Project 2 Report: Social Network Mining

Group 22

CAI, Shizhan CHEN, Zixin LIANG, Haoran ZHANG, Weiwen

Abstract

Abiding by the pipeline, we utilized Deep Walk and Node2Vector methods, together with hyper parameter searching to handle and analyze our data. After comparing different parameter training results, we finalize the best model based on the AUC score statistic. The results show that the Deep Walk / Node to Vector model with parameters setting: p=2.00, q=16.00, node_dim=12, num_walks=28, walk_length =12 has the best AUC score (0.9391). Above AUC-ROC results can be seen in Best_Model.ipynb.

1. Introduction

Goal of this project is to work on link prediction with the data of social network connections. Our goal is to find the best nodes encoder and its corresponding hyper-parameters so that the similarity in the embedding space approximates similarity in the original network best. And the similarity is calculated by a similarity function which specifies how relationships in vector space map to relationships in the original network. Essentially, node embedding methods are used to map each node to a low-dimensional vector. Here, two basic methods are provided: **DeepWalks encoding** and **Node2Vec encoding**. To compare the results of these two models, we did hyper parameter tuning and visualized the training results. Afterwards, we chose our best model and utilized this trained network to do link connectivity prediction. However, converse to the theoretical expectation that Node2Vector can outperform deepwalk, based on the statistics from the validation set, the best AUC -ROC score reaches **0.9391** with **node2vector**.

2. Data Exploration

Generally, Figure 1 visualizes the structure of the whole graph in our project.

More specifically, the dataset consists of three directed graphs. We represent the number of vertices and edges by invoking *edgesload_data(test_file).head()* and *load_data(test_file).head()* as the pair of (V, E). The training network G_{train} 's properties are (8474,119268), the testing network G_{test}'s properties are (8509,40000), and the validation network G_{valid}'s properties are (5440 ,19268). The clustering coefficients for these three graphs are: 0.1833 for training network, 0.0063 for testing network and 0.0296 for validation network. The transitivity index for these networks is:0.12657 for train, 0.017035 for test and 0.02526 for validation. (Figure 2)

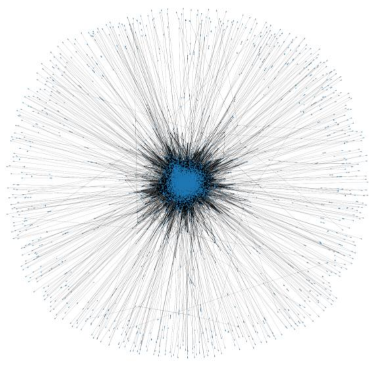


Figure 1: Graph Visualization

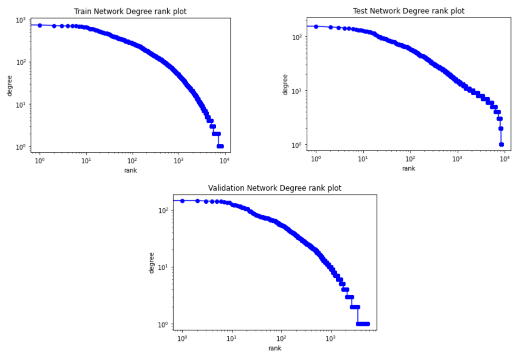


Figure 2: Degree Rank Plot

We can see that the network properties of the three networks are not quite alike: the training network has a higher clustering coefficient than other two networks and has a higher transitivity. By checking the degree distribution, we see that the graphs are similar to the Erdos-Renyi graphs.

3. Our Approach

We have tried many hyper parameter combinations on both models to satisfy this link prediction task. According to the theoretical analysis, the **DeepWalks** and **Node2Vec** respectively utilize *First-order Random Walk* and *Second-order Random Walk* to generate random paths. The random paths in both models are in analogy with 'sentences' in **Word2Vec** model. Essentially, we fit the random walks to some **Word2Vec** model and map similar nodes into similar vectors. Compared with *First-order Random Walk*, a random walker in the *Second-order Random Walk* moves to the next node based on the last two nodes. Not only the local microscopic view, but also the global macroscopic view of the graph can be explored. Theoretically, we expected after tuning the parameter *p* and *q*, **Word2Vec** may outperform **DeepWalks**.

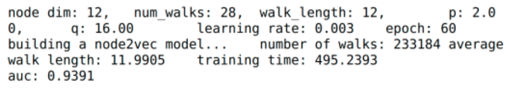
3.1. Parameter Tuning

We first used several for-loops on **DeepWalks** to tune the parameters 'node_dim', 'num_walks' and 'walk_length'. The tuning results show that when they are set to 12, 28 and 12 respectively, the model performs best with AUC score 0.9318. Since we expected **Node2Vec** can fit the training data better with the help of *p* and *q*, we fixed three parameters above and used more for-loops to try different combinations of *p* and *q*. However, no matter how we changed the p and q value, **Node2Vec** always performed worse than **DeepWalks**.

