Scalable Link Prediction in Online Social Networks

Akshay Bhat, Pracheer Gupta

Cornell University

{aub3, pg298}@cornell.edu,

ABSTRACT

We describe a link prediction method based on a scalable community detection algorithm. It can be used to recommend new links in a real world social network with millions of users. Using a Hadoop cluster, we test our implementation on a Twitter user network containing 40 million users and 1.4 billion connections. We show that communities detected can then be used to recommend new users to follow for existing users. We describe issues in evaluating a link recommendation algorithm. Finally we discuss the case of recommending influential users using a linear support vector machine classifier.

Categories and Subject Descriptors

H.2.8 Database Management: Database applications – Data mining

General Terms

Measurement

Keywords

Link Prediction, Community Structure, Community Detection, Label Propagation

1. INTRODUCTION

Over the last few years, online social networks have gained immense popularity. The ability of being able to recommend new users of interests in such networks is extremely useful. Since the more friends (and in case of twitter, interesting users) a person has (or follows), more likely he is to return to the website. This in turn leads to higher page views and consequently higher revenues.

A naïve approach for predicting friends would be to perform a two level breadth first search starting from the users and to recommend users which are found in the search and are not currently connected. However, since each user on an average is connected to 100 other users (in our dataset), doing a breadth first search for each of the 40 million users is computationally prohibitive. However, if we can find a set of users tightly connected to each other, then we would be able to make predictions for the entire group as a whole rather than for each individual user, making the process computationally better.

The process of detecting clusters in networks is called as Community Detection. A community, also sometimes referred as a cluster or a module, is defined as set of nodes in the networks with more edges between the members of the set as compared to the rest of the network. Numerous algorithms exist for detecting the community structure. Most of these algorithms rely on optimization of modularity, a quantity that measures goodness of community structure. In this work we utilize a different kind of algorithm, called as Label Propagation algorithm, first proposed by Raghavan et. al. in 2007 [6]. Label Propagation algorithm is extremely scalable, has a near linear time performance and does not require any prior information about the community structure [6, 5].

To the best of our knowledge, there hasn't been any work describing Community Detection or link prediction in networks with more than 10 million nodes. The closest work we found focused on determining 'betweenness centrality' for network of 40 million[1] Twitter users, using a 128 processor Cray XMT [2, 3]. We used the same dataset for testing our implementation [3].

In following sections we describe the community detection algorithm followed by link prediction methodology using the detected communities. We then describe the dataset used and results obtained. We also discuss results obtained using classification algorithms for predicting new followers for influential users. Finally we describe a method based on creating community specific singular value decomposition based models for link prediction.

2. Community Detection and Link Prediction

2.1 Label Propagation Algorithm

All nodes are initialized with a unique label and at every iteration each node assumes a label used most frequently in its neighborhood. Over iterations, sets of nodes strongly connected to each other, end up getting the same label. All nodes having the same label are considered part of the same community [6].

More accurately the algorithm is defined as:

1. Assign a unique label to each node in the network. For a given node x, let its label at time 0 is $C_x(0) = x$.
2. Set $t = 1$.
3. For each node x in the network, find the most frequently occurring label among all the nodes with which x is connected to. Ties are broken uniformly and randomly.

   $C_x(t) = f(C_{x1}(t-1), C_{x2}(t-1),..., C_{xk}(t-1))$.

1 The paper [2] inaccurately reports number of users in dataset from [3] as 61 Million, 61 Million is highest index (numeric twitter id), and not number of users.

4. Check for some convergence criterion, if met, stop, else set $t = t + 1$ and go to step 3.

Due to randomly broken ties, the algorithm is non deterministic in nature.

2.2 Community Detection using Map Reduce

For the purpose of Community Detection, a single iteration corresponds to a single map phase. A reduce phase is not required. The network in an adjacency list (of the original data set) serves as the input which is split across multiple machines. In each iteration, the current labels for all nodes are shipped to each machine running map process. Thus step 3 in the algorithm described above can be parallelized. The map process outputs the labels for current iteration.

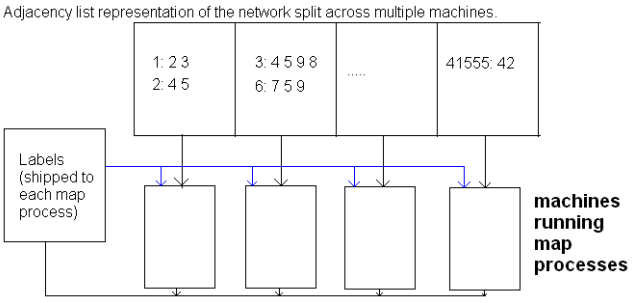

Fig 1. Community Detection using Map Reduce

2.3 Link Recommendation using communities

Once the communities have been detected using the label propagation algorithm, the process of recommending new users is simple. For each community, we enumerate all users followed by the members in the community. This is similar to performing a breadth first search for each user in the community, but with only one degree of separation instead of two. Since all members of a community are tightly connected to each other, there are numerous users which occur multiple times.

