Community Detection on 2-Hop Topology

HU, Pili and LI, Yichao

December 18, 2011

Abstract

Community detection in social networks has drawn great interest in recent decades, especially inspired by the emerge of wide online social network sites. In this paper, we study the two-hop topology from several observers in the "renren.com" network. Some important statistics are calculated first, along with some visualization. Then we compute mining(results) —**NEED-REVIEW-HERE**—

This document serves as the report of CSCI5180 course project[5, 6]. For more information, codes, and data, please refer to our open source repository[1].

Contents

1 Introduction

1.1 Motivation

Community detection has drawn great interest in the near decade. The explosion of online Social Network Sites make the study more practical but challenging. Traditional methodology aims at optimizing certain global metric. The resultant graph partition can thus be regarded as communities. The motivation of detecting communities on 2-hop topology comes in two folds:

1. **Data Availability**. The global topology of those SNS is kept confidential. However, users are allowed to view their buddies profile by default. This gives us the chance to construct a 2-hop subgraph from the whole graph. Take a step further, outsiders can always observe some forwarding sequences of tweet/status. This helps to reconstruct a subgraph more than 2-hop from the observer. The methodology developed here can thus be extended to such situations.

2. **Computation Availability**. Although those SNS providers have access to the global topology, it is computationally intractable to perform some well-developed global optimization. Local heuristics are widely used in many applications.

1.2 Application

The potential application may be:

1. **Friend recommendation**. This is one of the most important applications of SNS. Traditional ways try to gather other information first, like school, company, etc. Then the program recommends friends based on this information.

2. **Targeted advertisement**. Wide-sense community can represent a group of people sharing the same interests. Their consumption pattern may be similar. Detecting such "communities" in the context where nodes are not directly labeled(like forums, microblogs, etc) help social media advertisers to improve their lifting ratio.

1.3 Contribution

The contribution comes in three folds:

1. Output a crawler specialized for "renren.com".

2. Extract 9 proximity measures from 2-hop topology. Those meta tools can work on 200,000 edges in the order of seconds. They can facilitate future study.

3. Train several Multi-Layer Perceptron(MLP)s to combine those proximity measures.

Data limitation(only 2-hop topology without content information) distinguishes our work from previous work. This is also the most challenging part in our study.

1.4 Problem Definition

Given one observer node in an SNS network, and all nodes within 2-hop from the observer, together with the links, try to predict whether a node is in the same community with the observer. If so, the class label is set to 1, otherwise 0.

2 Data Preparation and Preprocessing

2.1 Data Source

The data source used in this paper is "renren.com", which is the most popular SNS in mainland China. At the mean while, "renren.com" has the following good characteristics:

- Web UI of "3g.renren.com" is very clean. Cawling through this interface doesn't require any API calls.
- Renren is a real name dense graph connected mainly through real communities. Intuitively speaking, people who are close to the observer should be observed within this 2-hop topology with high probability.
- Institution label for all nodes in the observed subgraph is available on "renren.com". It makes model validation easier and automated. If the model is proven effective, we can adapt it to other SNS's, where institution label may be not available.

2.2 Crawler

The crawler is developed using the following components:

- `perl`
- `wget`
- `bash`

We fake HTTP login and page requests like an ordinary user. Random delay between two adjacent requests is introduced to mitigate the automation pattern.

For more details, interested readers are recommended to our project [1]. The crawler is under directory 'crawler', together with detailed instructions on how to deploy one yourself.

2.3 Preprocessing

The preprocessing steps is as follows:

1. Parse crawled web pages to get user ID and institution name.
2. Create one vertex for each person crawled.
3. Create one edge for each friend relationship.
4. Do isomorphic tranformation on the original graph [9]. Vertex number and insitution label is shuffled according to a psudorandom sequence. This is to protect privacy of our volunteer observers.

