Combination of Structural and Attributed Phased DeepWalk Method for Link Prediction

Zilong Wang, Zhehao Zhang, Juntu Zhao

Abstract—Graph data mining is a very important branch of data mining. Many things in our life can be described by a graph structure, such as our social network and paper citations network. Therefore, the study of graph data is particularly important. Link prediction is a classic task in network science, which is used to predict which new links will appear in the network using the currently obtained network. However, networks with only structural similarity are not practical in our real life. On the contrary, networks with more information (attribute) about the node are widely used such as social networks. To better utilize the attribute information, we propose a new algorithm called CSAPDW for link prediction in attributed networks by making use of both the structural similarity and attribute similarity of the network. Efficient experiment results indicate that our methods perform better than others.

Index Terms—Link prediction, Attributed network, node similarity, random walk

1 Introduction

Complex networks can be used to describe many natural phenomena, such as our social network and paper citation network, etc. Nowadays graph network technology has developed into a very common and important field, in which we have encountered many important challenges such as Community detection, link prediction, etc. Link prediction is one of the very important topics. It uses past data to predict the future structure of a complex network, that is, existing unknown states and possible future states can be estimated through link prediction.

The methods used in link prediction can be mainly divided into three types: similarity-based prediction, model-based prediction, and embedded technology. Among them we use embedded technology that is a very efficient method for solving complex network analysis tasks, which maps the network to a low-dimensional space, preserving the desired main feature information. However, the embedded-based link prediction methods we have learned before mainly focus on the structural information of the graph. While in real scenarios, nodes in the network not only have structural features related to other nodes but also have their own feature information.

For example, in the classmate information table, each classmate has his own gender, student number, and dormitory number, this information should obviously also be used as important content in the interpersonal network. Through the above example, we can find that just like using the structural features of nodes, using the own characteristics of network nodes can also help improve the accuracy of link prediction.

In order to solve the above problems, we modify the deep walk algorithm random walk strategy in the attribute network. Because the original deep walk algorithm only uses the structural information of the network but does not use the point's own feature information (it is called attribute information in the attribute network). This paper uses a linear time complexity algorithm, uses local structural features and attribute features, maps the network to a low-dimensional space, and obtains node features for subsequent prediction.

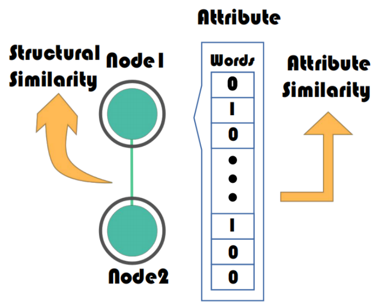

Fig. 1. Structural and attribute similarity between two nodes

At the same time, in order to further improve the accuracy of the algorithm, we also use the prediction results of each time to make loop predictions. Specific details will be described later.

In this project, we have used a lot of techniques in data mining such as node embedding and computation of similarity. Besides, graph mining is an important part of data mining. (**This paragraph is to show the relationship between our project and data mining**)

The rest of the paper is structured as follows: In Sect. 2, the introduction of related work on link prediction. In Sect. 3, The specific introduction and implementation of our algorithm. In Sect. 4, our experimental results. In Sect.5, the discussion, and in Sect.6, our conclusions.

2 Related Work

This section briefly introduces related research on link prediction and our work.

2.1 Similarity-based methods and Probabilistic model-based methods

First, similarity-based methods are the most common link prediction methods. By assigning similarity to each pair of nodes, we can judge the probability that there is an edge between them. Features used here include local topological features obtained using topological structures (for example some most commonly used similarity measure for link prediction, i.e. common neighbors[1] and its extensions, e.g. Adamic/Adar [2]) and global structure information obtained by path-based. Similarity-based methods perform very easily because we do not need any learning. While it will bring lower performance.

The second link prediction method is probabilistic model-based methods, which try to learn the best model to describe our network. The basic idea is to establish a set of parameters Θ based on the current network structure and existing links, and then use conditional probability $P(e_{ij}|\Theta)$ to calculate the probability of a connection between a pair of nodes. The relational Bayesian Networks[3] and relational Markov Networks[4] as we know them are classical examples that fall into this category. But the biggest challenge with this method is getting the right set of parameters to get the best results.

2.2 Embedding-based methods

Finally, one of the most popular methods in link prediction is the Embedding-based method. As we have mentioned before, this method is to map the information features of the graph structure to a low-dimensional space and preserve the important information in the graph structure as much as possible.

