Link Prediction on Social Network

Rui Hu, Zhangbin Cheng
The University of Melbourne

1. Introduction

The increasing ubiquity and boosting of social networks such as Facebook and Twitter has spurred lots of research in link prediction and recommendation, which aim at predicting unobserved or missing connections based on existing structure in a network. Relationships in a network are represented by a set of nodes and edges, in which nodes are principals and edges are interactions between those principals. In a real world setting, edge information is missing due to reasons such as incomplete data collection process or uncertainty of relationships or resource limitation. Besides, to predict future connections in a dynamic network is also a hot topic. In particular, link prediction can be applied to not only social networking websites, but also a variety of fields would be benefited. Social networking websites would like to customize new friend suggestions for users; intelligence agencies can prevent and predict criminal activities by monitoring potential relationships in malicious networks; financial organizations would like to detect fraudulent activities by inspecting transactional networks. Therefore, establishing a robust machine learning model to effectively predict potential links are worthwhile.

In this report, we reviewed a collection of state-of-the-art approaches in link prediction area (Section 2), inspired by which, we conducted experiments using different methods and build a final model which achieves our best prediction result against the metric abbreviated as AUC (area under the Receiver-Operator Characteristic (ROC) curve). In a nut shell, we have a train data file formatted as adjacent list and a test data formatted as source-destination pairs. For each pair in the test, we need to determine whether this edge is real (missing edge)in the train graph or not. Intuitively, the training network is a sub-graph obtained from the whole Twitter data, and it is noticeable that, unlike friendship on Facebook where edges are understood as undirected, Twitter features for its asymmetric nature of the follow relationship—namely a user can follow another without reciprocation. This factor is taken into consideration throughout the project with inbound edges & outbound edges being distinguished. Basically, we employed both supervised and unsupervised methods to our learning models and surprisingly discovered that the unsupervised method outperformed supervised methods, which is further discussed in following sections (Section 3 & 4).

2. Related Work

2.1 Unsupervised Learning

The link prediction or link recommendation problems are challenging by its sparse nature; that is, nodes have connections to only a very small fraction of all nodes in the network. Research has typically tackled this by using unsupervised approaches, and most of which either generate scores based on node neighbourhoods or path information. Liben-Nowell and Kleinberg [1] thoroughly evaluated a range of unsupervised methods and concluded that the *Adamic-Adar* measure of node similarity performed best. *Adamic-Adar*measure weights the importance of a common neighbour *v* by the rarity of relationships between other nodes and *v*. They also found the baseline common neighbours predictor worked surprisingly well compared with some more complicated predictors such as *Jaccard's coefficient*, *SimRank* [2] and random walk based *hitting time*. In [3], the author utilized a modified random walk approach--Personalized PageRank to calculate rank values for each node. A Personalized PageRank is like standard PageRank [4], except that when randomly teleporting to a new node, the surfer always teleports back to the given source node being personalized (rather than to a node chosen uniformly at random, as in the classic *PageRank* algorithm).Besides, a new unsupervised method *PropFlow* was introduced by [5], which was also a variant of *PageRank* but with more localized measure of propagation and insensitive to topological noise far from the source node. The computation of *PropFlow* does not require walk restarts or convergence but simply employs a modified breadth-first search restricted to a specified height.

2.2 Supervised Learning

While previous studies on link prediction have focused in unsupervised single metrics, [5] believed that supervised framework would considerably outperform them especially in terms of dealing with extreme imbalance in link prediction. The authors in [5] carefully explained some of the properties of imbalance in spare networks with its relationship to graph distance, and how to overcome this by supervised learning. They finally achieved desirable results by extracting such as in-degree, out-degree and some unsupervised measures such as number of *common neighbours*, *Adamic-Adar, shortest path* as features to train some ensemble classifiers like *Bagging* and *Random Forests*. Likewise, similarity scores were extracted as features for supervised classification in the studies of [6], which enriched the feature set in [5] to a more comprehensive one with 39 different features (e.g. *Cosine similarity*, *Bayesian Sets*, *EdgeRank*). In particular, they sampled false edges from the full set of nodes for the training, and used *Random Forests* as classifiers to distinguish between true and false. Besides, [7] came up with a novel algorithm called *Supervised Random Walks* which combined the information from the rich node and edge features and the network structure. On the other hand, our training graph does not tell any node and edges characteristics such as age, gender, salary and edge creation time etc.