After preprocessing, we get the raw data, which is in the following format:

```
---'link'---
NodeID1 '\t' NodeID2
NodeID1 '\t' NodeID3
...

---'node'---
NodeID1 '\t' Label1
NodeID2 '\t' Label2
...
```

To facilitate our model training and evaluation, we divide our data into two parts randomly and equally: one training set and one testing set.

3 Notations

The notations used throughout this paper are gathered in table(1).

4 Statistics

5 Feature Selection

5.1 Random Walk Based Techniques

The rationale behind random walk is: the nearer a node is to observer, it can be reached with a higher probability by a random walker starting from observer. PageRank is one of the most classical work in this field.

Table 1: Notations

Symbol	Explanation
n	node number
m	edge number
A	adjacency matrix
A_{ij}	1 if node i and node j are directly connected, 0 otherwise
$N(i)$	neighbourhood of i, namely $N(i) = \{j \mid A_{ij} = 1\}$
$d(i)$	degree of node i $d(i) = \sum_j A_{ij}$
o	observer
$\overrightarrow{v}$	ranking vector of nodes
l_i	the real label of node i
L_i	the predicted label of node i

We shortlist three variations[2] of the original PageRank, together with the explanation of using them in our context:

- PageRank with escape probability:

$$\overrightarrow{v} = \alpha A^{\mathrm{T}} \overrightarrow{v} + (1-\alpha)\frac{\overrightarrow{1}}{n} \tag{1}$$

  where A is the transition matrix, typically the adjacency matrix, and α is the transfer ratio. $\overrightarrow{v}$ is the resultant PageRank value.

  In our graph, the link is very sparse for edge node, which represents the friend of observer's friend. They provide many dangling links, which influences the outcome of original PageRank algorithm. These nodes may appear as probability sinks. By introducing in escape probability, such sinks can be eliminated.

  Our graph is rooted and thus biased at the observer. So the rank output by this algorithm can be a proximity measure between other nodes and observer.

- Personalized PageRank:

$$\overrightarrow{v} = \alpha A^{\mathrm{T}} \overrightarrow{v} + (1-\alpha)\frac{\overrightarrow{b}}{||\overrightarrow{b}||_1} \tag{2}$$

  The difference from eqn(1) is that, the all one's column vector $\overrightarrow{1}$ is substituted by a more general weight vector, also called the escape vector. This vector describes where and by what probability the random walker will escape to.

  With different escape vector $\overrightarrow{b}$, we can reach different goal:

– Unsupervised learning. Denote the subscript of observer as o, then:

$$b_i = \begin{cases} 1 & i = o \\ 0 & i \neq o \end{cases} \tag{3}$$

That means the random walker will always escape to the root node. Then the output can measure the proximity between root and other nodes.

– Semi-supervised learning. Denote the labeled set as L and their label as l_i. The personalized escape vector can be:

$$b_i = \begin{cases} l_i & i \in L \\ 0 & i \notin L \end{cases} \tag{4}$$

Typically, the miner can choose a subset of the nodes, which are already known to share the same label with the observer. We assign those nodes in set L some positive weights. Then the random walker can escape to this set with different probability. The rantionale is, if one node is in our community, it is probable to be reached from some already known members.

5.2 Simple Proximity Measures

Yet random walk based techniques are good proximity measures in many context, there exists some simple but effective measures. We shortlist the following metrics, together with explanation of using them in our context:

- Common Neighbours:

$$Common(i, j) = |N(i) \cap N(j)| \tag{5}$$

It is straight forward that if node i shares more common neighbours with observer o, it is closer to observer.