3.2 Further Exploration

After further analysis, we suspected that the learning rates and epochs in **Word2Vec** model might not be ideal for this data set. The original setting for learning rate is the default value 0.01 and 10 for epoch. We suspected this learning rate for a more precise **Node2Vec** model might be too big and the epoch might be too small for us to train the best model. Thus, we modified two functions, "*build_node2vec*" and "*build_deepwal"* and tried several times to find a better learning rate as well as epochs. The AUC scores proved that with a learning rate 0.005 and 50 epochs, the performance of both models improved much. (0.9378 for ***DeepWalk*** and 0.950 for ***Node2Vec***)

Since the AUC for **Node2Vec** still lower than **DeepWalk**, we started to tune *p* and *q* to search for a better result. Based on the comments of the original **Node2Vec paper**, p and q values are recommended to be power of 2 (from -2 to 2). [1] We basically follow their instructions and Figure 3 shows the best performance appeared at 'p = 2, q = 16' with an AUC score **0.9391**.



```
node dim: 12,   num_walks: 28,  walk_length: 12,        p: 2.0
0,      q: 16.00        learning rate: 0.003    epoch: 60
building a node2vec model...    number of walks: 233184 average
walk length: 11.9905    training time: 495.2393
auc: 0.9391
```

Figure 3: Result of Best Node2Vec Model

4. Result

When we initially tuning the parameter, we can reach the strong baseline by using **DeepWalks** model. But **Node2Vec** didn't perform well. And the results are not strong enough to perform the task very well. Results are visualized through heatmaps that are attached in the appendix, figure 4, 5, 6.

After our tuning, we beat the strong baseline by reaching 0.9391 on AUC-ROC scores. Our strategy is setting p=2.00, q=16.00, node_dim=12, num_walks=28, walk_length =12.

Note that all the visualizations with larger resolution are attached in the appendix.

References

[1] A. Grover and J. Leskovec, "node2vec: Scalable Feature Learning for Networks," *arXiv:1607.00653 [cs, stat]*, Jul. 2016, Accessed: May 02, 2021. [Online]. Available: https://arxiv.org/abs/1607.00653.

