COMP90051 Project 1: Twitter social network prediction –Team: knight coders

Introduction
Missing edges prediction is very important, in this project, a graph features based supervised link prediction system for Twitter social network is developed. It is based on the features of directed graph, such as common neighbors of source and sink nodes. NetworkX is used for basic graph features processing as it is well tested and efficient. Scikit-learn is use for evaluation and building machine learning model, for example, K nearest neighbor, Random Forest and Support Vector Machine. Keras is used for multilayer perceptron. A kaggle AUC score 0.92025 is achieved by Random Forest using regression method.

Table 1 Notation

Symbol	Meaning
G	A directed graph
N	A set of nodes in G
(V, E)	Directed edges from source node V to sink node E
$\Gamma(v)$	All neighbors of node v
$\Gamma in(v)$	Inbound set of neighbors
$\Gamma out(v)$	Outbound set of neighbors
$N(v)$	Neighbor set of the node v

Table 1 shows the symbols that will be used in this report and their meaning.

Final Approach Description
To establish a supervised link prediction system, both labeled data and features are required. The system has three modules: Data Processing, Feature Extraction and Classification Algorithms.

1. Data Processing
Firstly, the system will process the given dataset and produce positive (true links) and negative (fake links) samples. In addition, to accelerate feature extraction module, the system will calculate the most frequently used information, such as neighbor list, and store it in a python dictionary.

Positive samples (true links) can be chosen directly from the given dataset, however, negative samples (fake links) should be generated based on known nodes and edges. Since there are 4867136 nodes in the whole graph, only 20000 nodes are likely to be source nodes. To get a meaningful labeled dataset, samples will be generated according to these 20000 candidate source nodes. Positive samples are randomly chosen from the known true edges, and the system will choose only 10 edges for each certain source node. So that there are 200,000 positive samples in total.

On the other hand, it is more challenging to generate good negative samples which can cover different situations. To approach this challenge, the system will generate negative samples based on candidate source nodes: random choosing 20 nodes as candidate sink nodes, checking whether there is the link between a source and sink node pair and random appending 10 pairs who are not in the known edge set as negative samples.

Feature extraction takes a long time, more than 30 hours, as a result, it is essential to calculate frequently used information before extracting features. For each node v in N, 7 features: $\Gamma(v)$, $\Gamma in(v)$, $\Gamma out(v)$, number of $\Gamma(v)$, log of number of $\Gamma(v)$, number of $\Gamma in(v)$ and number of $\Gamma out(v)$, will be calculate and save into a dictionary. The information can be regarded as basic features and they will be used multiple times in the next module for getting higher level features.

2. Feature Extraction
In this module, the system will generate features for supervised learning, and 18 features are applied for training our model. The main idea of our features is to calculate different similarity scores based on common neighbors of each node pair and to have the ability of predicting links of node pairs who has no common neighbors, the system will expand it to 3 aspects.

2.1 Similarity Score Function
(a)Adamic-Adar similarity(AA) - Adamic-Adar is the first feature that we got, which is introduced by Lada Adamic and Eylan Adar in 2003. This method can be regarded as a weighted sum of shared items, and for this twitter network, shared items are the common neighbors. The more neighbors a common neighbor node has, the less weight it will get.

For example, if person A has only 2 followers, B and C, these 2 persons are very similar and there is high probability that B and C have a link.

(b) Jaccard's coefficient(JC) - The Jaccard's coefficient measures the probability that both x and y have a feature f for a randomly selected feature f that either x or y has. The feature is neighbor in this case. Score(x,y) = |Γ(x) ∩ Γ(y)| / |Γ(x) ∪ Γ(y)|. This heuristic gives AUC of **0.826** on test data.

(c) Cosine - Cosine similarity is a common measure for similarity. In our project, the number of common neighbors is regard as inner product and this number will be normalized by the product of the length of Γ(source) and Γ(sink).

