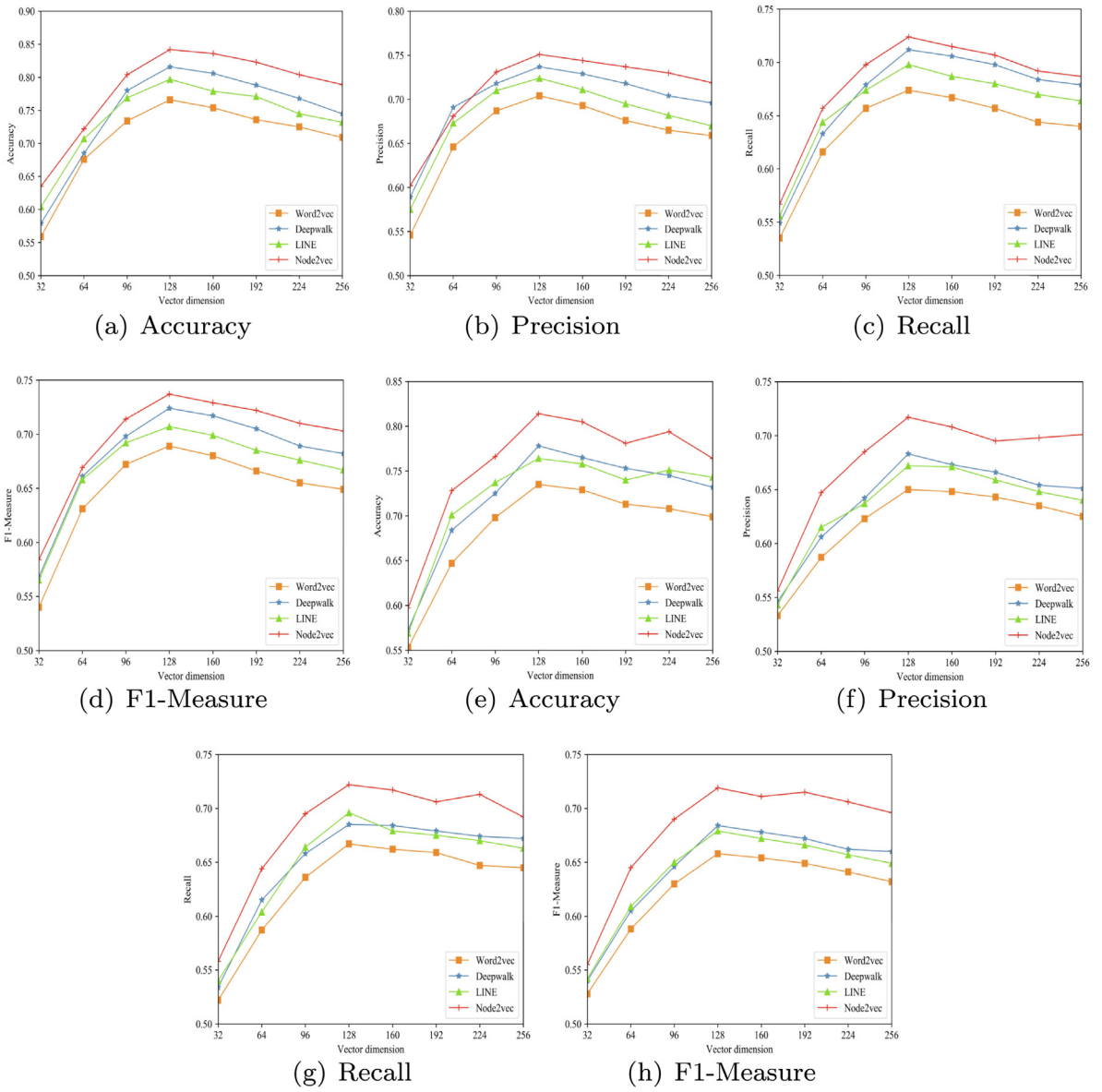


Fig. 7. Comparison of embedded dimensions of different network embedding. (a)–(d) are the results in Sina Weibo; (e)–(h) are the results in Twitter.

the *y*-axis is the values of four metrics: accuracy, precision, recall and F1-Measure. By comparing different vector dimensions, the effectiveness of node2vec algorithm can be verified.

