COMP90051 Link Prediction Project

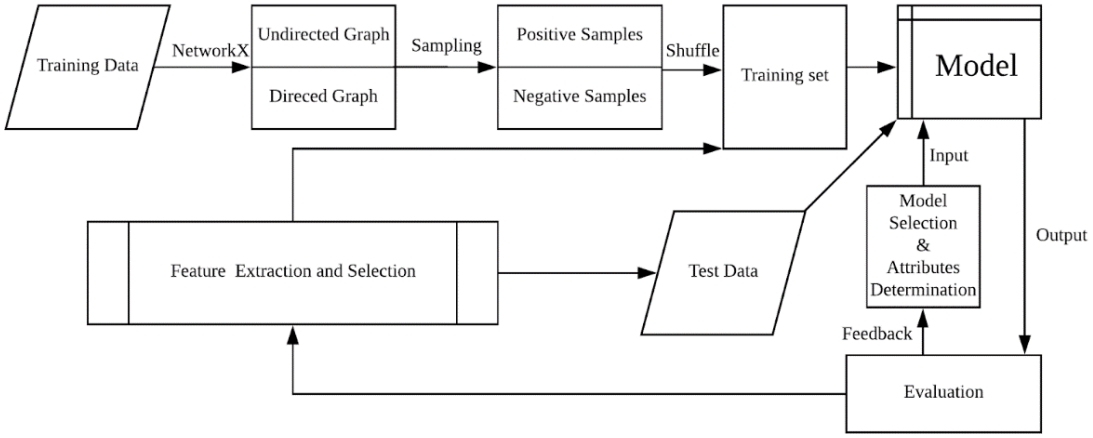
Team Name: W.M.M -- Zhenxiang Wang, Shukai Ma, Jianxing Ma

1. Introduction

Link prediction is to predict the existence of a link between two nodes in a network. Given the ubiquitous existence of networks, it has many applications such as friend recommendation, knowledge graph completion, music recommendation, etc. This report will firstly introduce the experimental pipeline for this link prediction project, then illustrate the feature engineering, sampling approach and models used in this project as well as the rationale behind them, respectively. Finally, it will conclude with how to improve our work in the future.

2. Experimental Pipeline and Data Description

The experimental pipeline of this project is shown in graph 2-1 below. In this project, our task is to distinguish the real edges from the fake ones based on the test data. The training data contains 20,000 records crawled from the Twitter social network, which has 4,867,136 nodes inside. The test data contains 2,000 edges between source nodes (sources) and target nodes (sinks); all the sources were withheld from the training data, while 378 out of 2000 sinks were withheld from the training data. Lastly, according to our validation, all the sources and sinks exist in the whole nodes set, and this fact is fairly essential in feature engineering.



Graph 2-1

3. Feature Engineering

In this project, the characteristics of the given training data derived from the network can easily lead us to associate it with a graph. Fortunately, NetworkX can help us to construct both directed and undirected graph based on the given network. In graph theory, there are various types of algorithms that can measure the different relationships between two nodes, which can help us to extract the latent features under to dataset. In this project, we construct both directed and undirected graph based on the given training data, and then for each nodes pair we separately extract features based on conventional graph algorithms, which are shown in table 3-1. In addition, a brief description for each undirected graph algorithms is shown in table 3-2.

In this experiment, we create a novel heuristic approach to extract features named “User Habits Index” from directed graph. The idea is inspired by the fact in real life: if user X follows A, B and C, while Y is quite similar to A, B and C, then it is possible for X to follow Y as well. In experiment, we create vectors contains the predecessors of A, B, C and Y, then calculate the cosine similarities to measure how similar between A, B, C and Y.

Features in Undirected Graph	Resource Allocation Index (RA), Jaccard Coefficient (JC), Adamic-Adar Index (AA), Preferential Attachment (PA), Cn_Soundarajan_Hopcroft (CN), Ra_index_Soundarajan_Hopcroft (RH) and Within_Inner_Cluster (WIC).
Features in Directed Graph	Successors size for each node(S1/S2), Predecessors size for each node(P1/P2), Predecessors sizes for source and sink (PP), Successors size for source and predecessor size for sink (SP), Users Habits Index (UHI).

Table 3-1

Graph Algorithm Name	Description
Resource Allocation Index	Calculating the modulus of the intersection of common neighbors set of two nodes multiplied by allocation factor.
Jaccard Coefficient	Defined as the size of the intersection divided by the size of the union of the sample sets.
Adamic-Adar Index	Defined as the sum of the inverse logarithmic degree centrality of the neighbors shared by the two nodes.
Preferential Attachment	Calculating by the multiplication between the modulus common neighbors set of two nodes
Cn Soundarajan Hopcroft	Computes the number of common neighbors and bonus one for each common neighbor belonging to the same community given two nodes.
Ra Index Soundarajan Hopcroft	Computes the resource allocation index considering only common neighbors belonging to the same community given two nodes.
Within Inner Cluster	Computes the distance between two nodes within the clusters.

