Community Detection on 2-Hop Topology (short version)[1]

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 extract 9 features from the 2-hop topology. All features are evaluated alone using ROC first. Then we train multiple MLP models under different settings.

The most effective features in our study is PageRank with escape vector to several in-community high degree nodes. By eliminating leaf nodes(degree=1), we can get a prediction precision 0.699 and recall 0.534. The confusion matrix shows this result is not trivial.

This document serves as the report of CSCI5180 course project[5, 6]. It is the short version which fits into the 8-15 page requirement. For more information, full version, codes, and data, please refer to our open source repository[1].

Contents

1 Introduction **2**
- 1.1 Motivation . . . 2
- 1.2 Application . . . 3
- 1.3 Contribution . . . 3
- 1.4 Problem Definition . . . 3

2 Data Preparation and Preprocessing **3**
- 2.1 Data Source . . . 3
- 2.2 Crawler . . . 4
- 2.3 Preprocessing . . . 4

3 Statistics **4**

4 Topology Visualization **5**

5 Notations **7**

6 Feature Selection **7**
6.1 Random Walk Based Techniques 7
6.2 Simple Proximity Measures 8
6.3 Configuration of 9 Features 9

7 Single Feature Evaluation **10**

8 Model Training and Testing **12**
8.1 MLP with Full Topology 12
8.2 MLP without Degree 1 Nodes 12
8.3 MLP without Degree 2 13
8.4 MLP with Only Level 1 Nodes 14

9 Conclusion **14**

10 Future Work **14**

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:

- *This list is cut because of space limit. Please see [1] for our full version.*

2.2 Crawler

This section is cut because of space limit. Please see [1] for our full version.

Crawler codes are availabel in our open source repository.

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 Statistics

Basic statistics are shown in table(1).

The number of nodes, number of level 1 nodes, and number of links are positive related.

We plot observer2's nodes whose degree is greater than or equal to 5. fig(1) shows the degree v.s. rank in log log scale. We can see that the curve is not a linear line, as appears in most Zipf like distribution scenario. The curve is super linear, which means it is more skewed. This makes sense because we only have 2-hop topology from the observer. Most edges of level 2 nodes are not present in our data.

Table 1: Basic Statistics

Observer Seq.	# of All Nodes	# of Level 1 Nodes	# of Links
1	52595	357	83820
2	88238	446	139224
3	129019	535	204006
4	46049	224	75654
5	48258	260	85479
6	22060	120	29860
7	43086	181	60719
8	175237	733	367785

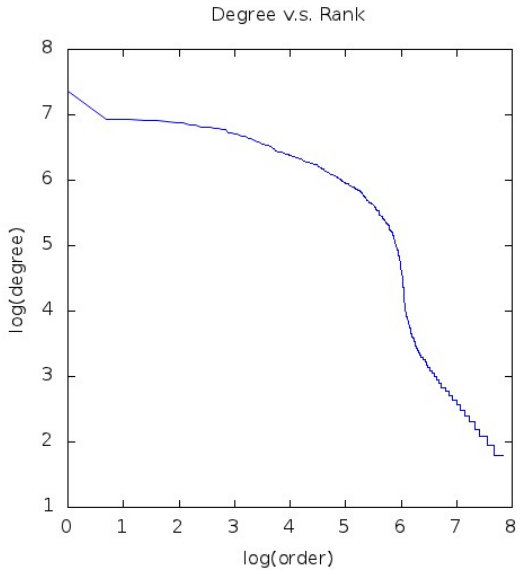


Figure 1: Observer2, Degree Distribution

4 Topology Visualization

In this study, we use NodeXL [10] to visualize our graph.

NodeXL is an open source Excel plugin. Though it is easy to use, the process ability is not high enough for our large graph(10w order of nodes and 20w order of edges). We do the following pre-processing to make NodeXL work:

1. Eliminate leaf nodes, whose degree is 1.

2. Sample edges in the rest graph.

Note that "renren.com" is a dense graph. A typical node will have more than 100 edges. This makes NodeXL hard to work. With random sampling, the topology can be visualized by NodeXL. fig(2) shows the result, where

pink nodes are in the same community with observer, while blue nodes are not.

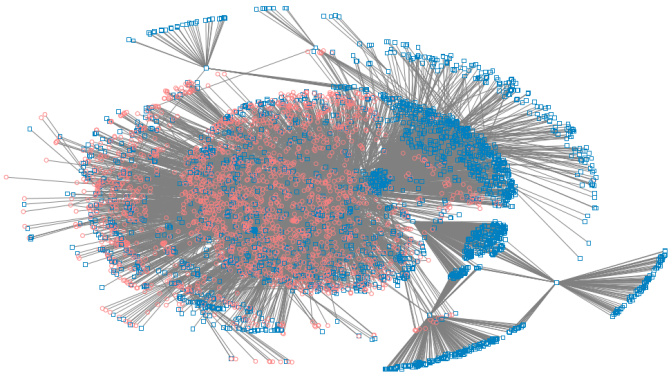


Figure 2: Topology Visualization

We extract all level one nodes and do the same visualization again. fig(3) shows the result. There are mainly three clusters(dense parts) on the graph. This is natural that our observer may belong to(or once belonged to) different communities. In the pre-processing stage, we simply label the observer using his/her largest community. Thus some very dense parts are marked blue in the plot.

