COMP90051 Project1 Report

Erya Wen (940717) Haobo Wang(1073871) Zixun Wu(1126832)

1. Introduction

Link prediction is a new challenge in machine learning. The purpose of this project is to predict the possibility of two authors to collaborate on a paper by learning an existing co-authorship network. The rest of the paper is organized as follows. In section 2, we describe the notation and dataset. Section 3 introduces our final implementation in detail. Different strategies are compared and analysed in section 4. At last, we make a summary and analyse the limitation and future improvement.

2. Notation and Dataset

The original training set represents academic authors and their co-authorships containing 9310 rows. Each row is corresponding to the authors list of a paper. It can be naturally described as an undirected graph that we use node *u* and *v* to represent two authors and each co-authorship is presented as a node pair *(u, v)*. The number of times two same authors collaborate are described as weight.

The test set contains 2000 instances. Each row represents a pair of two authors and the ID of them. Our purpose is to determine whether each pair of authors are truly part of the authorship network or if they are fake, to predict whether these test pairs of authors will coauthor at least one paper together. Except 49 new nodes introduced in the test set, all the other nodes are seen in the training set.

3. Methods

3.1 Data pre-processing

Our final implementation constructs the link prediction problem as a classification problem. Firstly, we construct the training set to be a graph(G): (i) each author as a node, ii) and two authors are connected by an edge if they are co-author for one or more papers, iii) the weight of an edge represents how many times the two authors are co-author. The edges in the training set can be seen as positive instances with label 1, then we need to sample some negative instances. Although there are some real edges not shown in the training set, we consider all the pairs of nodes who are not connected by an edge in the graph(G) as the negative samples, and then sample negative instances from them. We try three sampling strategies: random sampling, sampling each negative instance given to the probability relevant to its degree, and negative sampling technique from word2vec. We also experiment on different ratios of positive instances to negative instances. The difference between different strategies is discussed in Section 4.1. The experiments show random sampling and ratio of 1 performs better in this scenario, so we apply them to our final implementation.

3.2 Feature Engineering

To better capture the structure of the given network, we extracted features based on three categories: edge-based, node-based features and node embedding. Basically, the edge-based features focus on using the shared neighbourhood, paths and clustering information between a node pair *(u, v)* to investigate the similarity of two nodes. The value of similarity directly scores the probability of the existence of a link between nodes. While node-based features rely on functions to capture the independent topological properties of a node. They are useful when predicting the relationship of two nodes when they are from two independent groups which do not share any information.

Furthermore, we use the node2vec model [4] to generate the node embeddings with different in-out parameters and return parameters. Node2vec applies random walk as a flexible neighbourhood sampling strategy to get a sequence of nodes. The biased random walk with two parameters: in-out parameter q controls the probability of walking further or nearby; return parameter p controls the probability of visiting the same nodes again. After getting sequences of nodes, word2vec algorithm is used to learn the graph embedding. And then we concatenate the node embedding of two nodes in an edge horizontally as one instance.

Based on the property of our original graph which is undirected and weighted, we explored 8 pairwise features and 7 node-based features (14 features in total as 2 nodes as involved in each node pair instance) and node embedding through biased random walk in progressively. We tested them under (i) different combinations, (ii) with and without weight information. According to the performance, 22 features are used in the final approach. The detail of feature set is presented in Table-1.

	Features	Feature Description
Pair-wised	Jaccard Coefficient	Normalize the count of common neighbours
	Resource Allocation Index	Compute the sum of weight of common neighbours based on the idea that fewer shared-neighbours node pairs are tighter [1]
	Adamic-Adar Index	Variation of RA: increase weight of high-degree neighbours [1]
	Preferential Attachment	Based on the idea of higher-degree node more likely to connect [1]
	Within Inter Cluster	The ratio of within-cluster and inter-cluster common neighbours
	Common neighbour Soundarajan Hopcroft	Using community information to count the of a node pair
	Resource Index Soundarajan Hopcroft	Using community information to compute the RA of a node pair
	Shortest Path Distance	The minimum number of edges passing from node *u* to node *v*.
Node-based	Degree Centrality	Normalize the size of edges a node is connected
	Eigenvector Centrality	Measure the level of influence of a node within the network [2]
	Load Centrality	A betweenness-like measure that takes into account the fraction of all shortest paths of a node
	Closeness Centrality	Measure the average shortest path distance to node *u* overall reachable nodes
	Clustering	The ratio of triangles through node over the degree of node
	K-Core Number	Compute the largest subgraph contains degree k nodes. And k here we use the largest degrees of each node
	Average Neighbour Degree	Measure the dependencies based on the averaged degrees of neighbour nodes in the network

Table –1. Feature Set in our final approaches across two types: pairwise (8) and node-based (14)

4. Evaluation and Results

4.1 Sampling strategy

We experimented with three sampling methods with different ratios of positive to negative instances. The result of the experiment showed that random sampling and the ratio of 1 perform best. Actually this result surprised us at the beginning because we just random sampling as a baseline and the other two sampling strategies are used widely and have nice effectiveness in other fields. Sampling given the degree and negative sampling of word2vec seems to be more reasonable because they tend to sample among the frequent nodes. In common knowledge, nodes with a higher degree usually provide more information value than nodes with lower degree. In [3], Tomas Mikolov et al. showed that subsampling of frequent words can make the representations significantly more accurate. However, in the link prediction problem, there is some difference from word embedding. Since some real co-author relationships are not in the training set, these missing co-authors may be more likely to happen between the high-degree nodes. If we frequently sample among the nodes with higher degree, many real co-author relationships are considered as negative instances in the training set with high probability. Hence sampling among nodes with higher nodes may put many instances with wrong labels(noise) into the training set, which definitely influence the prediction.