DeepWalk[5] and Node2Vec[6] are popular ones in Embedding-based methods. First, they represent the graph as a sequence of nodes produced by random walks. Then they introduce a method called Skip-Gram[7] in natural language processing to maximize the co-occurrence probability of the nodes in the w-sized window in a random walk. And the difference between the two methods is that DeepWalk uses a pure random walk, while Node2Vec uses a biased random walk.

In previous Data Mining courses, we have learned relevant basic knowledge of link prediction. Also, as an exercise, we manually implemented a link prediction algorithm, in which we introduce the idea of word2vec in NLP to convert nodes in the graph into a vector describing its features for subsequent operations. As we can see, the process of word embedding is crucial. So we hope to improve this process based on the exiting link prediction algorithm, so as to enhance our prediction accuracy.

As we mentioned before, the embedding method we used earlier only focused on the structural features of the graph. The method used in this paper is to use the attributes of nodes and structural features of the graph as input to learning graph embedding through random walks.

3 METHODOLOGY

3.1 Definition

Before introducing our algorithm, we first introduce some concepts.

3.1.1 Definition 1

[Structural similarity (sim_structure)] Research has shown that In a graph network, the manual neighbors of two nodes are crucial in judging the relationship between two nodes. And this connection decreases as the distance between the points increases. Hence, when calculating the structural characteristics of nodes, only the first and second-degree neighbors are considered. The specific definition is as follows:

$$ss(v_i, v_j|\alpha) = A_{ij}\frac{2|Cover(v_i,\alpha) \cap Cover(v_j,\alpha)|}{|Cover(v_i,\alpha)| + |Cover(v_j,\alpha)|} \tag{1}$$

In wich A is the adjacency matrix of our graph, and $Cover(v_i,\alpha) = \{v_j \in v|dist(v_i,v_j) \leq \alpha\}$ where α is the acceptable depth of common neighbor network. In our experiment, its set to be 2.

3.1.2 Definition 2

[Attribute similarity(sim_attribute)] Studies have shown that people are more attractive to those with similar characteristics. In our graph structure, it is also obviously true. Our attribute similarity is completely independent of the topology of the entire network. Here we use Sorensen-Dice for feature criterion[8]to calculate the similarity of nodes' attribute feature. The Sorensen-Dice similarity here is a combination of total attribute and manual attribute between two nodes:

$$as(v_i, v_j) = \frac{|I(v_i) \cap I(v_j)|}{|I(v_i) \cup I(v_j)|} \tag{2}$$

in which I is the collection of a node's attribute features.

Instead of using the standard Sorensen-Dice similarity, we use TF-IDF (term frequency-inverse document frequency) to solve the problem of sparse feature, which is a trick widely used in natural language processing. The inverse document frequency (IDF) of the word is defined as:

$$IDF(i) = \log \frac{\text{number of nodes}}{\text{number of nodes in which i appears}} \tag{3}$$

When we compute the attribute similarity, we weight the same features by its IDF.

3.1.3 Definition 3

Consider our attributed graph $G = (V, E, F)$, we transform the original graph into a weighted graph by introducing structural features and node attribute features, where the weights of the edges in the graph are determined by the following equation:

$$w(v_i, v_j) = A(v_i, v_j)[\gamma \times ss(v_i, v_j) + (1-\gamma) \times as(v_i, v_j)] \tag{4}$$

where $a \in [0, 1]$ is a hyper-parameter that adjusts the balance between the structural features and attributed features.

3.2 Algorithm

3.2.1 CSAPDW

According to the definitions above, our Phased DeepWalk algorithm on the attribute graph is implemented on the basis of the traditional DeepWalk algorithm, as shown in the following CSAPDW algorithm:

Algorithm 1 CSAPDW

```
Input:
    Network $G = (V, E)$, Attribute Matrix $A$, Walk-per-vertex $T$, Walk Length $L$, Number of Phase $\rho$, parameter $\gamma$ and $\beta$.
Output: Embedding Vector of Nodes
 1: Compute MSS based on graph structure $G$.
 2: Compute MAS based on $A$ and $G$
 3: $W = \gamma MAS + (1 - \gamma)MSS$.
 4: Initialize $trajectory = [\,]$
 5: for $t = 1, \cdots, T$ do
 6:     for node in $V$ do
 7:         Initialize $rw = [node]$
 8:         for $l = 1, \cdots, L$ do
 9:             current node = $rw[-1]$
10:             neighbor nodes = Get_ Neighbor(current node)
11:             Select next node $n$ from neighbor nodes, the probability is proportional to $W$.
12:             Append $n$ to $rw$.
13:         end for
14:         Append $rw$ to $trajectory$.
15:     end for
16: end for
17: $f = skip - gram(V, d, trajectory)$.
18: for $l = 1, \cdots, \rho$ do
19:     for $(u, v)$ in $G.edges$ do
20:         Compute cosine similarity $p$ between $f(u)$ and $f(v)$.
21:         $P(u, v) = p$.
22:     end for
23:     $W = \beta P + (1 - \beta)(\gamma MAS + (1 - \gamma)MSS)$.
24:     Repeat step 4 to 17.
25: end for
26: return $f$.
```

In our algorithm, first, we calculate attribute similarity and structure similarity for the input graph G, and then get our weight matrix according to the calculated features. As for the following random walk process, the next node is selected according to the weight ratio of the edges connected to the current node. Next, we use the node string obtained in the previous step to learn the feature vector of each node according to the word2vec method [6], and then calculate the existence probability of each pair of edges.

3.2.2 Cycle prediction

After the above calculations, we can get the probability that there is an edge between two nodes. Next, we propose an algorithm for cycle prediction: After calculating the probability of the existence After the above calculations, we can get the probability that there is an edge between two nodes. Next, we propose an algorithm for cycle prediction: After calculating the probability of the existence of each edge according to the network, we believe that this probability can better represent the close relationship between two points. There, for any point pair(v_i, v_j), we've got the probability p_{ij} that there is an edge between them. If there is indeed an edge between them in the given graph G, we just set the weight w_{ij} on the edge (v_i, v_j) to p_{ij}, instead of the weight previously calculated based on attribute features and structural features. This way we get a new weighted graph. Next, we re-run the Deepwalk algorithm on this graph and repeat the above steps. of each edge according to the network, we believe that this probability can better represent the close relationship between two points. There, for any point pair(v_i, v_j), we've got the probability p_{ij} that there is an edge between them. If there is indeed an edge between them in the given graph G, we just set the weight w_{ij} on the edge (v_i, v_j) to p_{ij}, instead of the weight previously calculated based on attribute features and structural features. This way we get a new weighted graph. Next, we re-run the Deepwalk algorithm on this graph and repeat the above steps.

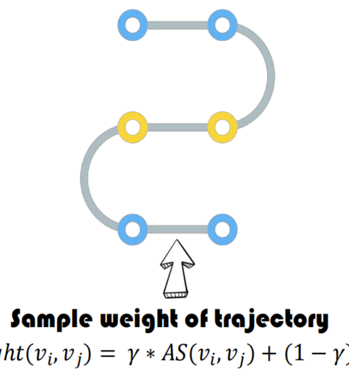
Sample weight of trajectory
$weight(v_i, v_j) = \gamma * AS(v_i, v_j) + (1 - \gamma) * SS(v_i, v_j)$

Fig. 2. The sample weight of two different nodes is defined according to both their attribute similarity and structural similarity.

4 Experiment Results

In our project, we conduct 4 classical methods on 4 different datasets including Cora[9], Cornell, Texas, Wisconsin. Cora is a citation network in which nodes present machine learning papers and an edge between two papers is formed only if one of them is cited by the other. Keywords in papers are treated as node attributes. It consists of 2708 nodes and 5278 edges. Cornell, Texas, and Wisconsin are subnetworks drawn from the WebKB dataset. Each of them contains webpages as nodes and links connecting them. Cornell has 195 nodes and 304 edges. Texas consists of 187 nodes and 328 edges. Wisconsin contains 265 nodes and 530 edges.

Deepwalk, Node2vec, and Line[10] are our baselines. In the experiment, we randomly reserve 50% of the edges as the test set for every method and dataset. After training, we use AUC as a metric to evaluate our methods. The epoch of training is 10 for every phase. We use the same optimizer (Adam with the learning rate of 0.01) for every method and dataset. The learning rate will decay with the factor of 0.95 exponentially. We set the embedding size 128 and $\gamma = 0.82$ in equation 4. For Cora datasets, we sample a trajectory of 20 nodes with additional 20 negative samples and for other datasets, we sample 8 nodes with 1 negative sample.

The full experiment results can be found in Table1 and Figure3.