Methods mentioned above give us a lot of motivation in both perspectives that how to examine a machine learning problem and how to evaluate our experiment results. Actually, there is still no evidence showing supervised methods will definitely beat supervised ones or vice versa, therefore we would like to conduct a series of experiments with various approaches to obtain a better understanding of the link prediction problems.

3. Methodology

3.1 Idea Overview

In our experiment, the training graph *G* is expressed as an adjacency matrix *A* consisting of 4,867,136 users joined by 20,000,000 edges. To predict whether a specific testing edge is real or fake, we pre-processed the training dataset by uniformly sampling 10,000 positive edges and 10,000 negative pairs based upon matrix A, which was chiefly due to the limitations of memory and processor. Then, each of these training edges would be transformed to a vector of features with a label of *real* or *fake*. Thus, link prediction problem $v \times u \Rightarrow \{0,1\}$ could be solved by employing unsupervised and supervised learning models with effective features.

3.2 Training Data Preprocessing

Followers matrix generating -- The training data given for experiments is a tab-delimited adjacency matrix, where each row represents a user and its outbound neighbours (*followees*). Based upon this dataset, we can efficiently extract followee-related statistics; however, retrieving follower-related information would be prohibitively expensive with a worst case of $O(N^2)$. Hence, we pre-generated a followers matrix *B* from original graph, for the sake of cheaply obtaining followers statistics.

Celebrities data removal -- In our training data, the average number of followees for each user is 93, whereas a few users (probably celebrities) have more than 100,000 followees. We believe that the huge difference between celebrities and normal users data could mislead our learning model. Therefore, these 'outlier' data is removed for the sake of obtaining a more accurate classifier.

Positive edges sampling -- Training 20,000,00 edges at once would be impossible due to the memory limitation. An natural alternative would be to perform training on just part of the data, which can be achieved by uniformly sampling. In our experiment, 10,000 positive edges are sampled as a fraction of the final training data.

Negative edges generating and sampling -- The original training data only provides us with positive edges, which means negative links need to be generated manually. Based upon forementioned matrix *A* and *B*, negative edges are generated in the following way: if user *v* is not included in the followees set of user *u*, it can be said that link $u \not\rightarrow v$ is negative. We randomly picked up user *v* 100,000 times to generate 10,000 negative edges as another fraction of the final training data.

3.2 Feature Set Extraction

Choosing appropriate feature sets is the most crucial part of any classification algorithm. Since feature possibilities are endless, extracting useful features is typically done by trial and error rather than any principled approach [7]. In this report, we would typically explore the use of three feature sets: *proximity*, *topical* and *aggregation* [8].

3.2.2 Proximity Features

Proximity features are characteristics that represent some form of proximity between the pair of nodes [2]. For instance, it is highly possible that a user could follow another with whom they share a mutual friend. Additionally, proximity features are usually cheap to be computed. Denotations used in this section are: $\Gamma_{in}(u)$ denotes the followers of *u*; $\Gamma_{out}(u)$ denotes the followees of *u*.

Common Followers Count -- Measures the overlap of the followers between *u* and *v*.
CFER$(u,v) = |\Gamma_{in}(u) \cap \Gamma_{in}(v)|$

Common Followees Count -- Measures the overlap of the followees between *u* and *v*.
CFEE(u,v) = $|\Gamma_{out}(u) \cap \Gamma_{out}(v)|$

Common Friends Count -- Measures the overlap of the friends between *u* and *v*, where bi-directional edge denotes friendship. It can be formularised as: CF(u,v) = $|\Gamma_{out}(u) \cap \Gamma_{in}(u)| \cap |\Gamma_{out}(v) \cap \Gamma_{in}(v)|$

Common Neighbours Count -- Measures the overlap of local networks of *u* and *v* by ignoring the directions of links. [2] This feature is included to evaluate the impact of direction on link creation. CN$(u,v) = |\Gamma_{out}(u) \cup \Gamma_{in}(u)| \cap |\Gamma_{out}(v) \cup \Gamma_{in}(v)|$

Similarity Measurements -- Measures the statistical similarity of the followees, followers, friends and neighbours between *u* and *v*. In our experiment, four methods are employed, which are *Cosine Similarity*, *Jaccard Coefficient* and *Adamic-Adar similarity* [8], respectively.

3.2.2 Ego-centric Features

Ego-centric features are those features concentrating on the local network of *u* or *v*. An example would be the number of followers of *u*. We believe that the contribution of this sort of features can be very helpful in link predictions. Specifically in our experiment, four ego-centric features are included:

the number of followers of *u* or *v*, the number of followees of *u* or *v*, the log ratio of *u*'s followers and *v*'s followers and the log ratio of *u*'s followees and *v*'s followees.