This predicted set of users to be followed is then ranked by the frequency of occurrence and top 500 users are selected as a prediction for each community. We use the community discovered after fourth iteration of label propagation algorithm, since the community size is smaller.

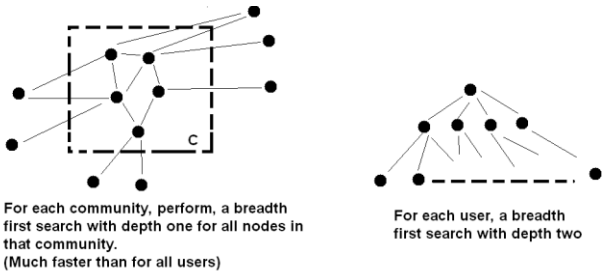

Fig 2. Comparison of new technique for Link Prediction vs Baseline technique

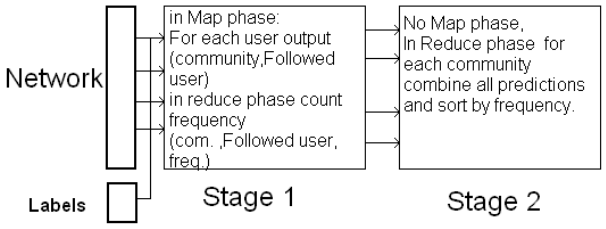

Fig 3. Link Recommendation using Map Reduce

2.4 Baseline - Breadth First Search

The naïve approach to link prediction employed by most of the social networking sites is based on breadth first search with at most two degrees of separation. The number of common friends with a second degree connection can indicate how likely one is to connect to them. We apply this methodology for users in test set to get a baseline for comparison.

2.5 Evaluation methodology and recall as metric for measuring

We use the new set of links formed by the users after a period (as on November 2010) as the ground truth to test against. Consider a user John Doe, who was following 100 users in our network dataset (July 2009), and now is following 20 more users. Our goal is then to find how many of those 20 users can be predicted by our algorithm. Thus we use recall as the metric of choice. For each user in test set, we generate predictions equal to twice the number of new users he connected to after a year. The ratio of correct predictions to number of users which were followed additionally (which is same as recall) is used as a metric.

3. Recommending followers for influential users

Since some of the users, such as celebrities, heads of state etc. have large following, we can predict whether given a user is likely to follow the influential user. This is similar to an unbalanced classification task. Consider the user BarackObama: as of Dec 2010, 6 Million users follow him. Thus we can construct a classification problem, where for each user we predict whether he is likely to follow BarackObama or not. This is feasible since large numbers of users are following a single user and thus this method is useful only for users who have at least hundred thousand followers.

We also account for the difference in class distributions while making the predictions. For this task, the features for each user would be other influential users he is following. We define influential users as ones with more than 1000 followers. This leads to large number of sparse features and consequently, SVM^{perf} [7] classifier is used for classification.

4. S.V.D. based prediction

Rather than simply calculating the frequently followed users by all users in the community, we can create a simple Singular Value Decomposition based model for each community. For a given community we represent the follower information for all users in the community using a sparse matrix (A). Each row in the sparse

matrix corresponds to a user in the community, while each column corresponds to a user being followed by at least one member of the community. When, say, a user1 is following user2 then

$$A[user1, user2] = 1$$

The matrix A is then approximated by truncated S.V.D.

$$A = U \times S \times V$$

Using the principal components we calculate A' an approximation of A.

$$U \times S \times V = A'$$

A' serves as a basis of prediction. If A'[user1, user2] is higher than what one would expect at random, then it means that user1→ user2 (user1 is following user2). We can use this property to predict new users to follow for a given user.

In order to test above hypothesis, while constructing the sparse matrix A, we leave out a small number of edges, i.e., we set

$$A[user1,user2] = 0.$$

These edges then belong to test set; we also create an equal number of random edges. Finally after performing S.V.D. we predict values in A' for all edges in test set, and determine whether the predicted value can discriminate between an original and a randomly created edge.

5. Dataset & Experimental Setting

We use Twitter's network dataset collected by Kwak et. al [3]. The network contains 40 million users, with 1.4 billion directed edges between them. This network was collected in June 2009 and contains nearly all users present at the time [3]. Since twitter contains few users who have large number of followers, we create a second network which contains only users who have less than 1000 followers (*in* degree). We believe this network is better at capturing the underlying social network. Also the edge directionality is maintained while performing calculations, i.e. to assign a label to a user, only the labels of users who he/she is following are taken into account.

To generate the test dataset, we collected the current follower information for 2000 users, each of have started following at least 3 new users (who had accounts prior to June 2009 and exist in our dataset).