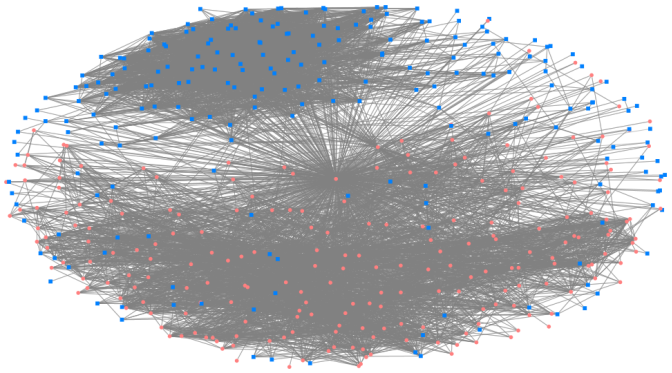


Figure 3: Topology Visualization, Level 1 Nodes Only

The pre-visualization result proves our problem is tractable. However, some true class nodes near the boundary of the graph lacks link information. They may lower our recall. On the other hand, there exist other false class nodes, which may correspond to popular public pages. Those nodes share large amount links with the community members. Those nodes may lower our precision. Note the blue dense parts that those history communities of our observer will have very high similarity measure. They may also cause low precision.

5 Notations

The notations used throughout this paper are gathered in table(2).

Table 2: 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

6 Feature Selection

6.1 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.

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.

6.2 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]. In our case, we have already shown that the degree v.s. rank is not a Zipf like distribution. Finding a better substitution of the log function remains a future work. In this study, we keep the original format of Adamic/Adar score.

- 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.

6.3 Configuration of 9 Features

Simple proximity measures mentioned in section(6.2) 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(3).

Table 3: Abbreviations of Features

No.	Abbrevation	Explanation
1	Common Neighbour	see section(6.2)
2	Ademic/Adar	see section(6.2)
3	Jaccard's	see section(6.2)
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

7 Single Feature Evaluation

Before we start complex mining, we want to examine the efficacy of those features we get. 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 measures, 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:

$$L_i = \{ \begin{array}{ll} 1 & P_i > T \\ 0 & P_i < T \end{array} \quad (8)$$

3. Calculate TPR:

$$\text{TPR} = \frac{|\{i : L_i = 1 \text{ and } l_i = 1\}|}{|\{i : l_i = 1\}|} \tag{9}$$

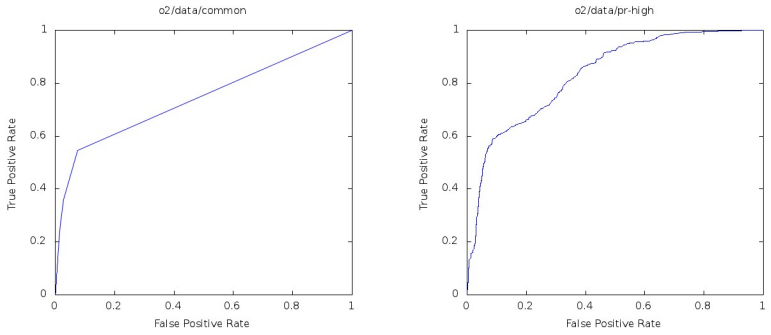
4. Calculate FPR:

$$\text{TPR} = \frac{|\{i : L_i = 1 \text{ and } l_i = 0\}|}{|\{i : l_i = 0\}|} \tag{10}$$

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 point 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 Appendix due to space limit.

We select observer 2 and draw 9 ROC curves. Two candidate plots are selected in fig(4). For a full gallery of 9 plots, please refer to Appendix.



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

Figure 4: 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(4(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.

8 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.

8.1 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(4).

Table 4: 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.

8.2 MLP without Degree 1 Nodes

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.

Fig(5) 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

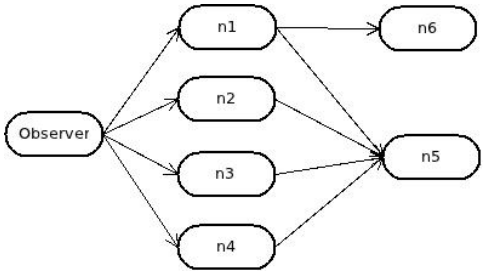


Figure 5: Illustration of Two Levels

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(5).

Table 5: 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.

8.3 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 6: 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.

8.4 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 already established one link with observer. Nevertheless, taking a look at the result is of great interest. table(7) shows the result confusion matrix.

Table 7: Confusion Matrix: MLP, Observer2, Only Level 1 Nodes

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

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

9 Conclusion

1. The most effective features in our study is PageRank with escape vector to several in-community high degree nodes.

2. By eliminating leaf nodes(degree=1), we can get a prediction precision 0.699 and recall 0.534. The confusion matrix shows this result is not trivial.

3. Community detection on 2-hop topology is difficult because of lack of information.

10 Future Work

This section is cut because of space limit. Please see [1] for our full version.

Acknowledgements

This section is cut because of space limit. Please see [1] for our full version.

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`

[10] NodeXL, `http://nodexl.codeplex.com/`

Appendix

ROC Algorithm

This section is cut because of space limit. Please see [1] for our full version.

ROC Curve of Observer 2

This section is cut because of space limit. Please see [1] for our full version.

Declaration

This section is cut because of space limit. Please see [1] for our full version.