(d) Salton Similarity - It is defined as the number of common neighbours between two nodes divided by the square root of the multiplication of the degrees of the nodes. This feature gives us the AUC of **0.84**

The formula for this similarity is : score(x,y) = $|\boldsymbol{\Gamma}(\mathbf{x}) \cap \boldsymbol{\Gamma}(\mathbf{y})| / (\boldsymbol{\Gamma}\boldsymbol{in}(\mathbf{x}) * \boldsymbol{\Gamma}\mathbf{out}(\mathbf{y}))^{1/2}$

(e) Resource Allocation - a resource or a signal that a node could send to another through their common neighbours.

Score(x, y) = $\sum_{u \in N(x) \cap N(y)} 1/|N(u)|$. This feature gives AUC of **0.85**.

(f) Preferential Attachment - We calculate richness of a node by calculating the multiplication between the number of friends or the number of vertices each node has. The formula for this similarity is : score(x,y) = $|\boldsymbol{\Gamma}(\mathbf{x}) \cap \boldsymbol{\Gamma}(\mathbf{y})| / |\boldsymbol{\Gamma}\boldsymbol{in}(\mathbf{x}) * \boldsymbol{\Gamma}\mathbf{out}(\mathbf{y})|$ This features gives the AUC of **0.58**

2.2 Extension

Since an analysis shows that it it possible that 2 nodes have link but no common neighbors, besides thinking how similar is source node $v1$ to sink node $v2$, we need think about how similar are $\Gamma out(v1)$ to $v2$ and how similar are $v1$ and $\Gamma in(v2)$. For example, B is a superstar, and people A is similar to B's followers, then there is a high probability that A follows B. As a result the system will extend the similarity score to these 2 aspects. All the previous 6 features focus on common neighbours between the source node and the sink node.The feature importance analysis of our best random forest predictor shows that these 2 kinds of extended features take a significant amount of feature weight(approximately 16%).

One problem associated with feature extension is computing time. Calculating the similarity scores for inbound and outbound neighbor will takes a very long time, especially for those who need iterate each common neighbors, such as Adamic-Adar similarity. To speed up our system, only cosine similarity and Jaccard's coefficient will be extended, and this saves more than 20 hours on data processing.

3. Classification Algorithms

Random Forest is an ensemble method which can be used for regression and classification. After going through the data we realised that our dataset is very much biased. And random forest is a collection of decision trees. Now if we use many trees together, eventually many of our features have been included. By including all the features, can help us in limit our error due to bias and error due to variance. First we tried to train our model as classification method for which we got the result of 0.81 and then we tried to implement it as regression, for which we got the result **0.92**.

Discussion of Results

1. Data Processing

Using total random negative samples gives us a pretty low score, **0.76**, which is even less than single feature performance. The reason why random generating does not work is that not all nodes in graph can be source nodes, only 20000/4867136 nodes are likely to be a source node. After generating negative samples from these 20000 nodes, our system achieved **0.86** AUC score, which is a great improvement.

2. Feature Extraction

Since single features, such as Resource Allocation, are able to achieve **0.85** AUC score, we want to merge different similarity scores to get a better performance. Rank is more important than absolute score for this task, as a result, each score will be normalised by its rank. However, it is impossible for us to try all kinds of combination method, so we have to go for machine learning or neural network.

Another problem is no-common neighbor node pairs. All our main features are calculated based on common neighbors. If 2 nodes do not have common neighbors, all the higher level features. such as AA, will be **0**. This will lead to a worse performance of our model since an analysis shows that about 20% positive samples share no neighbors. Feature extension is used for this problem. By adding extended features, the AUC score increase from **0.86** to **0.92**.