- Adamic/Adar score:

$$Adamic/Adar(i, j) = \sum_{k \in N(i) \cap N(j)} \frac{1}{\log |N(k)|} \tag{6}$$

Adamic/Adar can be seen as an extension of simple common neighbours, which take the neighbour's degree into consideration. The contribution from each common neighbour k is weighted as $\frac{1}{\log |N(k)|}$. Higher degree nodes may stant for public pages, or very well-known people in the network. Sharing such kind of common neighbour is a much weaker evidence that two people are in the same community. Thus the it is given a lower weight. The use of log scale stems from previous statistical studies, that node ranking tends to be Zipf distributed. [3]

- Jaccard's coefficient:

$$J(i,j) = \frac{|N(i) \cap N(j)|}{|N(i) \cup N(j)|} \tag{7}$$

  This coefficient can be seen as another extension of common neighbours. It is effective when the graph is sparse. The numerator calculates the common neighbours of two nodes. The denominator takes their own size into consideration. It's reasonable that two nodes with higher degree will have more common neighbours. The denominator act as a normalizer, which makes the output of different node pairs relative comparable.

5.3 Configuration of 9 Features

Simple proximity measures mentioned in section(5.2) does not require any configuration. As to PageRank(PR) series, there are basically three configurations:

1. Transfer ratio: α. This determines how much weight a node transfers to its neighbours, while $(1-\alpha)$ determines what's the probability that a random walker escape.
2. Escape vector: $\overrightarrow{b}$. By setting different escape vectors, we can get different variations of Personalized Page Rank(PPR) mentioned above.
3. Dangling link problem. When our data is crawled, the original link is unidirectional. The leaf nodes may cause leakage of weights. There are two methods to deal with this problem:
   - Introduce a super node. Every node is connected to this node, and vice versa. In the following study, this pre-processing is denoted as "(+SN)".
   - Add reverse edges of all links. This makes all links bidirectional. It will guarantee no leakage naturally. Without any note, we refer PR to this version of pre-processing.

α is set to 0.9 in all of our experiments. Finding the best α is left as future work.

The abbreviations we used to notate those features are gathered in table(2).

6 Single Feature Evalutaion

6.1 Quality Functions

After we label the community of all nodes, we can calculate some quality functions to indicate how well the community is detected. Different

Table 2: Abbreviations of Features

No.	Abbrevation	Explanation
1	Common Neighbour	see section(5.2)
2	Ademic/Adar	see section(5.2)
3	Jaccard's	see section(5.2)
4	PR:EV=All 1's	Escape vector is set to all 1's. This means uniform restart when the walker escpases
5	PR:EV=High 3	Pick up 3 highest degree nodes in the same community from training set, and set them to 1. Others, 0.
6	PR:EV=Root	Only root node is set to 1, others 0.
7	PR:EV=All 1's(+SN)	Like 4, with SN pre-processing
8	PR:EV=High 3 (+SN)	Like 5, with SN pre-processing
9	PR:EV=Root(+SN)	Like 6, with SN pre-processing

researchers have different preference of quality functions, to name a few:

- Normalized cut:

$$Ncut(S) = \frac{\sum_{i\in S, j\in \overline{S}} A(i,j)}{\sum_{i\in S} d(i)} + \frac{\sum_{i\in S, j\in \overline{S}} A(i,j)}{\sum_{j\in \overline{S}} d(j)} \tag{8}$$

  where S is the set of one community, and $d(.)$ is the digree of a node, which is defined as $d(i) = \sum_j A_{ij}$.

  The smaller the value, the better community detection quality. Two numerators measure the number of inter community links, and the value is normalized by the number of intra community links.

- Conductance:

$$Conductance(S) = \frac{\sum_{i\in S, j\in \overline{S}} A(i,j)}{\min\{\sum_{i\in S} d(i), \sum_{j\in \overline{S}} d(j)\}} \tag{9}$$

  This quality function is positive related with eqn(8). When community sizes are highly skewed, conductance may provide better evaluation.

- Modularity:

$$Q = \sum_{i=1}^{K} \left[\frac{A(V_i, V_i)}{m} - \left(\frac{d(V_i)}{2m} \right)^2 \right] \tag{10}$$

  where

$$A(V_i, V_j) = \sum_{u\in V_i, v\in V_j} A_{uv} \tag{11}$$

and

$$d(V_i) = \sum_{u \in V_i} d_u \tag{12}$$

K is the number of communities, which is 2 in our case (we only distinguish whether the node is in the same community with observer or not).