In Fig. 7, we find that node2vec method is the best. Compared with Word2vec, Deepwalk and LINE, the feature representation obtained by node2vec has a greater impact on prediction performance in two data sets. By comparing the accuracy, precision, recall and F1-Measure of node2vec and other three methods, it can be seen that when the feature dimension is around 128, the prediction results are optimal. When the dimension is too low, the learned vector cannot accurately represent the network structure features. Increasing the dimension of the learned vectors will express their features more accurately, but that leads to the time consumption increased greatly. When the vector dimension reached 128, the performance tends to saturate. Based on the above analysis, the algorithm used in this paper works best when the dimension is 128.

5.2.4. Analysis of feature contribution

Furthermore, the impact of single and multiple features on link prediction is verified. As Twitter data set only contains network structure, this experiment only uses Sina Weibo. By setting the input form of single feature and multi-feature, we construct two submodels and one fusion model: Sub-NF, Sub-TF and Fusion model. By comparing with these submodels,

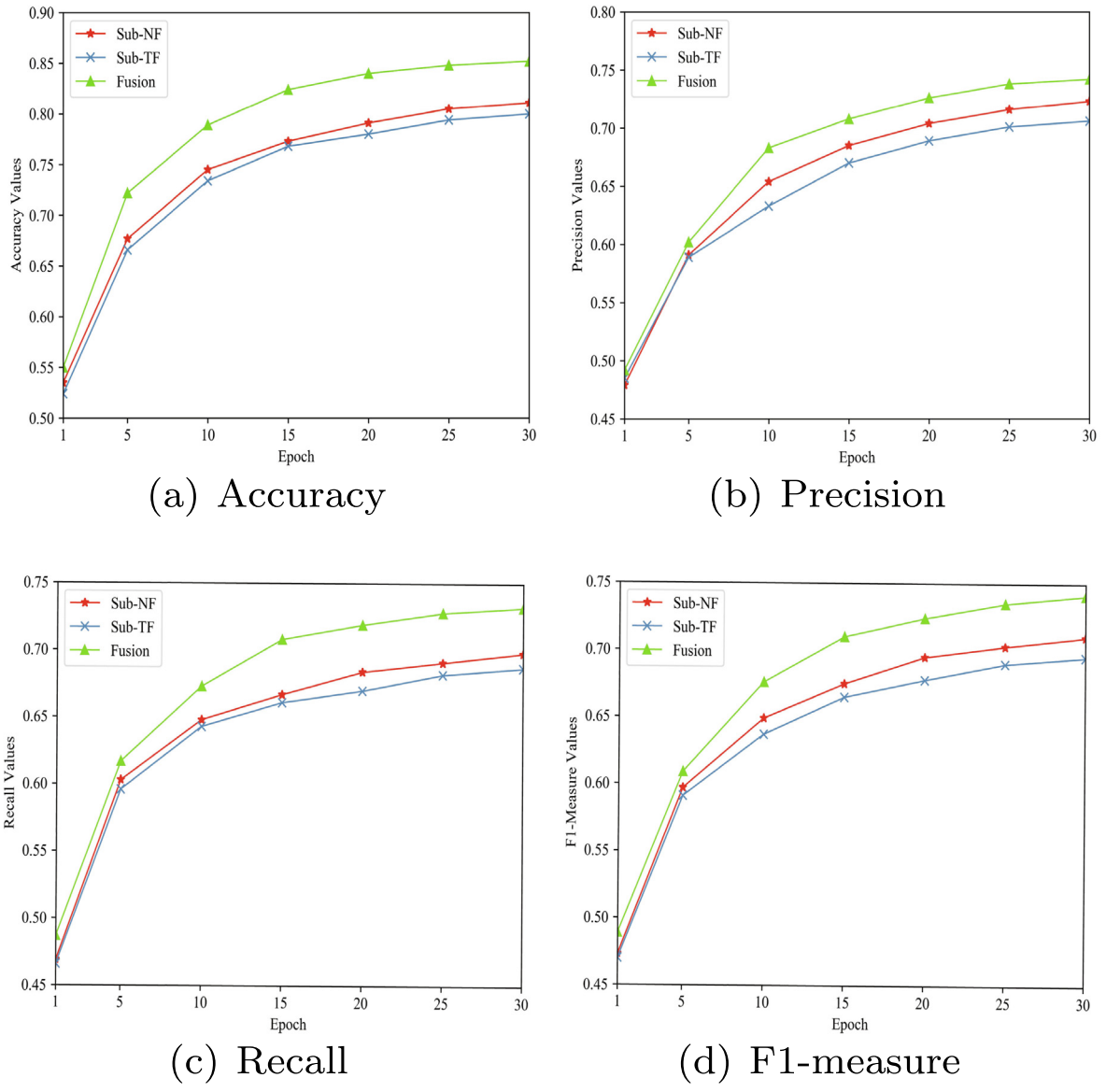


Fig. 8. Comparison of prediction results between submodels and fusion model in Sina Weibo.

the relationships among the number of training iterations of the network and the performances of this model are shown in Fig. 8. The *x*-axis represents the number of epoch, and the *y*-axis represents the values of four metrics: accuracy, precision, recall, and F1-Measure.