We use a 55 node Hadoop cluster [8] for performing community detection as well as for generating ranked list of users to be followed for each community. Each machine in the cluster is a 2.66 GHz Quad Core Intel Xeon with 16 GB Memory.

For performing classification for influential users, we create a training set of 100 thousand users, validation set of another 100 thousand users and test set of 500 thousand users. We vary the ratio of positive instances (users who are following) to negative instances (users who are not following) and parameter C. We also take precaution of removing the influential user who is being predicted from the feature set.

6. Community Detection Results

A single iteration for complete network took only 7 minutes, which includes time taken for shipping the labels to all nodes in the cluster. Due to limitation of the space provided here, all results as well as the code used in the paper have been made available on the supporting website [1].

6.1 Results for the complete network

Due to existence of users with more than a million followers, after the first iteration a large proportion of the users following an influential user end up in the community of the influential user. We found by end of 10^{th} iteration a giant community of 31 million users having "MrsKutcher" as label/user. There were few other communities, e.g. "buzztter" (a Japanese user) and "marcelotas" (a Brazilian journalist). Out of 556,279 communities detected only 1065 contained more than thousand users. This chaotic behavior of community detection algorithm supports the need for removal of influential users prior to community detection.

Label / User	Size	Related Nationality
chrishasboobs	10882151	United States of America
monovolume	1139344	Brazil
riskaydrama	1037555	United States of America
bcb8	305996	India
33	137553	Japan

Table 1. Top 5 Communities after 15th iteration for the network without influential users

6.2 Results for network without influential users

For network without influential users, we find that formation of mega communities (communities with million+ users) is significantly delayed, they occur after ~10 iterations. Such mega communities are byproduct of the epidemic nature of the algorithm. Several modifications have been proposed to overcome this limitation [5].

We find that the mega communities tend to represent distinct nationalities and are thus actually useful. As illustrated by Table 1, we find a large community with label "chrishasboobs" (an abandoned account of an internet celebrity). After randomly selecting ~100 users and inspecting their profiles, we found that this community contains mostly American users. The second largest community is similarly found to contain users primarily from Brazil. The third community is also found to contain mostly American users. The next two communities contain Indian and Japanese users, other countries such as Netherlands, France, Portugal, etc. also occur in distinct communities. Since the algorithm is stochastic in nature, we compared results across multiple runs and found similar distribution of users in difference communities representing nationalities.

For lower number of iterations, e.g. after 7th iteration, we get communities which represent special interests, employers,

universities and geographical locations such as cities. By looking for membership of few select users we could find communities of employees of a tech company, users associated with W3C and Semantic Web, users who do research or are associated with MIT Media Lab.

You further explore these results in [1].

7. Results for Link prediction using Community Detection

Method	Recall
Baseline	4.7%
1st iteration	2.5%
4th iteration	4.7%
5th iteration	5.6%

Table 2. Recall for Link Prediction

The above table shows the performance of the algorithm. Note that the low recall values are due to hard nature of the problem. Consider that 1000 users out of 2000 users in the test set doubled their social network. Thus it is extremely difficult to predict who they started following using information about previously followed users.

Since the baseline results are also lower, it shows that the large number of users who were added later lie outside the usual two degrees of separation. Such connections are extremely hard to predict, using the network information alone. In many cases we find that the new connections lie outside the area of interest of the community, i.e., a user might be part of a community involving his university friends, however, the new users which he started following would belong to some special interest group or say may be colleagues at the company he is employed. These changes are extremely hard to predict.

8. Results for follower prediction for influential users

Validation Set Results

C	Accuracy	Precision	Recall	ROCArea
1	72.83	71.39	76.20	80.20
10	73.56	71.08	79.43	79.79
100	71.61	70.44	81.35	78.77

Size of negative set = Size of positive set.

C	Accuracy	Precision	Recall	ROCArea
1	84.59	84.01	28.35	83.63
10	85.18	82.29	33.02	84.15
100	85.32	80.48	35.11	83.49

Size of negative set = 4 * Size of positive set.

C	Accuracy	Precision	Recall	ROCArea
1	90.21	88.62	13.68	84.19
10	90.66	83.78	19.81	84.68
100	90.78	80.54	22.45	83.64

Size of negative set = 8 * Size of positive set.

Table 3: Variation of results with C and size of negative set

We considered users with more than 500 thousand followers. Above results are for Ellen Degeneres who at present has 5 million users. We constructed a simple classification problem as described earlier.

As we can observe from the results on the validation set, the recall drops as we increase the proportion of the negative set, but at the same time the ROC area increases. For test set, we use negative set with size around 8 times the size of the positive set (We also find that this is the proportion is the one observed for users with highest number of followers. Also, we are still investigating the effects of bigger size of Negative set).

