***Report (Project Two)***

Student Name: Yupeng Wu

Student ID: 45960600

Student Email: yupeng.wu@uqconnect.edu.au

## ***Task***

**Katz method:**

This method ranks the prediction according to the Katz score. The math formulation is as followed:

First we generate a graph by the training data. Then we can get the Katz score with a given source and target in the validation set. Here I got accuracy with in the cosine similarity. Finally, compute the Katz score with the test data and output the top 100 pairs as a prediction.

**Deep walk method:**

In this method I tried to generate three different random walks: random walk, biased random walk and BFS&DFS mixed walk.

Generate different walks by different random walk methods in order to train a skip-gram model. We can use the word2vec function in the model and get a vector which represents a node in the network.

As soon as we succefully get a vector for a node, it is possible to compute the proximity score between two different nodes. The top 100 pairs from the test set with a higher proximity score will be regard as a prediction.

**Random walk (RW):**

This is a very basic method that only randomly choose a neighbor of the source as next node. With a given length, we can build a walk with a loop.

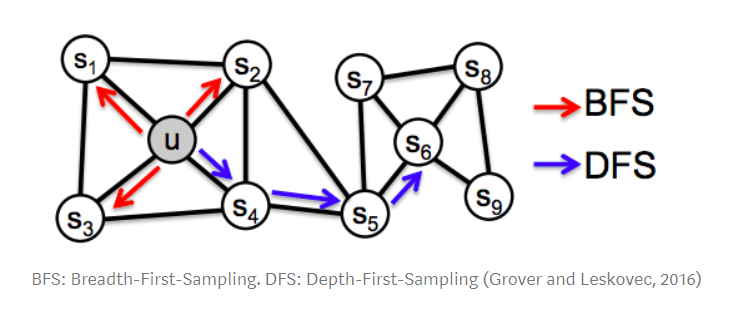
**Biased random walk (biased RW):**

Unlike randomly choose a neighbor of a node, the probabilities of the potential new nodes are unequal. Here I took the betweenness centrality of node as the probability of jumping to node . In other words it will be more likely to jump to a neighbor with higher betweenness centrality, which is defined as below:

Based on the above equation, the recurrence time to a node in the biased walk is given by:

**BFS&DFS mixed walk:**

Instead of walking randomly, here we use Breadth-Fast-Sampling (BFS) to reach immediate neighbors while Depth-First-Sampling (DFS) prefers node away from the source.



## ***Summary***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Method | Katz | RW | Biased RW | BFS&DFS |
| Acc |  |  |  |  |

For the four methods I’ve introduced above, the Katz method has the best performance on the validation data, which have 85% of accuracy. This is a very different finding because I thought deep walk will be a better method since they are more complicated. This reminds us that the advanced method is not always a good method to a project and menmorized method can also do a good job on small size data.

For the biased random walk method, I choose the ‘network.betweenness\_centrality’ function to compute every nodes’ centrality in the graph, which causes the code to be tremendous slow. The outout of this method is also not satisfied so this might be a very bad method.

Finally it is a pity to me that I failed in using GCN to do the link prediction. I challenged myself and tried my best to build a GCN for training and testing by ‘stellargraph’. But I keep getting an accuracy of zero as a result. This will be my goal to complete this task by GCN. Currently my GCN always give me a meaningless output and I believe I can finish it in the future.

## ***Reference***

1. Random Walk in Node Embeddings (DeepWalk, node2vec, LINE, and GraphSAGE),

Edward Ma,

https://medium.com/towards-artificial-intelligence/random-walk-in-node-embeddings-deepwalk-node2vec-line-and-graphsage-ca23df60e493

1. https://github.com/pranavkulkarni/Link\_prediction\_social\_network/blob/a1928c162450c93ada0d2dec80aa4b5bfb341e2e/link\_prediction.py#L64
2. https://stellargraph.readthedocs.io/en/stable/demos/link-prediction/index.html#find-algorithms-and-demos-for-a-graph
3. Tutorials in INFS7450.