As shown in Fig. 8, with the increase of training epoch, the values of evaluation metrics increase first and then decrease of the three models, and compared with the performance of submodels, the fusion feature is better than the single features. It means that combining multiple features effectively improved the prediction results. At the same time, it can also be seen from Fig. 8 that the network structure feature has a greater impact on link establishment than the user text feature. In addition, the number of epoch also reflects the convergence trend of the model, and the prediction trends in Fig. 8 also follow those of the experimental results in Section 5.2.1.

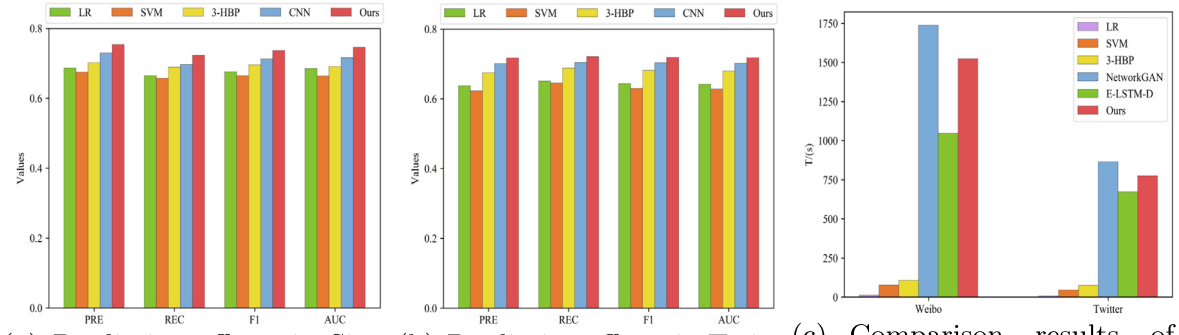
5.2.5. Comparison with other methods

Finally, the performance of the proposed method was compared with those of various baseline methods. In this section, some classical link prediction methods are chosen to compare: RA, AA, PA, and JC. The performances of them are shown in Table 1. Meanwhile, we also choose LR and SVM, which are two better classifiers, and 3-HBP for link prediction. These performances of these two data sets are shown in Fig. 9(a) and (b). As can be seen from the two figures, the experimental performances on the Sina Weibo data set are better than that on the Twitter data set, which proves that the model incorporating user text is effective for link prediction. The *x*-axis in Fig. 9(a) and (b) represents four metrics: precision, recall, F1-Measure and AUC, and the *y*-axis represents the values of four metrics.

Table 1 indicates that the proposed method has the best performance. Compared with the basic CNN model, we can find that the improvement of accuracy is 3.1%, precision is 2.4%, recall is 2.5%, F1-Measure is 2.6% and AUC is 3.0% of our method. Comparing the precision, recall, F1-Measure, and AUC in Fig. 9, our method outperformed the binary classification methods and the basic CNN model. Moreover, we compare the running time of these methods, as shown in Fig. 9(c). These methods are implemented using python 3.6, the cpu running environment is 2.6 GHZ, memory is 8 GB, hard disk is 500 GB, system is 64 bit Windows 7. As shown in Fig. 9(c), the running time of our method is between the other two neural network algorithms. Based on the experimental results, our model based on representation learning and feature fusion can effectively improve the performance of link prediction.

Table 1
Comparison of different methods.

Method	Accuracy	Precision	Recall	F1-Measure	AUC
AA	0.598	0.570	0.549	0.560	0.569
RA	0.631	0.595	0.557	0.574	0.574
PA	0.585	0.549	0.541	0.542	0.538
JC	0.613	0.587	0.554	0.570	0.575
CNN	0.821	0.730	0.707	0.717	0.716
Ours	0.852	0.754	0.732	0.743	0.746



(a) Prediction effects in Sina Weibo (b) Prediction effects in Twitter (c) Comparison results of computational complexity

Fig. 9. Comparison of different methods.

6. Conclusion

This paper proposed a link prediction model based on network embedding, which combines multiple information. In this model, network structure feature and user text feature are used to predict links. According to different features, we extract the correlations among users by different representation learning algorithms, and select only the top-k users of each node to simplify the complexity of model. Finally, a attention layer and a fusion layer are added to the convolutional neural network. This architecture predicts the links by combining the network structure and user text. The model was experimentally validated on real datasets. The experimental results show that both the network structure and user content are important for link prediction. In addition, the proposed method also improves the performance of link prediction. In the future work, we will consider incorporating additional information, such as the user attributes and image information published by users, and focus on the applications of link prediction.

