Using DeepWalk for Node classification and link prediction in academic network

Eduardo Wang Zheng,Feng Zeng,Yuan Li

Abstract—Deepwalk is a basic algorithm for network representation learning.In this reserch, we intend to use it to solve the link prediction and node classification problems in academic networks.

Index Terms—DeepWalk,Node classification,link prediction

1 Introduction

The node classification task is one where the algorithm has to determine the labelling of samples (represented as nodes) by looking at the labels of their neighbours. Link Prediction in a network is how to predict the likelihood of a link between two nodes in a network that have not yet produced a link, such as known network nodes and network structures. These two methods have played an important role in network science.

Deepwalk is a basic algorithm for network representation learning, which is used to learn the vector representation of vertices in the network.It can be used to solve link prediction and node classification problem.

In this project, we intend to solve the following two academic network problems using Deepwalk algorithm

1)Judge which of the 10 meetings each scholar has published in 2016-2019

2)Determine whether two scholars have collaborated in publishing articles in 2020

2 Methods

2.1 Deepwalk

The algorithm consists of two main steps: the first step uses the Random Walk algorithm to sample the node sequence, and the second step uses the skip-gram algorithm to learn the expression vector.

2.1.1 Random Walk

The basic idea of the random walk algorithm is to traverse a graph from one or a series of vertices. At any vertex, the traversal will walk to the neighbor vertex of this vertex with probability 1-a, and randomly jump to any vertex in the graph with probability a. We call a the probability of jump occurrence, which is obtained after each walk A probability distribution that describes the probability of each vertex in the graph being visited. Use this probability distribution as the input for the next walk and iterate this process repeatedly. When certain preconditions are met, this probability distribution tends to converge. After convergence, a stable probability distribution can be obtained.The randomwalk algorithm is as follows.Let f(x) be a multivariate function with n variables, and x=(x1,x2,...,xn) is an n-dimensional vector.

1. Given the initial iteration point x, the first walking step length l, the control accuracy err (err is a very small positive number, used to control the end of the algorithm).

2. Given the number of iteration control N, k is the current iteration number, set k=1.

3. When k<N, randomly generate an n-dimensional vector between (−1,1) u=(u1,u2,...,un),(−1<ui<1,i=1,2,...,n), and standardize it to get u =u/ ui^2.$Letx1 = x + lutocompletethefirststepofwalking.$

4. Calculate the value of the function. If f(x1)<f(x), that is, a better point than the initial value is found, then k is reset to 1, x1 is changed to x, and step 2 is returned; otherwise, k= k+1, go back to step 3.

5. If no better value can be found for N consecutive times, it is considered that the optimal solution is in the N-dimensional sphere with the current optimal solution as the center and the current step size as the radius (if it is three-dimensional, it is just the space Sphere in). At this point, if l<err, the algorithm ends; otherwise, let l=l2, go back to step 1, and start a new round of random walk.

- .

2.2 Skip-Gram

The neural network model of skip-gram is improved from the feed-forward neural network model, which is based on the feed-forward neural network model, through some techniques to make the model more effective. The following picture shows the neural network model of skip-gram:

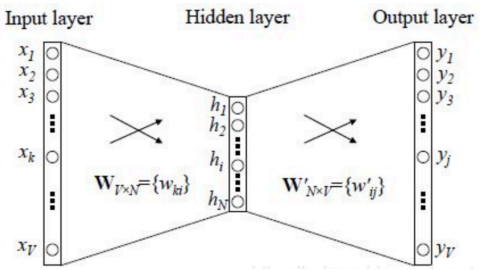

Fig. 1. skipgrammodel

In the above figure, the input vector x indicates the one-hot encoding of a word, and the corresponding output vector y1 ,..., yC . The ith row of the weight matrix W between the input layer and the hidden layer indicates the weight of the ith word in the vocabulary. Next comes the key point: this weight matrix W is the goal we need to learn (same as W), because this weight matrix contains the weight information of all words in the vocabulary. In the above model, each output word vector also has an N×V dimensional output vector W . Finally, the model has a hidden layer of N nodes. We can find that the input of the hidden layer node hi is the weighted sum of the inputs of the input layer. So because the input vector x is one-hot coded, only non-zero elements in the vector can generate input to the hidden layer. So for the input vector x, where xk=1 and xk'=0, k!=k'. Finally, we use the softmax function to generate the polynomial distribution of the C-th word as folliwing:p(wc,j=wO,c|wI)=yc,j=exp(uc,j)/ exp(uj).This value is the probability size of the j-th node of the C-th output word.