The rationale behind modularity is the comparison with a random graph. The father resultant community is from a random graph, the better the quality. "One of the advantages of modularity is that it is independent of the number of communities the graph is divided into"[2].

Those quality functions are popular among different research groups. They also capture different characteristics of graphs. In this project, we'll check if these global metric can be used to evaluate our extreme local case.

6.2 Receiver Operating Characteristics

The ROC curve plots True Positive Ratio v.s. False Positive Ratio [7], when some parameters are changed.

In this study, we evaluate each single feature's ROC by scaling a threshold from smallest proximity measure to largest proximity measure. Since all of our proximity measures are similarity meausres, we proceed according to the following rule:

1. Set a threshold of T.

2. For each node i, judge its proximity measure P_i. Set predicted label of i:

$$L_i = \{ \begin{array}{ll} 1 & P_i > T \\ 0 & P_i < T \end{array} \tag{13}$$

3. Calculate TPR:

$$\text{TPR} = \frac{|\{i : L_i = 1 \text{ and } l_i = 1\}|}{|\{i : l_i = 1\}|} \tag{14}$$

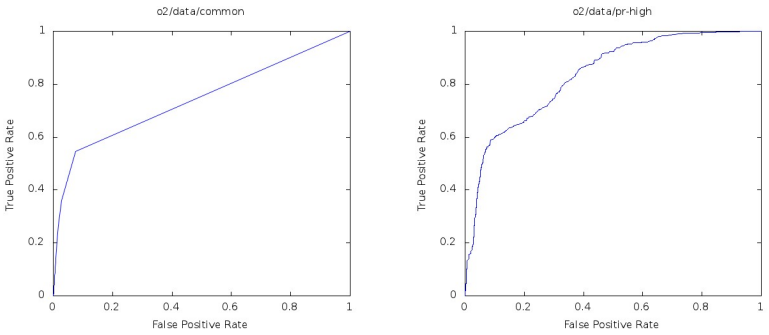
4. Calculate FPR:

$$\text{TPR} = \frac{|\{i : L_i = 1 \text{ and } l_i = 0\}|}{|\{i : l_i = 0\}|} \tag{15}$$

Note that a naive implementation of this ROC algorithm requires $O(n^2)$ operation: first level iteration is to enumerate an T; second level iteration is to classify every ponit according to T(and calculate TPR and FPR at the same time).

We propose one efficient algorithm to calculate this kind of threshold based ROC, which runs in $O(n)$ time. The algorithm is put in Appdendix due to space limit.

We select observer 2 and draw 9 ROC curves. Two candidate plots are selected in fig(1). For a full gallary of 9 plots, please refer to Appendix.



(a) Common Neighbours (b) PR: EV=High 3

Figure 1: Selected ROC Curves

Our observations of those ROC curves are listed below:

1. All ROC curves blend above the line TPR=FPR. This means, those proximity measures can reflect similarity between observer and other nodes. They are absolutely better than random guess.

2. All ROC curves have a switching point, which locates at the low FPR region. Before this point, the curve increases sharply, after this point, the tangent of the curve decreases. This means, those proximity measures alone can be used to detect community with high precision when T is large. They can hardly distinguish nodes with low proximity measure.

3. Among all the 9 curves, PageRank with escape vector set to 3 highest degree nodes performs the best. See, fig(1(b)). The area under this curve is the largest. This proves the efficacy of our proposed semi-supervised version of PPR. The introduction of high influential nodes in the same community can help to detect other members.

7 Model Training and Testing

From the above analysis, we notice that although those proximity measures reflect community information to some degree, none of them can provide perfect ROC alone. In this section, we train Multi-Layer Perceptron using weka[8], aiming at combining those features and outputing better results.