3.2.3 Aggregation Features

In order to make features become more related to both *u* and *v*, the simplest aggregation function SUM is applied to generate effective features for link prediction, which are *sum of followees*, *sum of followers sum of friends*, *sum of neighbours*, respectively.

3.3 Edges Classification

A vast number of classification algorithms can be chosen for link predictions. In this report, we would experiment unsupervised learning methods by employing forementioned similarity features as well as supervised learning algorithms including KNN, Random Forests, Bagging + Random Forests and SVM. For unsupervised learning methods, JAVA would be used to program the algorithm. For the rest of the algorithms, an external machine learning package called WEKA would be employed.

3.3.1 Unsupervised Learning

We would start classification with unsupervised learning since it is more straightforward than supervised learning. To ensure that unsupervised learning would produce desirable results, we need to apply some features that in a way can reflect the overall statistical pattern of the testing data. From previous feature sets section, it can be seen that similarity features suit our objective best. Specifically, the similarity score (*Cosine*, *Jaccard* and *Adamic-Adar*) of *u* and *v* would be computed and then scaled to [0,1] as the confidence score of whether the link between them is real.

3.3.2 Supervised Learning

As for supervised learning, we would mainly experiment four algorithms, which are KNN, Random Forests, Bagging + Random Forests and non-linear SVM, respectively. KNN was chosen as our classifier candidate is because KNN is a very simple classifier that works well on basic recognition problems. Additionally, KNN is robust to noisy training data and effective when the training data is large. By applying Bagging which is one of the ensemble techniques, we expect that the performance would be significantly better that the base classifier. As for non-linear SVM, even though the training time is relatively long, it is capable of capturing complex relationships between nodes without the need of performing difficult transformation on our own. Thus, we would like to employ SVM method to see if it could produce desirable results in the context of link prediction.

4. Experiment Results and Analysis

In both training and testing dataset, counts of real edges and fake edges were almost the same, which means a baseline classifier would have an accuracy 50% by predicting all testing edges are 1 or 0.

Table 1 depicts the performance results of unsupervised learning on similarity features. It can be seen that all the similarity features we attempted achieved accuracy above 80%, which indicates that these similarity features have a good capability of discriminating real and fake links.

Table 2 shows the performance comparison for different supervised classifiers on training and testing dataset. The performance results in Table 1 were produced by 10-fold cross-validation on training data and Table 2 was constructed based on the AUC scores over testing data. As we can see from these two tables, non-linear SVM performed the best for both datasets with an accuracy of 76.3% and 75.5%, respectively. Moreover, all classifiers achieved a better performance on training data than testing data, which reveals that even though cross-validation is applied to mitigate the risk of overfitting, these classifiers still overfit training data to some extent. More interestingly, it is notable that the accuracy of Bagging plus Random Forests did not outperform the individual Random Forests classifier - 73% to 74%, which implies that the majority of wrong classifications might come from the bias error introduced by non-discriminated feature values.

To compare the results of Table 1 and Table 2, it can be surprisingly found that unsupervised learning methods perform significantly better than supervised learning models. We believe that two aspects contribute to this result, one of which is sampling bias introduced by the edges sampling done in pre-processing phase. The other would be that our selected features are still not discriminative enough to support supervised models to learn effective rules.

To evaluate our features, the distributions of real and fake edges for six important features in training are shown in Figure 1 (please refer to Appendix section to view all features). It can be clearly seen that for feature F3, F6 and F13, the distribution of real and fake classes exhibit significant bigger differences than that for F2, F7 and F11. Hence, it can be said that F3, F6 and F13 are more effective for classification algorithms to learn patterns. Additionally, it can be inferred that wrong classifications are most likely contributed by the fractions sitting under the critical overlap regions for most features [8].

