REPORT

Name: Vighnesh Chenthil Kumar
Student ID: 1103842

Network Link Prediction

Introduction

This project deals with the problem of network link prediction in an academic setting. A network is usually a dynamic structure, meaning that over the course of time, new links are formed, while old ones are severed. The problem that we are trying to solve here is whether we can predict the possibility of two nodes having a link between them. In our case, can we find out whether two authors, who have never had any mutual interaction in the past, will write an academic paper together in the future?

Method

As we can infer from above, all we have is a training network from a particular point of time, consisting of nodes (authors) and the corresponding links (papers) between them. Neither the past nor the future information about these links are given to us. So, the first hurdle is to convert this to a supervised machine learning (ML) problem.

(i) Sampling: As a first step, we need to generate positive and negative samples, i.e., instances with a label of 1 and 0 respectively. After a preliminary analysis, we find that the given training network has 4016 nodes and 26937 edges (links). Since we know for a fact that these edges exist, all 26937 links form our positive samples. Next, using an adjacency matrix, we try to find all edges that don't currently exist in the training network and randomly select 404,055 such "non-edges". This 1:15 ratio between positive and negative samples is aimed at simulating a real-world scenario where the number of "non-edges" vastly outnumber the number of actual edges. (Note: We tested our models on a 1:10 as well as 1:20 ratio of positive-negative samples, but the results were not satisfactory.)

(ii) Feature selection: This is perhaps the most important step in the Network Link prediction problem since a collection of good features can increase the performance of even an average model, while poor features can potentially bring down the scores of a very good model.
For each of the node-pairs (any 2 authors) in the training and test instances, we calculate the following features: Common Neighbors (CN), Jaccard Coefficient (JC), Adamic-Adar index (AA), Resource Allocation (RA), Preferential Attachment (PA), Neighborhood Distance (ND), Total Neighbors (TN), Source Node Degree (UD), and Target Node Degree (VD). (Note: $|\Gamma(u)|$ represents the number of neighbours of node u.) Selection of these features was inspired by the work done by Esders (2015) and Liben-Nowell and Kleinberg (2007), especially since these nine features capture the structural information of the node-pairs to a good extent.

$$CN(u,v) = |\ \Gamma(u) \cap \Gamma(v)\ |$$
$$JC(u,v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|}$$
$$AA(u,v) = \sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{log(|\Gamma(w)|)}$$
$$RA(u,v) = \sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{|\Gamma(w)|}$$
$$PA(u,v) = |\ \Gamma(u)\ | \times |\ \Gamma(v)\ |$$
$$ND(u,v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{\sqrt{|\Gamma(u)| \times |\Gamma(v)|}}$$
$$TN(u,v) = |\Gamma(u) \cup \Gamma(v)|$$
$$UD = |\ \Gamma(u)\ |$$
$$VD = |\ \Gamma(v)\ |$$

While the model trained on these nine features formed the baseline, four other features were extracted from the JSON file, namely the number of papers (n1 and n2) published by each of the two authors in consideration, and the number of common keywords (k) and venues (v) between both authors. The number of papers (n1 and n2) published by both authors is a similar feature as UD and VD, since it captures how probable a given author is of writing a paper. The common keywords and venues between two authors is a measure of how similar their past works have been, thus greatly increasing the probability of them working together in the future.

Furthermore, all these features had a non-zero positive correlation with the corresponding class labels, thus implying that they could all be used together without hampering the performance of the model.

(iii) Models: We observe that the Perceptron (PER) model converges in less than 50 epochs on our dataset, thus implying linear separability of the dataset (Elizondo, 2006). Based on this observation, a variety of ML models (linear/tree-based/ensemble) were selected, including Logistic Regression (LR), AdaBoost (ADA), RandomForest (RF), Gaussian Naive Bayes (GNB), Multinomial Naive Bayes (MNB), and XGBoost (XGB). These models were run on 30% of the test (public) dataset on Kaggle, using the nine baseline features, and their AUC scores were recorded. (Note: The Perceptron model performed the worst of the lot, thus showing that while it may converge in finite time for linearly separable data, it need not give an optimal solution.)