Table 3-2

3.1 Feature Removal Experiments

To better understand which features were most important to our model's predictions, we conducted a set of experiments in which we removed features and re-trained the model and then re-evaluate the model. Table 3-1-1 below presents a part of our experimental data. After this feature removal experiment, features combination "RA, JC, AA and UHI" were selected as the best feature combination. The rationale behind this selection is that: apart from the best performance of this feature combination, (1) the mechanism of UHI is similar to the collaborative filter in recommendation system, which performs well in this project; (2) adding community features such as CN, RH seems not works in this project for the enormous nodes number but deficient edges number; (3) JC can better capture the similarity between two nodes; (4) RA and AA consider the topological sorting near the target node and (5) SS and SP are oversimplified so that they explained less for the given network.

All features =>AUC=0.76552	{RA, JC, AA, PA, PP, SP} => AUC=0.79425
{RA, JC, AA, UHI} =>0.85454	{RA, JC, AA, UHI, SS, SP} => 0.83671

Table 3-1-1

4. Sampling Approach

Simple Random Sampling is applied to generate respectively 50,000 positive and negative samples, which are fed into training model to extract rules. However, this method will bring in noise information due to incomplete information collection of nodes. Better sampling methods like Importance Sampling can be considered in the future work.

5. Models and Parameters Selection

5.1 Models Selection

LightGBM is applied here to learn a function from the input feature space for link prediction between nodes, which was proposed by Microsoft in 2017 as a new GBDT implementation (Ke et al., 2017). As a kind of boosting methods, LGBM refers to an effective method of generating a very high precise prediction rule by combing rough and moderately inaccurate rules of thumb (Schapire, 2001). For better link prediction, we start with a 'base' algorithm for discovering the rough rules from input feature space, and LGBM calls the 'base' regression learning algorithm repeatedly in a manner and combines these weak rules into a preferable one.

Besides, with over 4 million instances and 4 features for each instance in training set, other boosting methods like GBDT are proportional computational complexity to both the number of features and instances. However, it is proven over multiple public datasets that LGBM accelerates the training process by up to over 20 times while reaching almost the same accuracy (Ke et al., 2017). We also implement Logistic Regression as a training model to extract the rules, while it

generally performs worse than LightGBM with lower AUC and precision, which is consistent with advantages of boosting methods. Finally, we implemented a feedforward neural network with 3 fully connected layers: the first two layers contains 512 neural and chose "Relu" as the activation function, and the third layer has 1 neural using "sigmoid" as the activation function to predict the probability. However, although we adopted "dropout" in case of overfitting, the performance of DNN was not satisfied, and this may due to the fact that the feature we used in DNN were the same feature used in LightGBM and Logistic Regression. It may be more suitable for DNN to automatically extract features given node embedding. Comparison of implementation final results using the three models is shown in the table 5-1-1.

	AUC	Precision
Logistic Regression (Baseline)	0.82308	0.91
LightGBM	0.85454	0.94
DNN	0.81353	0.90

Table 5-1-1

5.2 Parameters Tuning

Parameters tuning is performed based on LightGBM Parameters Tuning Documentation and results of three sets of core parameters are compared in the following with analysis. Bayesian Optimization automatic approach is applied in finding optimal hyper-parameters which plays a crucial role in obtaining a good optimizer (Snoek, Larochelle and Adams, 2012). 'Binary objective' is defined to fit the input feature space with labels in {0, 1}. Leaf-wise tree algorithm in LGBM is designed for faster convergence but may result in overfitting. To avoid overfitting, num_leaves and max_depth as two critical parameters to control model complexity are defined in a relation of 'num_leaves less than 2^(max_depth)', and subsample with 0.8715623 and L1 & L2 regularization. A lower learning rate is adopted for higher precision.

Parameters (default)	AUC	Parameters (optimized)	AUC	Parameters (adopted)	AUC
boosting_type: dart matric: {12 for regression} reg_alpha: 0.0 reg_lamba: 0.0 learning_rate: 0.1 max_depth: -1 num_leaves: 31	0.76552	boosting_type: gbdt matric: auc reg_alpha: 0.041545473 reg_lamba: 0.0735294 learning_rate: 0.05 max_depth: 3 num_leaves: 15	0.82113	boosting_type: gbdt matric: auc reg_alpha: 0.041545473 reg_lamba: 0.0735294 learning_rate: 0.02 max_depth: 4 num_leaves: 15	0.85454

Table 5-2-1

6. Conclusion

In this paper, we propose a link prediction system using ensemble machine learning method of LightGBM. Graph-based features generated by random sampling and fed into training model to extract rules between nodes. In the future, a Network Embedding method node2vec based on DeepWalk will be implemented for learning better scalable features, and a better designed DNN using node embedding to automatically extract features will be implemented.

7. References

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. and Liu, T. (2017). *LightGBM: A Highly Efficient Gradient Boosting Decision Tree.* [ebook] Available at: http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradi [Accessed 6 Sep. 2018].

Schapire, R. (2001). *The Boosting Approach to Machine Learning An Overview*. [online] Citeseerx.ist.psu.edu. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.24.5565&rep=rep1&type=pdf [Accessed 6 Sep. 2018].

Snoek, J., Larochelle, H. and Adams, R. (2012). *Practical Bayesian Optimization of Machine Learning Algorithms.* [online] Papers.nips.cc. Available at: https://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf [Accessed 7 Sep. 2018].