Unsupervised Classifier	Accuracy (on testing)
Cosine Similarity of neighbours	81.55%
Jaccard Similarity of neighbours	81.06%
Cosine Similarity of followers	80.70%
Jaccard Similarity of followers	80.54%
Cosine Similarity of followees	80.30%
Jaccard Similarity of followees	80.09%

Table1: Performance of unsupervised learning on different features

Supervised Classifier	Accuracy (cross-validation on training)	Accuracy (on testing)
Non-linear SVM	76.3%	75.5%
KNN	72.8%	68.3%
Random Forests	74.8%	71.09%
Bagging + Random Forests	73.5%	70.56%

Table 2: Performance of unsupervised and supervised classifiers

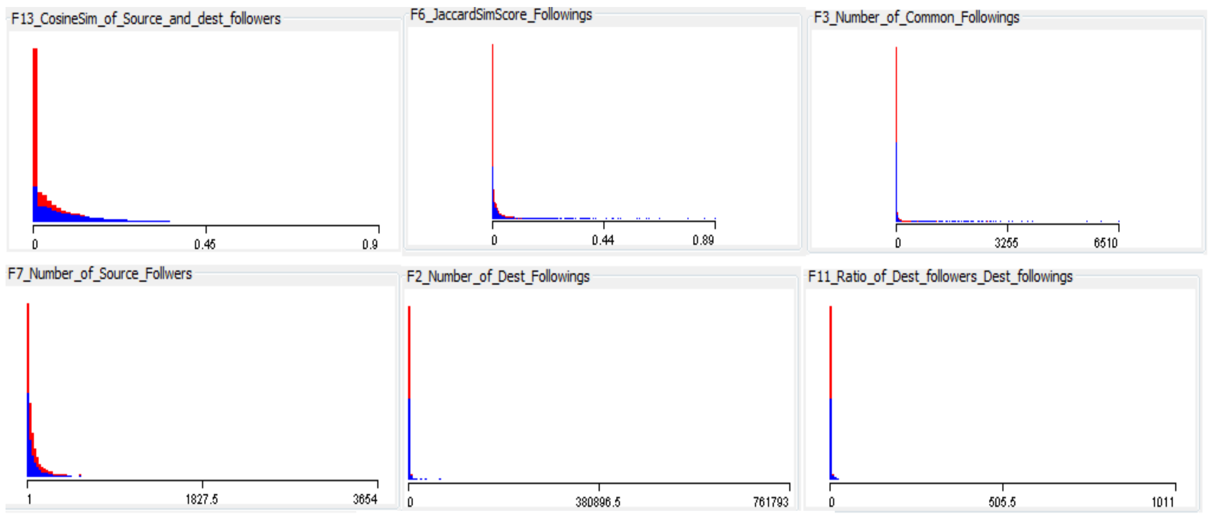

Figure 1: Partial Features Evaluation

5. Conclusion

Link prediction in large graph is still very challenging and attracting a lot of research focus on it. We reviewed available literatures in this field and implemented a number of algorithms and models to explore their predictive capability. We found that in unsupervised domain, *Cosine Similarity* performed best followed by *Jaccard's coefficient* and *Adamic-Adar*. In supervised domain, we extracted 14 features from the network structure and applied classifiers on them. To our surprise, *Bagging with Random Forests* did not achieve the best performance but *None-Linear SVM* outperformed it instead. The reasons were discussed in this report, and we believe our sampling and feature extraction need to be improved for a better result in terms of supervised learning.

There were many promising approaches which were not implemented and tested due to hardware or time constraints; we would like to explore more possibility in link prediction area in the future.

References

[1] David Liben-Nowell and Jon Kleinberg, “The link prediction problem for social networks,” in *Proceedings of the twelfth international conference on Information and knowledge management*,NewYork, NY, USA, 2003, CIKM ’03, pp. 556–559, ACM.

[2] Glen Jeh and Jennifer Widom. SimRank: A measure of structural-context similarity. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and DataMining*, July 2002.

[3]Edwin Chen. (2012).*Edge Prediction in a Social Graph: My Solution to Facebook's User Recommendation Contest on Kaggle.* Retrieved September 15, 2013 from http://blog.echen.me/2012/07/31/edge-prediction-in-a-social-graph-my-solution-to-facebooks-user-recommendation-contest-on-kaggle/

[4] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the Web. Technical report, Stanford University, 1999.

[5]Lichtenwalter, R. N., Lussier, J. T., &Chawla, N. V. (2010, July). New perspectives and methods in link prediction.In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 243-252).ACM.

[6]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.

[7] Backstrom, L., &Leskovec, J. (2011, February). Supervised random walks: predicting and recommending links in social networks. In *Proceedings of the fourth ACM international conference on Web search and data mining* (pp. 635-644).ACM.

[8] Al Hasan, M., Chaoji, V., Salem, S., & Zaki, M. (2006). Link prediction using supervised learning. In *SDM’06: Workshop on Link Analysis, Counter-terrorism and Security*.

[9] Li, P., Liu, H., Yu, J. X., He, J., & Du, X. (2010). Fast Single-Pair SimRank Computation. In *SDM* (pp. 571-582).