Appendix

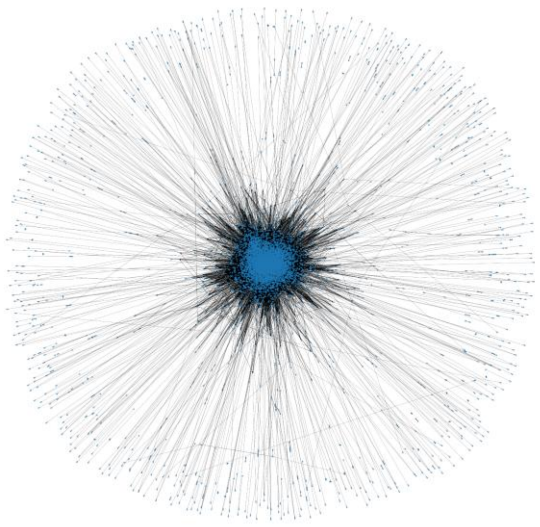


Figure 1

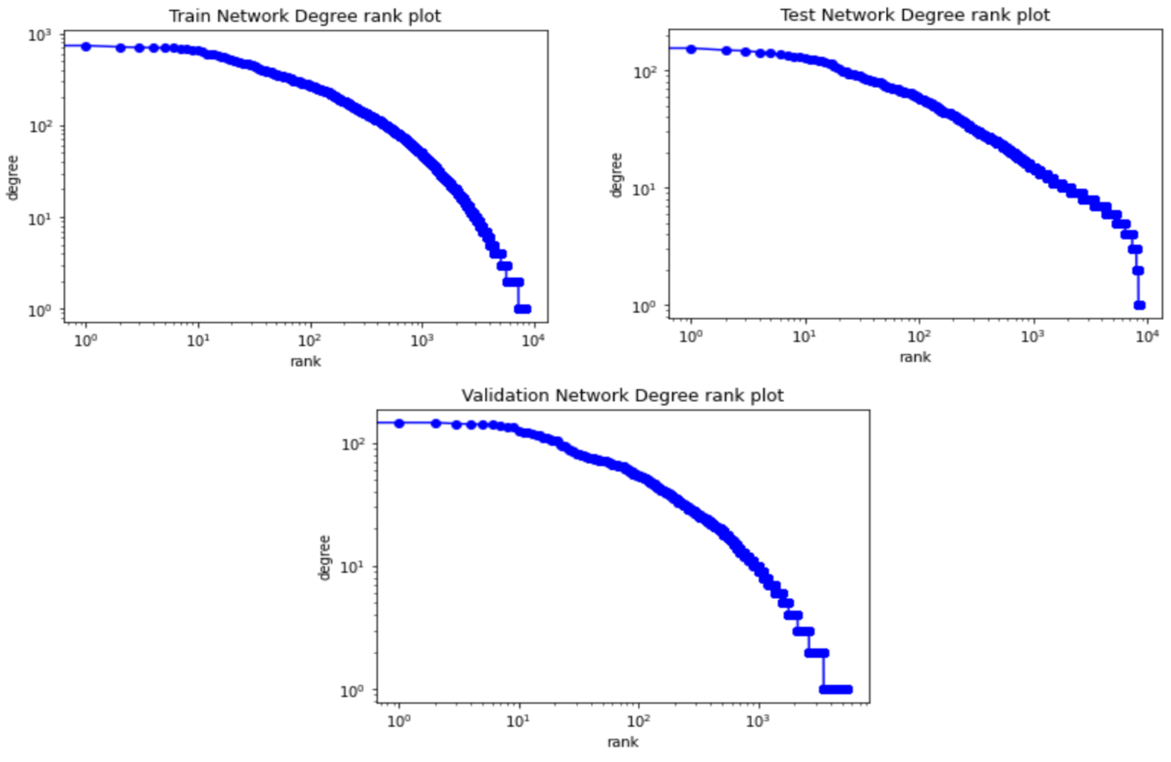


Figure 2

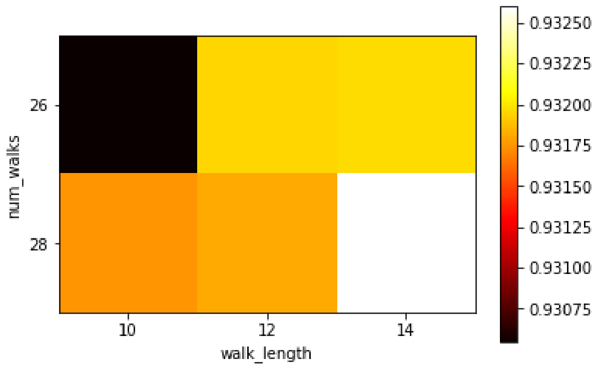


Figure 4: Heatmap for Deepwalk

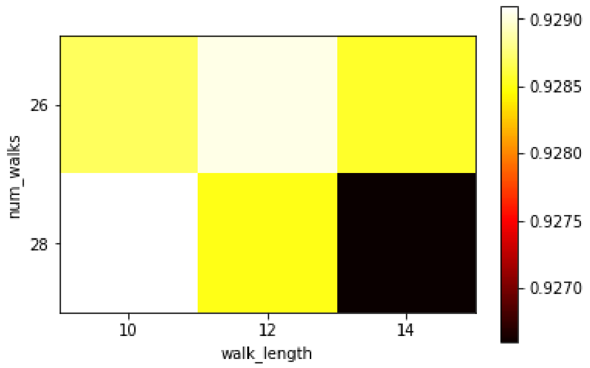


Figure 5: Heatmap for Node2vector

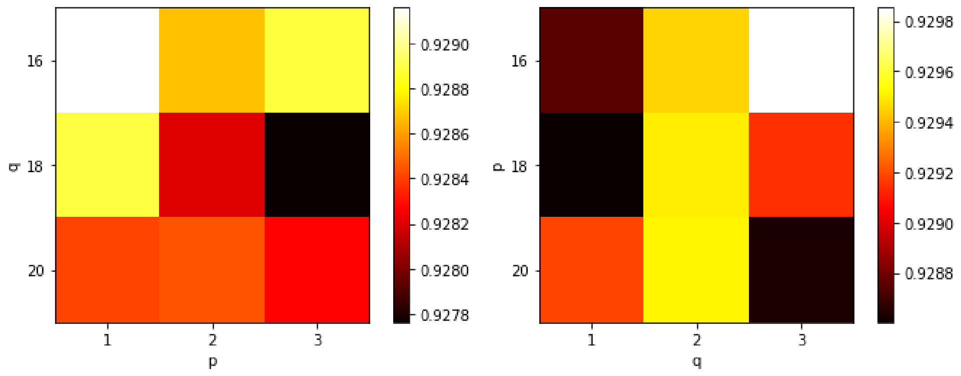


Figure 6: Heatmap for Deepwalk p&q