Keeping the size of Negative Set 8 times the size of Positive Set and keeping the value of C as 10, we evaluated the results on the test set:

Accuracy	Precision	Recall	ROCArea
90.65	83.60	19.75	84.81

Table 4. Results on Test Set for optimal value of C and Size of Negative set a chosen by validation set

Given a user not following an influential user, there are two cases possible:

1. The user knows about the existence of the influential user and yet he is not interested in following him.
2. The user is not aware of existence of the influential user (on the social network) and thus he is not following him.

The second case should appear as a False Positive while predicting using above method. Since a false positive would imply that the algorithm predicts that the user might be interested, but the user is unaware of the presence and thus influential users are suitable candidate for recommendations.

9. Results using S.V.D. based model

We used a community of 10871 users living in Cleveland, Ohio. The size of the sparse matrix created was 10871× 60168 which implies that 10871 users were following 60168 users.

We generated two test sets; in the first test set, for each user who is following at least 10 users, we removed one edge and added it to test set, and we also created an edge between the same user and a randomly selected user from 60168 users. This led to a balanced test set containing 7750 edges.

For the second test set for each user who is following at least 20 users, we removed one edge and added it to test set, and we also created an edge between the same user and a randomly selected

user from 60168 users. This led to a balanced test set containing 4712 edges.

Note that not only the sizes of the two test sets are different, but also in first test set (the larger one), we are even removing an edge from each user having just 10 edges. Thus this set is considerably harder to predict.

We varied the number of principal components starting from 1 and then in increments of 25 upto 125. We measured both ROCArea well as Classification error.

The results are described in the table below:

Test set of 4712 edges		
Number of Principal Components	ROCArea	Classification error
1	70%	30%
25	83%	21%
50	83%	20%
75	82%	20%
100	80%	22%
125	79%	22%

Table 5. Variation of AUC and error by number of principal components for test set of 4712 edges

Test set of 7750 edges		
Number of Principal Components	ROCArea	Classification error
1	70%	34%
25	79%	24%
50	79%	24%
75	78%	24%
100	78%	24%
125	77%	24%

Table 6. Variation of AUC and error by number of principal components for test set of 7750 edges

As observed from above two tables, the S.V.D. model is capable of distinguishing between a randomly created edge and a real edge. Thus it might be possible to use such models for recommending new edges and thus in turn, users to follow. We also find that as number of principal components increase there is increase in the discriminative capability as measure by both ROC Area as well the classification error. However adding more than 50 principal components lead to decrease in the discriminative capability. This may be due to the fact that it is now over fitting the matrix and the calculated A' is now closer to A rather than being an approximation. It might also be an effect of the algorithm used (Lanczos) used to perform the S.V.D. and the 50+ principal components that contain mostly noise. However, by using just 25 principal components we could achieve ROCArea of 83% and 79% for the smaller and the larger test sets respectively.

10. Discussion & Conclusion

In this project we evaluated three different approaches for performing link prediction/recommendation for a real world large scale online social network. Except for the unbalanced classification model based approach which is suitable only for Twitter like networks where there are extremely influential users, the other two approaches are scalable and applicable for any social network. The approach using only communities and enumeration of all users followed by members of the community is shown to be useful considering low computational costs involved. However the S.V.D. based model is also shown to be useful. In future we hope to compare the two approaches using similar sets of users.

11. ACKNOWLEDGMENTS

We thank Prof. Ashutosh Saxena, for his guidance and support during the entire course of the project.

We thank Haewoon Kwak, Changhyun Lee, Hosung Park, Sue Moon, J. Yang. & Jure Leskovec for providing the dataset used in this paper.

The cluster used is funded in part by National Science Foundation grants CNS-0403340, SES-0537606, IIS 0634677, and IIS 0705774.

12. REFERENCES

[1] Supporting Website http://www.akshaybhat.com/LPMR

[2] D. Ediger, K. Jiang, J. Riedy, D. A. Bader and C. Corley. Massive Social Network Analysis: Mining Twitter for Social Good. In *ICPP '2010Parallel Processing, International Conference on*, pp. 583-593, 2010

[3] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a social network or a news media?. In *WWW '10: Proceedings of the 19th international conference on World Wide Web*, pages, 591-600.

[4] J. Leskovec, K. J. Lang, and M. Mahoney. 2010. Empirical comparison of algorithms for network community detection. In *WWW '10: Proceedings of the 19th international conference on World Wide Web*. pages, 631-640.

[5] I.X.Y. Leung, P. Hui, P. Liò and J. Crowcroft. Towards real-time community detection in large networks. *Physical Review E*, 79:066107, 2009.

[6] U.N. Raghavan, R. Albert and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76:036106, 2007.

[7] http://svmlight.joachims.org/svm_perf.html

[8] http://www.infosci.cornell.edu/hadoop