CRediT authorship contribution statement

Rui Li: Writing - original draft.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This paper is partially supported by the National Natural Science Foundation of China (Grant No. 61772098, 62006032, 62072066); Chongqing Graduate Education Teaching Reform Project (Grant No. yjg183081); Science and Technology Research Program of Chongqing Municipal Education Commission (Grant No. KJZD-K201900603, KJQN201900629) and Chongqing Graduate Scientific Research and Innovation Project (Grant No. cys19263).

References

[1] C. Ahmed, A. ElKorany, Enhancing link prediction in twitter using semantic user attributes, in: 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), IEEE, 2015, pp. 1155–1161.
[2] Z. Bu, Y. Wang, H.J. Li, J. Jiang, Z. Wu, J. Cao, Link prediction in temporal networks: Integrating survival analysis and game theory, Information Sciences 498 (2019) 41–61.
[3] E. Bütün, M. Kaya, R. Alhajj, Extension of neighbor-based link prediction methods for directed, weighted and temporal social networks, Information Sciences 463 (2018) 152–165.
[4] H. Cai, V.W. Zheng, K.C.C. Chang, A comprehensive survey of graph embedding: problems, techniques, and applications, IEEE Transactions on Knowledge and Data Engineering 30 (2018) 1616–1637.
[5] Y. Cao, S. Wang, X. Li, C. Cao, Y. Liu, J. Tan, Inferring social network user's interest based on convolutional neural network, in: International Conference on Neural Information Processing, Springer, 2017, pp. 657–666.
[6] J. Chen, J. Zhang, X. Xu, C. Fu, D. Zhang, Q. Zhang, Q. Xuan, E-lstm-d: A deep learning framework for dynamic network link prediction, IEEE Transactions on Systems, Man, and Cybernetics: Systems (Early Access), 2019.
[7] Y.L. Chen, C.H. Chuang, Y.T. Chiu, Community detection based on social interactions in a social network, Journal of the Association for Information Science and Technology 65 (2014) 539–550.
[8] M. Coskun, M. Koyutürk, Link prediction in large networks by comparing the global view of nodes in the network, in: 2015 IEEE International Conference on Data Mining Workshop (ICDMW), IEEE, 2015, pp. 485–492.
[9] P. Cui, X. Wang, J. Pei, W. Zhu, A survey on network embedding, IEEE Transactions on Knowledge and Data Engineering 31 (2018) 833–852.
[10] A. De, S. Bhattacharya, S. Sarkar, N. Ganguly, S. Chakrabarti, Discriminative link prediction using local, community, and global signals, IEEE Transactions on Knowledge and Data Engineering 28 (2016) 2057–2070.
[11] N.Z. Gong, B. Liu, You are who you know and how you behave: attribute inference attacks via users' social friends and behaviors, in: 25th USENIX Security Symposium (USENIX Security 16), 2016, pp. 979–995.
[12] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 855–864.
[13] W. Gu, F. Gao, X. Lou, J. Zhang, Link prediction via graph attention network, 2019, arXiv preprint arXiv:1910.04807.
[14] H. Hu, Y. Wen, T.S. Chua, J. Huang, W. Zhu, X. Li, Joint content replication and request routing for social video distribution over cloud cdn: a community clustering method, IEEE Transactions on Circuits and Systems for Video Technology 26 (2015) 1320–1333.
[15] B. Jiang, J. Liang, Y. Sha, R. Li, L. Wang, Domain dictionary-based topic modeling for social text, in: International Conference on Web Information Systems Engineering, Springer, 2016, pp. 109–123.
[16] M.M. Keikha, M. Rahgozar, M. Asadpour, Deeplink: A novel link prediction framework based on deep learning. 2018, arXiv preprint arXiv:1807.10494.
[17] S. Kopeinik, E. Lex, P. Seitlinger, D. Albert, T. Ley, Supporting collaborative learning with tag recommendations: a real-world study in an inquiry-based classroom project, in: Proceedings of the Seventh International Learning Analytics & Knowledge Conference, ACM, 2017, pp. 409–418.
[18] Q. Le, T. Mikolov, Distributed representations of sentences and documents, in: International Conference on Machine Learning, 2014, pp. 1188–1196.
[19] X. Lei, X. Yang, H. Fujita, Random walk based method to identify essential proteins by integrating network topology and biological characteristics, Knowledge-Based Systems 167 (2019) 53–67.
[20] C.K. Leung, F. Jiang, T.W. Poon, P.É. Crevier, Big data analytics of social network data: who cares most about you on facebook?, in: Highlighting the Importance of Big Data Management and Analysis for Various Applications, Springer, 2018, pp 1–15.
[21] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, H. Liu, Attributed network embedding for learning in a dynamic environment, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, ACM, 2017, pp. 387–396.
[22] L. Li, J. He, M. Wang, X. Wu, Trust agent-based behavior induction in social networks, IEEE Intelligent Systems 31 (2016) 24–30.

