COMP90051 Statistical Machine Learning – 2020 Sem2 Project 1

Yinsong Chen, Lin Fan, Xi Qu

1 Introduction

With the fast development of the Internet and the growth of users on social media, pairwise relationships for people on social media (e.g. Twitter) become a popular topic in both academic and industry fields. Data scientists are exploring approaches to predict the relationships of social media users in an efficient way, link prediction technique has been used frequently to predict the relationship between people and thus build a comprehensive user profile. For example, previous work done by Cherry Ahmed, Abeer ElKorany and Reem Bahgat introduced several score based algorithms on link prediction strategies [1].

In the following report, the main approaches to solve link prediction problem including sampling techniques, feature engineering and selection of suitable algorithms will be introduced Followed by a discussion of alternative methods and a conclusion.

2 Approaches

2.1 Data Description

The train and test data are directed paired nodes. There are 20 thousand sources, 4.8 million sinks and 24 million edges in the train.txt. During the EDA phase, we have noticed and exploited the characteristics of data:

- 2.8 millions of sinks in train.txt only appear in the set once and are not source of any other nodes, those nodes only provide information of how many sink the source is following but don't contribute to graph level features such as jaccard_coefficient.
- All the 2 thousand sources in test-public.txt are included in the sources in train.txt, this indicates the features around the source node itself (e.g. what is the community they belong to) would be helpful.

2.2 Sampling Technique

We have generated 20 thousand samples with the same positive/negative ratio as the test data. Negative samples are randomly generated edges that are not present in either train or test file. The source and sink of positive sample data are randomly selected from edges in train.txt. A graph based on the networkx library is built and sample data are used for feature extraction from the graph, as shown in the Figure 1.

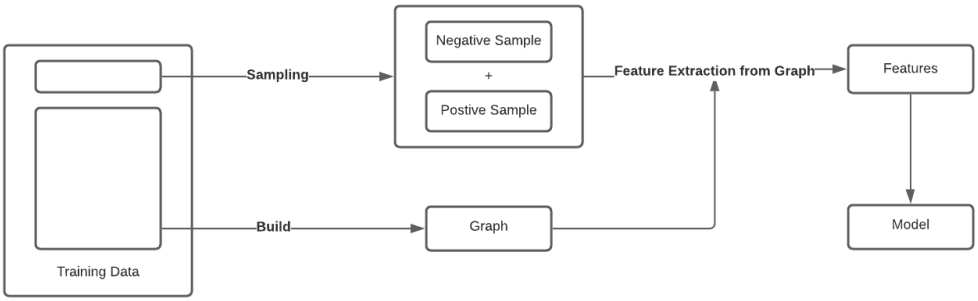

Figure 1: Data preparation pipline

2.3 Feature Engineering

2.3.1 Features based on undirected graph

Networkx library provides a set of link prediction algorithms for undirected graph, we have selected some of them to generate our features: (1) Resource_allocation_index (RAI); (2) Jaccard_coefficient (JC); (3) Adamic_adar_index (AAI); (4) Cn_soundarajan_hopcroft (CSH); (5) Ra_index_soundarajan_hopcroft (RISH); (6) Within_inter_cluster (WIC). These features indicate similarity between two nodes with different metrics.

2.3.2 Features based on directed graph

The twitter data has a directed follower-following relationship, direction is important in this relationship. We have extracted features based on the directed graph: (1) Number of followers of sink node (NFS1); (2) Number of followings of the source node (NFS2); (3) Number of common followings between source and sink node (NCF1); (4) Number of common followers between source and sink node (NCF2); (5) Average number of common followings between the source node and followers of the sink node (ANCF1); (6) Average number of common followers between the sink node and followings of the source node (ANCF2); (7) Cosine similarity [2] between followers of the source node and followers of the sink node (CS1); (8) Cosine similarity between followings of source node and followers of the sink node (CS2); (9) Average cosine similarity between followers of the followings of source node and followers of the sink node (ACS1). Specifically, for each following of the source node, we will calculate the cosine similarity between their other followers and the source node itself and compute an average score.

2.3.3 Information Leak

Originally, we used the entire train.txt data to build our graph; however, we have noticed the performance of testing data is much better compared to Kaggle results. The reason could be the information leak as the graph built for feature extraction already contains edges of positive sample data. Hence, we have revised our logic to remove positive samples from the graph. As shown in Figure 1, the train.txt data has been split into graph data and positive sample data.

2.4 Model Build

At this stage, we have converted the original link prediction problem into a binary classification problem, there are many established algorithms to solve for that. And in our feature engineering step, we have extracted many features which may/may not be useful predicting outcome. So it is important to have a process to select an effective algorithm with right features. Figure 2 highlights the process taken to build our optimal model.

In order to minimize the impact of uneven sampling in model & feature selection process, we have utilized kfold split algorithm to execute each step multiple iterations and taking average scores.

