COMP90051 Statistical Machine Learning

Project 1: Network Link Prediction

Yazad Jamshed Davur
[redacted]

Introduction

On a broad level, the main aim of this project is to accurately perform Network Link Prediction to predict if a given link/edge is actually a real link in the network or a fake one. A node in the network represents an author and a link between two nodes *A* and *B* indicates that authors *A* and *B* have published a paper together as co-authors.

The training data used is a subgraph of the academic co-authorship and is in a tab delimited edge list format where each row represents a node and its neighbors. The test data consists of 2000 instances where each row has two authors (node pair). The machine learning model needs to output a float value in the range [0,1], which corresponds to the confidence that a link between the two authors is a real one. These values will further be used to calculate the AUC (Area Under the Curve) performance of the model.

All the coding involving the preprocessing of data along with the training of models was performed on the Google Colab platform in Python 3 with dependencies on the *numpy, sklearn, pandas, networkx* and *node2vec* open source libraries.

Methods & Analysis

Initially, the training data given as an adjacency list is read into a graph network with the help of the *Networkx* library in Python. On careful observation, it was noticed that a few nodes in the test data were present that did not have any edges with other nodes in the training data, hence those nodes were manually added to the graph created. There are two main approaches followed for the task in hand.

Initial Approach : Logistic Regression

The training data directly consists of positive samples (real edges), but we require negative samples as well to train a supervised learning model. Hence, negative edges are obtained by creating an adjacency matrix from the training data graph and extracting all those pairs of nodes that do not have a link in between them. This gives us around 8 million negative samples. These negative samples are randomly shuffled and around more than 20 times the number of positive samples are selected as the negative samples. The imbalance between the positive and negative samples is because the negative samples need to cover almost all nodes present in the graph network . Also, positive samples are more robust than the indirectly derived negative samples as suggested in related literature (Tang, Chang, Aggarwal, & Liu, 2015).

Labels are manually created as 1's and 0's corresponding to the combined positive and negative data, 1 indicating a positive sample and 0 indicating a negative sample. The data and labels are fit into a Logistic Regression (LR) model with the help of the *sklearn* library. This model is used as it learns to maximize the likelihood function whereas more complex models might overfit the data at some point. Also, in order to prevent overfitting, the model uses L2 regularization (Julian, Kyle, Lu, 2015) which is simply a penalty applied to the weights by making them small. Using this model, the predicted probability of the 1's class is extracted for each edge in the test data. The AUC value was quite average (**51%**) as no features apart from the source and target node IDs were used to train the model.

In other related work, a group of researchers extracted a set of topological features that can estimate the chance that a given edge indeed exists in the graph in his experiment to identify missing links. These topological features include the node as a feature, node degree as a feature and common neighbors between two nodes as a feature (Fire et al., 2011). On adding just the degree of each node for a node pair as a feature to the model, the AUC increased to about **63.93%**. The reason for the increase can be

attributed to the fact that an author who has co-authored many papers is more active and popular and hence is more likely to form a link with other authors in the future.

The number of common neighbors feature is adopted by many popular online social networks such as Facebook for the friend recommendation task and is based on the common sense that two nodes are more likely to form a link, if they have many common neighbors (Symeonidis & Mantas, 2013). Using this as motivation, the number of common neighbors between the two authors was also added to the feature set. Furthermore, from 8 million negative samples, only those were chosen that had **zero common neighbors** as it is quite unlikely that they will form an edge even in the future. Feeding these features to the logistic regression model paid off with an increase in the AUC to **89.36%** on the test data. This model was used as a new base to perform further experiments on.

Using the given additional feature file, the vectors of specific *keywords* and specific publication *venues* were extracted. Experiments with other given features did not make much of a difference to the predictions. For each node pair in the positive, negative and test samples, the number of common keywords and venues were counted separately and chosen as features and added to the above feature set. This gave a marginal rise in the AUC to **90.3%**. The reason for the expected rise can be attributed to the fact that authors who have more common keywords, perform research in similar domains. Also, authors who have more common publication venues are more likely to meet and work together in the future.

Final Approach: Node2Vec model

An alternative approach to the task in hand involves using an algorithmic framework for representational learning on graphs known as *node2vec.* It is a supervised learning algorithm that maximizes the likelihood of preserving network neighborhoods of nodes (Grover & Leskovec, 2016). The learning is achieved by developing a family of biased random walks, which efficiently explores diverse neighborhoods for a given node (Grover & Leskovec, 2016). Since random walks are based on the connectivity structure between nodes, the model learns different edge features automatically and can be used as a powerful model for link prediction.

Using this as motivation, the node2vec model is trained from the graph network that was initially extracted from the training data, with the help of the *node2vec* library which gives a vector of features for each node as the output. For each node pair in the test data, a prediction is generated by using the cosine similarity measure between the two vectorized nodes as generated by the model. This is done to compute how likely they are to form an edge. The higher the similarity value, the more the confidence that the two nodes will form a real link. Using this model led to a rise in the AUC score to **93.34%** as compared to the previous approach.

One of the main reasons as to why this model is better is because the algorithm automatically learns different structural features of the graph for every node. Also, the biased random walk allows for efficient exploration of the diverse neighborhood around the node in a Breadth First Search as well as a Depth First Search manner rather than just either one of them (Grover & Leskovec, 2016). The flexibility in exploring different neighborhoods is a major advantage to learning richer representations of the network as compared to restricting learning to a small set of extracted features as in the previous approach. Hence this is used as the final approach.

Conclusion

While there might be even more powerful models to perform Network Link Predictions, on experimenting with the given data and features, the node2vec model proved to be the best considering its unique algorithmic framework for reasons as stated in the previous section.

Model (Features)	AUC Score
LR (Only IDs used as feature)	51%
LR (ID, Node Degree as feature)	63.93%
LR (ID, Degree, Common Neighbors)	89.36%
LR (ID, Degree, Common Neighbors, Keyword & Venues)	90.3%
Node2Vec Model	93.34%

Table 1: Performance Comparison between Models

References

Fire, M., Tenenboim, L., Lesser, O., Puzis, R., Rokach, L., & Elovici, Y. (2011). Link prediction in social networks using computationally efficient topological features. *Proceedings - 2011 IEEE International Conference on Privacy, Security, Risk and Trust and IEEE International Conference on Social Computing, PASSAT/SocialCom 2011*, 73–80. https://doi.org/10.1109/PASSAT/SocialCom.2011.20

Grover, A., & Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-Augu*, 855–864. https://doi.org/10.1145/2939672.2939754

Julian, Kyle; Lu, W. (2015). *Application of Machine Learning to Link Prediction*. 3–7. Retrieved from http://cs229.stanford.edu/proj2016/report/JulianLu-Application-of-Machine-Learning-to-Link-Prediction-report.pdf

Symeonidis, P., & Mantas, N. (2013). Spectral clustering for link prediction in social networks with positive and negative links. *Social Network Analysis and Mining*, *3*(4), 1433–1447. https://doi.org/10.1007/s13278-013-0128-6

Tang, J., Chang, S., Aggarwal, C., & Liu, H. (2015). Negative link prediction in social media. *WSDM 2015 - Proceedings of the 8th ACM International Conference on Web Search and Data Mining*, 87–96. https://doi.org/10.1145/2684822.2685295