[23] Z.L. Li, X. Fang, O.R.L. Sheng, A survey of link recommendation for social networks: methods, theoretical foundations, and future research directions, ACM Transactions on Management Information Systems (TMIS) 9 (2018) 1.
[24] V. Martínez, F. Berzal, J.C. Cubero, A survey of link prediction in complex networks, ACM Computing Surveys (CSUR) 49 (2017) 69.
[25] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, 2013, arXiv preprint arXiv:1301.3781.
[26] X. Pan, Y.X. Fan, J. Jia, H.B. Shen, Identifying rna-binding proteins using multi-label deep learning, Science China Information Sciences 62 (2019) 19103.
[27] M. Peng, J. Zhu, H. Wang, X. Li, Y. Zhang, X. Zhang, G. Tian, Mining event-oriented topics in microblog stream with unsupervised multi-view hierarchical embedding, ACM Transactions on Knowledge Discovery from Data (TKDD) 12 (2018) 38.
[28] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014, pp. 701–710.
[29] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M.P. Reyes, M.L. Shyu, S.C. Chen, S. Iyengar, A survey on deep learning: algorithms, techniques, and applications, ACM Computing Surveys (CSUR) 51 (2019) 92.
[30] A. Shahmohammadi, E. Khadangi, A. Bagheri, Presenting new collaborative link prediction methods for activity recommendation in facebook, Neurocomputing 210 (2016) 217–226.
[31] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: large-scale information network embedding, in: Proceedings of the 24th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.
[32] C. Tu, H. Liu, Z. Liu, M. Sun, Cane: Context-aware network embedding for relation modeling, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2017, pp. 1722–1731.
[33] C. Tu, W. Zhang, Z. Liu, M. Sun, et al., Max-margin deepwalk: discriminative learning of network representation, in: IJCAI, 2016, pp. 3889–3895.
[34] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 1225–1234.
[35] P. Wang, B. Xu, Y. Wu, X. Zhou, Link prediction in social networks: the state-of-the-art, Science China Information Sciences 58 (2015) 1–38.
[36] Z. Wang, J. Liang, R. Li, Exploiting user-to-user topic inclusion degree for link prediction in social-information networks, Expert Systems with Applications 108 (2018) 143–158.
[37] L. Wu, Y. Ge, Q. Liu, E. Chen, R. Hong, J. Du, M. Wang, Modeling the evolution of users' preferences and social links in social networking services, IEEE Transactions on Knowledge and Data Engineering 29 (2017) 1240–1253.
[38] Y. Xiao, X. Li, H. Wang, M. Xu, Y. Liu, 3-hbp: A three-level hidden bayesian link prediction model in social networks, IEEE Transactions on Computational Social Systems 5 (2018) 430–443.
[39] M. Yang, J. Liu, L. Chen, Z. Zhao, X. Chen, Y. Shen, An advanced deep generative framework for temporal link prediction in dynamic networks, IEEE Transactions on Cybernetics (Early Access) (2019).
[40] M. Zhang, Y. Chen, Link prediction based on graph neural networks, in: Advances in Neural Information Processing Systems, 2018, pp. 5165–5175.
[41] Q. Zhang, Y. Gong, J. Wu, H. Huang, X. Huang, Retweet prediction with attention-based deep neural network, in: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, ACM, 2016, pp. 75–84.
[42] Z. Zhang, H. Yang, J. Bu, S. Zhou, P. Yu, J. Zhang, M. Ester, C. Wang, Anrl: Attributed network representation learning via deep neural networks, in: IJCAI, 2018, pp. 3155–3161.
[43] Z. Zhao, Y. Wu, Attention-based convolutional neural networks for sentence classification, in: INTERSPEECH, 2016, pp. 705–709.
[44] Y. Zhou, C. Huang, Q. Hu, J. Zhu, Y. Tang, Personalized learning full-path recommendation model based on lstm neural networks, Information Sciences 444 (2018) 135–152.