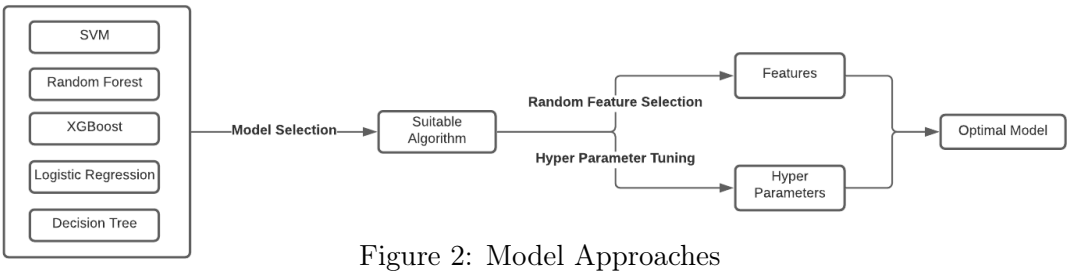

Figure 2: Model Approaches

First, we train data via a list of popular algorithms utilizing all the available normalized features. The objective of this is to pick a few algorithms that perform well for further analysis. We will discuss the algorithm selection in more detail in section 3.2. Next, we perform feature selection using a technique called 'Random Feature Elimination' (RFE), it is essentially generating multiple models each with a different combination of features and comparing model performances. Since different algorithms may respond differently to features, it is important to perform this step for each of the candidate algorithms. The last step is to fine tune hyperparameters given the set of optimal feature inputs. Similar to the feature selection step, Grid search algorithm is used to identify the optimal hyper-parameter settings for the model.

3 Evaluation and Alternative Consideration

3.1 Feature comparison

Feature selection process using RFE during the model build stage has provided some candidate feature sets that yield good performance. We now need to review candidate feature sets based on our understanding.

In reviewing those candidate sets, undirected graph features can be categorised into two types based on their dependency to the community information: features without community (RAI, JC, AAI) and features with the community (CSH, RISH, WIC). Based on similarity and correlation of the features, we can divide the undirected graph features into two subsets: (RAI, WIC) and (JC, AAI, CSH, RISH). Each subset includes features with and without community information. Similarly, directed graph features can

be categorized into three subsets including node features (NSF1, NSF2), edge features (NCF1, NCF2, ANCF1, ANCF2), cosine similarity features (CS1, CS2, ACS1).

It's likely that we might encounter the overfitting problem if too many similar features are included in the model. Hence, we want to explore the impact of reducing correlated features to the model performance. Table 1 shows part of the result. It's evident that our analysis were correct and removing similar & redundant features has improved result.

Table 1: Feature Performance Comparison

Features	Kaggle AUC
(NSF1, NSF2, NCF1,NCF2, ANCF1, ANCF2, CS1, CS2, ACS1, RAI, WIC, JC, AAI, CSH, RISH)	84.923%
(NSF1, NSF2, NCF1,NCF2, ANCF1, ANCF2, CS1, CS2, ACS1, RAI, WIC)	84.750%
(NSF1, NSF2, NCF1, NCF2, ANCF1, ANCF2, JC, AAI, CSH, RISH)	86.143%
(NSF1, NSF2, NCF1, NCF2, ANCF1, ANCF2,RAI, WIC)	87.130%

3.2 Algorithm Comparison

A set of algorithms are tested in search for the optimal algorithm for this problem and the Table 2 shows their average scores.

Table 2: Model Performance Comparison

Model	Testing AUC	Model	Testing AUC
LogisticRegression	90.589%	DecisionTreeClassifier	84.303%
RandomForestClassifier	94.401%	XGBClassifier	95.863%
SVM	90.124%		

- **Logistic Regression** serves as a simple benchmark in determining algorithm suitability.
- **DecisionTree algorithm** has the lowest score as the feature data are all numeric values and sparsely populated. It may be difficult for a single decision tree to find the optimal decision boundary, this also suggests that proper scaling of feature input is an essential step.
- **SVM** generally performs better with numerical features compared to decision tree-based algorithms, however, it takes long time to train and as the result shows it could not compete with ensemble algorithms like RandomForest or XGBoost.
- **Decision tree based ensemble algorithms** have advantages over others. This is consistent with their popularity in recent years: ensemble method could aggregate weak learners into a powerful learner. So RandomForest Classifier and XGBClassifier will be our candidate algorithms for further feature selection & hyperparameter tuning.

4 Conclusion

In this project, we have developed a data processing pipeline with directed and undirected graph features and XGBClassifier to solve link prediction problem. To further improve the performance of the model, future work could focus on reducing information leak in sampling and feature extraction. Also, due to high edge volume, sampling and feature extraction process was time-consuming and often resulted in memory error, utilizing parallel computing in a distributed system could be helpful in solving this problem.

References

[1] C. Ahmed, A. ElKorany, and R. Bahgat1, "A supervised learning approach to link prediction in twitter," *Soc. Netw. Anal. Min*, 2016.

[2] N. Shibata, Y. Kajikawa, and I. Sakata, "Link prediction in citation networks," *Journal of the American society for information science and technology*, vol. 63, no. 1, pp. 78–85, 2012.