3. **Classification Algorithms**

Three different classification algorithm(KNN, Random Forest, and Multilayer Perceptron) are used in this project. First of all, the training data is normalised for faster convergence of optimizer like SGD. An accuracy of 0.85 was given by KNN for k value '13'. Compared with KNN, multilayer perceptron with 200 epochs was resulted with a score of 0.89 then we increased the epochs to 1000 which increased our AUC from 0.89 to 0.913. During the process we realised that the computation time for neural network is very long and neural network requires setting more parameters for execution. Where as in Random Forest tried for regression method we got an AUC score of 0.92 which is better than that of Multilayer Perceptron. We choose to go with Random Forest. The main reasons for choosing random forest as our final method are, first one being AUC score at the competition. Secondly the computation time which is less than the that of Neural Network. Third reason is the Random Forest is a more robust model than Neural Network. So finally, we got the conclusion that using Random Forest as a regression method will give better link prediction for this data set based on our features. KNN and perceptron will be talked in the following paragraph.

KNN Classifier:

K- nearest neighbour is the non-parametric method which is used for classification and regression. for determining the value of K depends on the size of the data. As our size of data is very huge, if we choose the larger values of k than it can reduce effect of the noise on the classification. We are considering different features like AA, Jaccard and Salton as weights between the nodes. The advantage of this classifier is firstly, it is easy to implement, secondly in K-NN the only parameter we need to change apart from the features is the value of 'k'. The only disadvantage is that it doesn't learn anything from the training data, simply uses the training data for classifications.

Multilayer perceptron

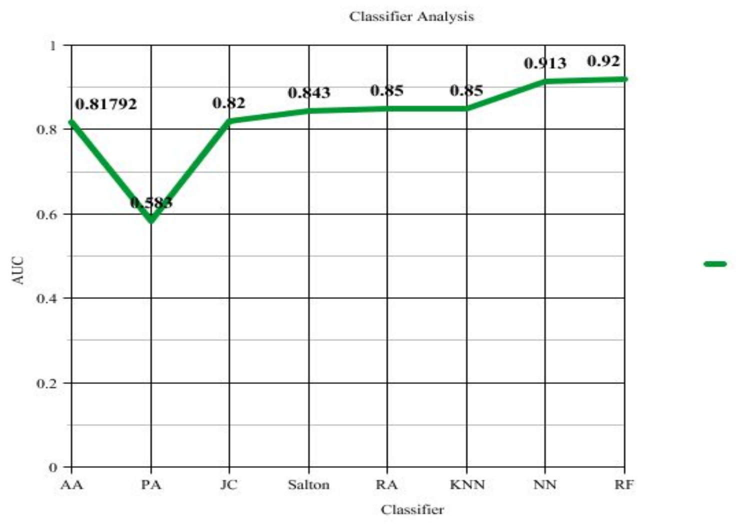
Multilayer perceptron is a class of feedforward artificial NN. The network architecture consists of three fully connected layers. There are 60 units, 30 units and 1 unit of each layer respectively. Sigmod active function is used for the last layer, because the output of system is supposed to be in the range [0, 1].

Classifier	**AUC**	**Hyperparameters**
Multilayer perceptron	0.91	epoch = 1000

Random Forest	0.92	criterion = mse
		n_estimators=100
		max_depth=50
		min_samples_split=50
		min_samples_leaf=50

K-Nearest Neighbour	0.85	k = '7' -------- 0.81
		k = '13' ----------0.85
		k = '31' ----------0.78

Table 1: Classifier Analysis



Conclusion

A graph features based supervised link prediction system for Twitter social network is developed. Candidate source nodes based method for generating negative is used to get a meaningful training dataset. Common neighbor based features and their extension have been used for training Random Forest. A kaggle AUC score **0.92** is achieved by our system.

Future work

Firstly, larger training data could be generated since only 200,000 positive edges has been used by far. Secondly, the feature extension could be applied for other measurements like salton similarity given enough time.

References:

1. Cukierski, W., Hamner, B., & Yang, B. (2011, July). Graph-based features for supervised link prediction. In Neural Networks (IJCNN), The 2011 International Joint Conference on (pp. 1237-1244). IEEE.
2. Link Prediction algorithms, accessed on 26 August 2018, retrieved from - http://be.amazd.com/link-prediction