[10] Latha, R. H., & SathiyaKumari, K. Survey On Link Prediction In Facebook And Twitter.

[11] Rowe, M., Stankovic, M., & Alani, H. (2012). Who will follow whom? exploiting semantics for link prediction in attention-information networks. In *The Semantic Web–ISWC 2012* (pp. 476-491). Springer Berlin Heidelberg.

[12] Gupta, P., Goel, A., Lin, J., Sharma, A., Wang, D., & Zadeh, R. (2013, May). Wtf: The who to follow service at twitter. In *Proceedings of the 22nd international conference on World Wide Web* (pp. 505-514). International World Wide Web Conferences Steering Committee.

Appendix

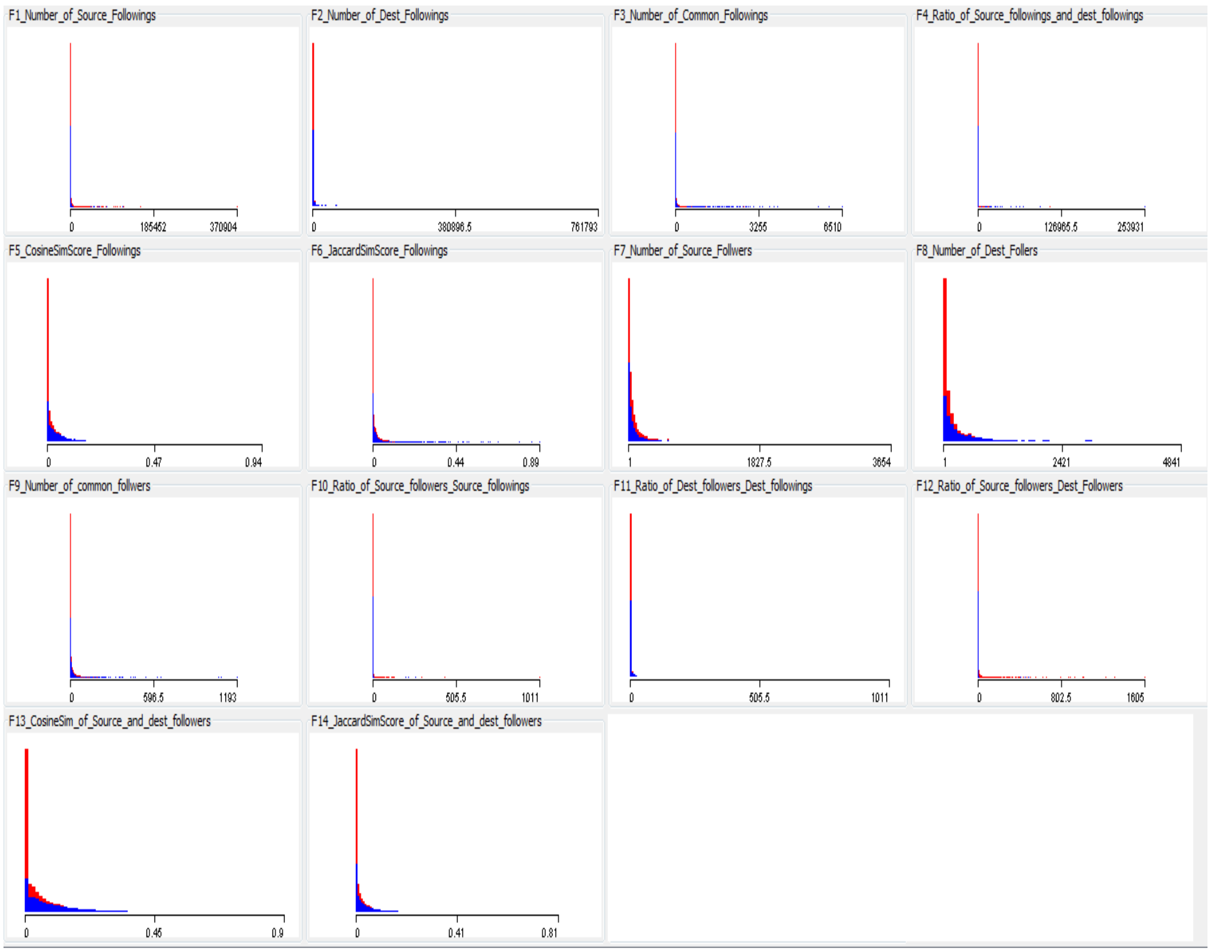

Figure 2: Evaluation of all selected features