7.1 MLP with Full Topology

Our first experiment is to train an MLP with full topology. Let observer 2 be an example, the confusion matrix on testing data is shown in table(3).

Table 3: Confusion Matrix: MLP, Observer2, Full Topology

a	b	← **classified as**
1812	2915	$a = 1$
1212	38180	$b = 0$

The precision is 0.599 and recall is 0.383. The overall accuracy is 0.906. Although the overall accuracy is very high, we can conclude that this classification is of low application value. The True Negative population is very large, which means our trained MLP tends to classify nodes into class 0.

One of our targeted use of the predicted labels can be friend recommendation. It's nonsense to recommend a lot of friends at a time. So our application is not sensitive to recall rate. Thus we work towards higher precision in the successive experiments, and at the same time avoid making the predication trivial.

7.2 MLP without Degree 1 Nodes

—NEED-REVIEW-HERE—

According to our previous statistics, large amount of nodes are of degree one. This doesn't necessarily mean the real degree, but means the degree within our observation. The 2-hop topology is so limited that we see many leaf nodes.

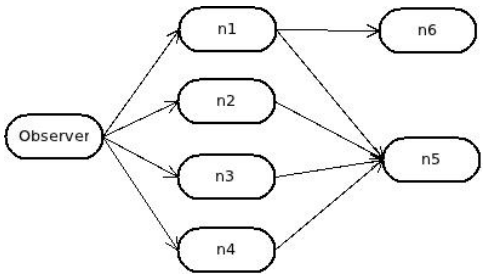


Figure 2: Illustration of Two Levels

Fig(2) illustrates one part of the 2-hop topology. The real friend relationship is bidirectional. The arrows in the illustration just represents the direction of our crawler. Nodes n5 and n6 stand for two typical situations of those level 2 nodes. It's very probable that n6 with single link from a level 1 node is not in the observer's community. On the contrary, n5, which many level 1 nodes point to is very likely to be in the same community as observer.

Nodes like n6 will bring up priori probability of class 0 as shown in the previous experiment. We eliminate all nodes with degree 1, and run the training stage again. Result confusion matrix is shown in table(4).

Table 4: Confusion Matrix: MLP, Observer2, without Degree 1

a	b	← **classified as**
1382	1205	$a = 1$
595	2348	$b = 0$

The precision is now 0.699 and recall is 0.534. Both metrics are improved, while the overall accuracy degrades to 0.675. Bearing our potential application in mind, this result is better, which means if we target 10 nodes from the predicted positive class, we may have a chance of 70% to hit observer's community.

7.3 MLP without Degree 2

Take the last experiment further, it's rational to think whether eliminating more nodes can help.

We train the same model after eliminating degree 2 nodes. The result confusion matrix is shown in table().

Table 5: Confusion Matrix: MLP, Observer2, without Degree 2

a	b	← **classified as**
1706	23	$a = 1$
963	88	$b = 0$

The precision is now 0.639 and recall is 0.987. Though we get a very good recall rate, the precision is lowered. Looking into the confusion matrix, we find this classification becomes **trivial**. It tends to predict class 1. Even a naive guess that classifies all nodes into class 1 can reach the precision of 0.62194 and recall 1.0.

With more lower degree nodes being eliminated, we may think this problem becomes more significant. Since the priori probability of class 1 increases, the naive guess precision can increase by keeping a 100% recall.

7.4 MLP with Only Level 1 Nodes

If we limit our mining scope to only level 1 nodes, the application can not be friend recommendation, since they have alreay established one link with observer. Nevertheless, taking a look at the result is of great interest. table(6) shows the result confusion matrix.

The precision is 0.813, and the recall is 0.685. If we aim at the "community" defined as a group of people sharing the same interest, rather than

Table 6: Confusion Matrix: MLP, Observer2, Only Level 1 Nodes

a	b	← **classified as**
74	34	$a = 1$
17	81	$b = 0$

those institutions, this result suggests targeted advertisement can be performed effeciently.