Model	LR	ADA	RF	GNB	MNB	XGB	PER
AUC	0.90455	0.91365	0.91577	0.89909	0.78065	**0.91855**	0.83466

While LR is a good baseline model (for network link prediction tasks) and generalises well (has a good tolerance towards outliers), it is a simple model and can be outperformed by many other complex models. Naive Bayes relies purely on the conditional probability of a particular feature given a class label, and does not "learn" the features in any way, which explains the sub-optimal AUC score it receives. Moreover, while MNB assumes the feature vectors to represent the frequency counts, the GNB model assumes them to belong to a continuous distribution (like in our case), thus explaining the better AUC score of the latter.
RF is a tree-based model which does majority voting on the results returned by a collection of decision trees, to predict the final class label. Ensemble models (like ADA and XGB) use a collection of decision trees too, however, instead of relying on voting, they try to boost the performance by assigning higher weights to misclassified instances by each tree, thus reducing the error rate eventually.
Due to the "boosting" done by these ensemble models, we notice that the XGB model (Chen & Guestrin, 2016) outperforms the rest, including the tree-based models. So we run XGB on possible combinations of the remaining four features, in addition to the nine baseline features.

XGB (9 features)	k	v	n1, n2	k,v	k,n1,n2	v,n1,n2	k,v,n1,n2
AUC	0.91519	0.92589	0.91643	0.92478	0.91992	0.92353	**0.92655**

As we can see from above, the XGBoost model trained on all thirteen features gave the best AUC score by far, primarily because we are now combining author-specific features (like k, v, n1, n2) with the graph-specific features, thus enabling the model to converge at a more optimal solution.

Other Approaches

While our final approach did give us a reasonably good AUC score, we observe a glaring flaw; the feature engineering entirely depends on the author's choice. There could be many other features which capture the structural information better, but end up being missed out.
So, we turn towards a Node2Vec approach (Grover & Leskovec, 2016), wherein we perform multiple random walks (Perozzi et. al, 2014) across the training network to generate node sequences. These sequences provide structural context to each node (degree, nearest neighbours etc.), and can be run through a Node2Vec model to obtain vector representations for each node in a d-dimensional vector space, which is equivalent to the feature engineering that we had done earlier.
We then compute the cosine similarity (a measure of proximity or "closeness") between the vector embeddings of two nodes, since the closer two nodes are in the vector space, the more probable they are of forming a link.
This approach gave us an AUC score of **0.92394**. While this score may be comparable to the better scores received by the XGB model above, it still is not high enough as it does not incorporate author-specific features.

Conclusion

We observe that ensemble models, trained on a mix of graph-specific and author-specific features, work best for the task of network link prediction. We also saw a Node2Vec approach which, although primitive, altogether eliminates the need for complex hard-coded feature engineering. The future work in this field can be to explore how well we can incorporate Node2Vec embeddings with ensemble models, to combine the best of both the approaches we have seen so far.

References

Adamic, L. A., & Adar, E. (2003). Friends and neighbors on the web. *Social networks*, *25*(3), 211-230.

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794).

Elizondo, D. (2006). The linear separability problem: Some testing methods. *IEEE Transactions on neural networks*, *17*(2), 330-344.

Esders, K. (2015). *Link Prediction in Large-scale Complex Networks*. Bachelor's Thesis at the Karlsruhe Institute of Technology.

Grover, A., & Leskovec, J. (2016, August). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 855-864).

Liben-Nowell, D., & Kleinberg, J. (2007). The link-prediction problem for social networks. *Journal of the American society for information science and technology*, *58*(7), 1019-1031.

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014, August). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 701-710).

Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.