TABLE 1
Comparison with different methods on different datasets

Dataset	Cora	Cornell	Texas	Wisconsin
Deepwalk	0.73	0.44	0.60	0.70
Node2vec	0.74	0.44	**0.61**	0.68
Line	0.57	0.48	0.54	0.46
CSAPDW	**0.76**	**0.60**	0.60	**0.71**

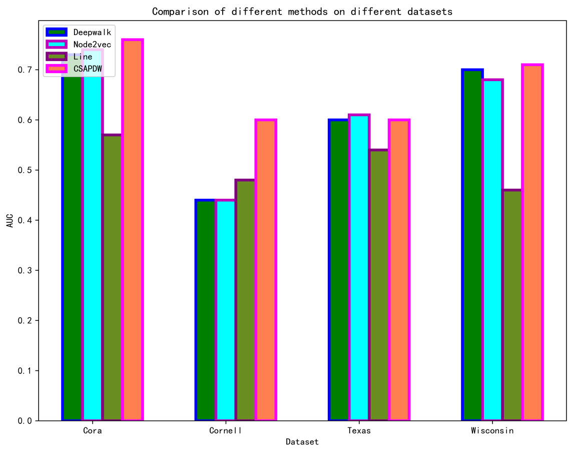

Fig. 3. Comparison of different methods on different datasets

5 Discussion

In this section, we give some remarks on our CSAPDW algorithm for link prediction.

Comparison with the traditional DeepWalk: Our CSAPDW algorithm is a modification of DeepWalk, which considers both structural similarity and attribute similarity. In addition, we use the results predicted in the last phase to tune the transition matrix in order to make the biased random walk process more reliable. These improvements make sure that CSAPDW would perform better than the traditional DeepWalk method. Extensive experiments also show the advantage of our methods.

Restriction on Attribute: As we can see, the attribute similarity is equal to the Jaccard similarity between two nodes. Unfortunately, this results in attributes that can only be binary and discrete. Thus we can extend it to a more general attribute setting if we could find another metric to measure the similarity of attributes.

Choice of parameters: There are several hyper-parameters in our CSAPDW. γ and β are parameters that do not exist in traditional DeepWalk. γ balances the contribution between structural similarity and attribute similarity, while β balances the result in the last phase and the origin weight matrix. The choice of these two parameters depends on the network structure and Characterization ability of attributes.

The ratio left for validation: In the experiment, for a fair comparison, we reserve 50% of the edges for testing on every dataset. However, the Texas dataset only contains a small number of nodes and edges. It is obvious that the ratio of 50% is quite large for this dataset.

6 Conclusion

In this project, we have surveyed several methods in the field of link prediction. Then, we observe that the traditional link prediction task with only nodes' structural similarity is not very practical. It is more common to use attributed networks in our real-life such as social networks. After that, we propose a new algorithm called CSAPDW for link prediction in attributed networks. The transition matrix is computed by making use of the structural similarity and attribute similarity of the network. Moreover, we also propose a phase method, which changes the transition matrix based on the result in the last phase. Compared with other methods of link prediction, CSAPDW performs the best in several datasets.

References

[1] Emily, M., Jin, Michelle, Girvan, M., E., J., and Newman, "Structure of growing social networks," *Physical Review E*, vol. 64, no. 4, pp. 46 132–46 132, 2001.
[2] Lada, A, Adamic, Eytan, and Adar, "Friends and neighbors on the web," *Social Networks*, 2003.
[3] P. Sarkar and A. Moore, "A tractable approach to finding closest truncated-commute-time neighbors in large graphs," *Proc Uai*, 2012.
[4] H. Kashima and N. Abe, "A parameterized probabilistic model of network evolution for supervised link prediction," pp. 340–349, 2006.
[5] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," *ACM*, 2014.
[6] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," *the 22nd ACM SIGKDD International Conference*, 2016.
[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *Computer Science*, 2013.
[8] E. Faizal, "Case-based reasoning untuk mendiagnosa penyakit cardiovascular dengan metode weighted minkowski," *Universitas Gadjah Mada*, 2013.
[9] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, p. 93, Sep. 2008. DOI: 10.1609/aimag.v29i3.2157. [Online]. Available: https://ojs.aaai.org/index.php/aimagazine/article/view/2157.
[10] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15, Florence, Italy: International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077, ISBN: 9781450334693. DOI: 10.1145/2736277.2741093. [Online]. Available: https://doi.org/10.1145/2736277.2741093.