8 Conclusion

This paper shows 2-hop link topology contains some information, though it is limited. We highlight the following observations from the study:

1.

9 Future Work

As is in our proposal [1], there are other metrics we want to try. Due to project time and report space, we only name a few in this section:

- Katz Score:

$$Katz(i,j) = \sum_{l=1}^{\infty} \beta^l A^l(i,j) \tag{16}$$

  The matrix series results in

$$Katz = (I - \beta A)^{-1} - I \tag{17}$$

  By tuning β, this score can measure number of paths between two nodes, with preferred length range.

- Simrank:

$$s(a,b) = \frac{\gamma}{|I(a)||I(b)|} \sum_{i \in I(a), j \in I(b)} s(i,j) \tag{18}$$

  This metric measures the expectation of γ^l, where l equals the time when two random walkers start from a and b meet. By proper matrix construction, Simrank can be solved as an eigenvalue problem.

For more information, interested readers are recommended to [2].

The links used in this study is mere static topology. With 2-hop limit, the information can be extracted is not abundant. In terms of data source, we propose two promising directions:

1. Dynamic link information. "renren.com" is a very dense graph. However, many edges are not effetive in the sense that little information flow through them. We can build links using user interactions like status forwarding, blog reply, etc.

2. Content based prediction. Note that links can only capture static topology or dynamic interaction, but not content. Nodes in a well-defined community share similar content with very high probability. Combining topology and content may be a promising direction.

Similar thoughts have been proposed in many related literature, to name a few[4, 2].

The variations of PageRank have great potential. We shortlist a few possible directions:

1. What's the most appropriate transition ratio α.

2. How many influential nodes should be introduced in the semi-supervised version of PPR, considering both cost and accuracy.

Acknowledgements

The authors would like to thank the following people for their contribution of personal data: Bo WANG, Hongshu LIAO, Huan CHEN, Shouxi LUO, Xiaoyu LUO, Xinyue ZHENG, Yan WANG, Yi LUO. The authors appreciate brain storm discussion with professor Wing Lau and Deyi Sun. The authors would like to thank course instructors and TAs of CSCI5180.

References

[1] Hu Pili and Li Yichao. Community detection on 2-hop topology. GitHub, https://github.com/Czlyc/2C-Web-Research, 12 2011. course project of CUHK/CSCI5180.

[2] C.C. Aggarwal. *Social network data analytics*. Springer-Verlag New York Inc, 2011.

[3] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 126–134. IEEE, 1999.

[4] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.U. Hwang. Complex networks: Structure and dynamics. *Physics reports*, 424(4-5):175–308, 2006.

[5] CSCI5180, Data Mining Lecture Notes, http://www.cse.cuhk.edu.hk/~lwchan/teaching/csc5180.html

[6] CSCI5180, Data Mining Tutorial Notes. (available in Moodle. Please contact the teacher/TA if you have interest)

[7] Receiver Operating Characteristics, http://en.wikipedia.org/wiki/Receiver_operating_characteristic