4.2 Feature Selection

The feature selection is generally done as two main parts: exploring the best node embedding and the best manually extracted features combination. For pairwise and node-based features, Support Vector Machine Regression (SVR) is used to fit. While, for node embedding, a neural network model is applied to better learning the high dimension encoding features. To be noted, since the scale of the feature's values are different. To normalize the values, StandardScaler() from sklearn is used. Through our experiments, the best node embedding output is with 64-dim features. We used this for further investigation.

The details of the result are shown in Table-2 and Table-3. The result is actually unexpected to see that our best node embedding gives quite a low score of 75%. One of the possible reasons is that there are many connected components in the graph, and some of them consist of just several nodes. In this situation, the random walk may wander around just a few nodes to generate a path and it's hard to generate some useful features from such paths. Furthermore, the authorship graph is not large enough to guarantee node2vec to have a good performance.

For manually extracted features, it is shown that 8 pairwise only features set gives 83% but still lower than the 22-feature set combine pairwise and node-based. It is expected as pairwise features for negative node pairs usually are all 0 as they do not share similar neighbours, paths and groups. This does not include much diversity in the feature and most importantly, new nodes are contained in the test set and gives all 0 pairwise values. Hence, the model tends to predict all these as negative. While we introduced the independent node-based feature to also learn the node itself. The result shows this is very necessary. But node-based features only give only 63% since it ignores the most edge information in link prediction. Moreover, we found weighted information does not help much in learning and even worse it introduces some noise. Furthermore, we attempted to select K best features with Chi2 score function. The result shows all features bring some useful information to this task.

	22-F&Node embedding	Node Embedding	8-Pairwise without weight	14-Node without weight	22-F with weight	**22-F without weight**
AUC	0.86709	0.75441	0.83392	0.63536	0.88387	**0.89485**

Table-2 The result for different feature sets

	K = 6	K = 10	K=14	K=18
AUC	0.72060	0.81749	0.85525	0.87467

Table-3 The result for Select K best features

4.3 Model Selection

As mentioned in Section 3, we consider the task as a binary classification problem. 4 models are tried including Logistic Regression (LR), Support Vector Regression (SVR), Neural Network (NN) and XGBoost classifier. The test results for each model are shown in Table-4.

	Logistic Regression	SVR	Neural Network	XGBoost
AUC	0.76706	0.89485	0.87775	0.75441

Table-4 The test result for four models

From the table, **LR** gives a quite low result. One of the possible reasons is that it is not effective for non-linear data because LR is a linear model. The **XGBoost** also shows a bad performance, which can be due to that it contains too many parameters which introduce noise. Since our feature selection is based on SVR, the feature set may not be suitable for XGBoost. We establish a **feed-forward neural network(FFNN)**. The network consists of three hidden layers and the SoftMax transformation as output layer. The hidden layers are all fully connected layers, and dropout is applied to each hidden layer to avoid overfitting. The Adam method is used to optimize the binary cross-entropy loss. Although the performance of FFNN is a bit worse than SVR due to the inappropriate parameter and hyperparameter tuning, we believe there is much potential improvement for the neural network.

SVR performs the best among the models we experiment. The main advantage of SVR is that it uses radial basis function (RBF) kernel to transform the non-linear data into higher-dimension space where the data become linear. The parameters for our model are: kernel ='rbf', tol = 0.05, C = 1.0, epsilon = 0.1.

It can be seen that the 'C' penalty coefficient is set to 1.0 because higher 'C' could make the model overfit and then lower the AUC rate. The 'tol' represents tolerance stop criteria, that once the error rate reaches this threshold the model stops training. For this sample, the default value which is 0.001 is too low to get a good AUC, and 0.05 is a suitable value. Therefore the SVR model is the final choice.

5. Discussion and Future Improvements

We try several kinds of features and several kinds of models for this link prediction problem. Our final implementation consists of sampling, extracting features manually and training a support vector machine regression model.

From our perspective, the graph embedding and neural network should have a potential improvement for this problem, although we fail to use them to get a good result. For the future improvement, we will do more research on how to generate appropriate graph embedding and how to take advantage of the graph embeddings or on some end-to-end neural network that can extract features automatically (such as graph convolution network).

6. Reference

[1]. E. C. Mutlu, T. A. Oghaz, A. Rajabi, and I. Garibay, "Review on Learning and Extracting Graph Features for Link Prediction", 2019, p. arXiv:1901.03425. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2019arXiv190103425M

[2]. A. Ghasemian, H. Hosseinmardi, A. Galstyan, E. M. Airoldi, and A. Clauset, "Stacking Models for Nearly Optimal Link Prediction in Complex Networks", 2019, p. arXiv:1909.07578. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2019arXiv190907578G

[3]. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean, " Distributed Representations of Words and Phrases and their Compositionality", 2013, p. arXiv:1310.4546. [Online]. Available: https://arxiv.org/abs/1310.4546

[4]. Aditya Grover, Jure Leskovec, "node2vec: Scalable Feature Learning for Networks", 3/7/2016, p. arXiv: 1607.00653v1.[Online] Available: https://arxiv.org/abs/1607.00653