3 Experiments

3.1 Link prediction

First, we analyze the data set to get the approximate relationship between the authors and the articles. The relationship between the nodes is roughly as Fig2.Next we enter the data and build the network.Read in auth-or-paper-all-with-year.csv , labeled-papers-with-authors.csv and paper-reference.csv.Treat each author as a node, and construct edges between nodes based on whether they have published articles in cooperation or not.So that we can construct a homogeneous network G .Then,we can capture the random walk sequences for all nodes in the graph G.After extracting random walk sequences, we need to use the data to train the word2vec model and skip-gram (word2vec) model.The parameter settings of the model are shown in Figure 3.With this skipgram model, we can use it to predict links and predict whether scholars will collaborate to publish papers this year.

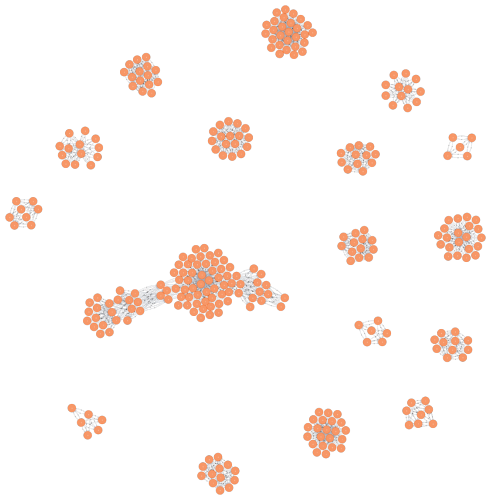

Fig. 2. data

```
model = Word2Vec(window = 4, sg = 1, hs = 0,
                 negative = 10, # for negative sampling
                 alpha=0.03, min_alpha=0.0007,
                 seed = 14)

model.build_vocab(random_walks, progress_per=2)

model.train(random_walks, total_examples = model.corpus_count, epochs=20, report_delay=1)
```

Fig. 3. model parameter

3.2 Node classification

Node classification is to train a model to learn in which class a node belongs. Here a node means a paper and the class means some conference. This is a common machine learning task, and is used to predict a non-existing node property based on other node properties.

We will still use the model trained in previous section, applying the DeepWalk network. Our goal is to predict at which conference some paper had published, with some rows of paper whose publish place is already known. In this sense, we will read the data in this form:

Read authors_to_pred.csv . This is the node to be predicted. We read every row and store the elements as a

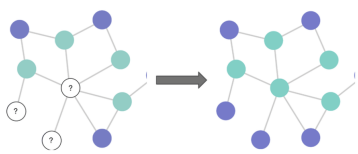

Fig. 4. Node classification

```
links=model.similar_by_word(str(node_prediction[i]))
for link in links:
    flag=0
    if (link[0]!=-1):
        for author in authors:
            if (author.author_id==link[0]):
                if (author.label!='100'):
                    flag=1
                    node_label.append(author)
                    break
if (flag==0):
    node_label.append(fake)
```

Fig. 5. Utilize Word2Vec for result

one dimensional vector. This is going to be the input of our model.

With every exactly known papers sorted in previous part, we store the existing paper-conference relation in the node_lable vector. Then we use this information to obtain result through the similar_by_word method of the Word2Vec network.

4 Results

	node classification	link prediction
score/AUC	0.19454	0.50075

Fig. 6. result

5 Conclusion

Judging from the results, the mission was completed in its entirety. But the final result actually has a lot of room for improvement. When composing and modeling, we can actually take more factors into consideration to get a better model, so that the predicted results may be more accurate. In addition, the effect of using Deepwalk for both link prediction and node classification does not seem to be good. The score of node classification is only 0.19. If you want to further improve, you can try another method, and you should get a better effect.

6 Contributions

6.1 Eduardo Wang Zheng

1) Link prediction
2) Presentation

6.2 Feng Zeng

1) Node classification

6.3 Yuan Li

1) Report

References

[1] https://www.analyticsvidhya.com/blog/2019/11/graph-feature-extraction-deepwalk/

[2] https://zhuanlan.zhihu.com/p/27234078

[3] Distributed representations of words and hrases and their compositionality[J]. advances in neural information processing systems, 2013, 26:3111-3119.

[4] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '14). Association for Computing Machinery, New York, NY, USA, 701–710. DOI:https://doi.org/10.1145/2623330.2623732