[8] Weka, http://www.cs.waikato.ac.nz/ml/weka/index.html

[9] Wikipedia, Isomorphic Graph, http://en.wikipedia.org/wiki/Graph_isomorphism

Appendix

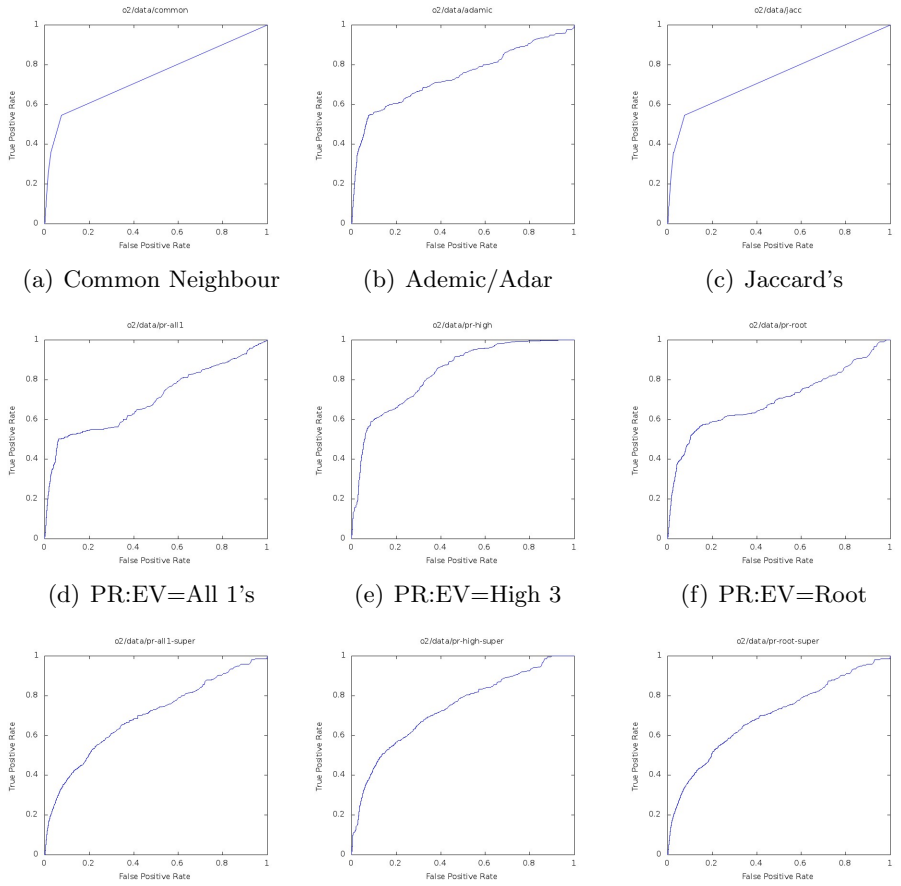
ROC Algorithm

This algorithm runs in $O(n)$ time, where n is the number of data points. This algorithm is applicable where TP and FP is monotonic w.r.t the T threshold.

```
FUNCTION (Nodes[])
        calculate CountPositive, CountNegative
        T = -infinite
        TP = CountPositive
        FP = CountNegative
        sort Nodes according to proximity measure.
        FOR i = 0 ; i < CountTotal ; i ++
                IF T < Nodes(i).proximity THEN
                        OUTPUT T, TP / CountPositive, FP / CountNegative
                END IF
                T = Nodes(i).proximity
                IF Nodes(i).label == 1 THEN
                        TP --
                ELSE
                        FP --
                END IF
        END FOR
END FUNCTION
```

ROC Curve of Observer 2

ROC curves of 9 features are shown in fig(3). It's obvious that PageRank with escape vector using 3 highest degree nodes and without super node performs the best alone, see fig(3(e)).



(a) Common Neighbour (b) Ademic/Adar (c) Jaccard's

(d) PR:EV=All 1's (e) PR:EV=High 3 (f) PR:EV=Root

(g) PR:EV=All 1's(+SN) (h) PR:EV=High 3 (+SN) (i) PR:EV=Root(+SN)

Figure 3: ROC Curve of 9 Features

Declaration

Meta tools like PageRank, Adamic/Adar, etc, are learned from the book *Social Network Data Analytics*. For original sources of these algorithms, please refer to the bibliography pages of that book.

The analysis/adaptation/extension/application of these tools and the use in community detection on 2-hop topology mentioned in this document is originality. Please refer to our open source repository for citation issues.

The data in our project repository is limited to research use only. The authors of this paper own the transformed version. You are welcome to use them in your study but remember to make a notification to the authors first. Our contact information is given in the project 'README' file.