Analysis of Social Networks based on Retweet Dynamics

Dongguo Zhang

Department of Electrical and Computer Engineering
The Ohio State University

Abstract

In this work, we build a visual analytic system to visualizing and analyzing data from Sina Weibo, which is the largest microblog service in China. This system has an offline data crawling part to collect retweet network data under some topic. Then the data is stored in the graph database, Neo4j graph database. The visualization part is implemented using JavaScript and D3.js library. During the observation of the retweet network, we find three modes of the retweet network. Furthermore, different centrality measurements are studied with the help of the visualization system. Their advantages and applying cases are shown in this work.

Key words: Weibo, Social network, Data visualization, Centrality.

1 Introduction & background

Microblog service such as twitter and Sina Weibo is popular. There are about 284 million active users on twitter. Over 400 million tweets are generated every day[1]. In China, twitter is censored. Internet users use Sina Weibo as their microblog services. There are about 46.2 million active users on Sina Weibo[2].

The microblog service is not only a place for users to communicate with their friends. It is also a new form of media. The microblog service is now a platform for marketing. Companies have their own accounts to increase the influence of their product. Singers and actors use the media to interact with their fans. The microblog service has also emerged as an important source of real-time news. News and rumors spread quickly through the microblog social network. Users post items quickly and reaches the audience in seconds. Furthermore, it is becoming a platform for public relations. Political movements can happen from it. For example, Twitter played an important role in socio-political events such as the Arab Spring and the Occupy Wall Street movement.

By studying how information spreads through the network, it is easier to control the information propagation. Companies can advertise more efficiently. Politician can let the political ideas known by more people. To control how the information spread, it is important to find the central nodes. The central nodes are users in the network who can lead information spreading quickly and widely. If the central nodes are willing to retweet your tweets, it is much more possible for your tweet to be popular and seen by other users

The retweet network on some topic is the main part to study. On twitter, retweet network is hard to build because the Twitter doesn't show the intermediate nodes. When retweeting, you can only retweet from the original tweet even if you see the tweet from your friends' retweet. So Twitter is not so good for understanding the retweet network. Instead, Sina Weibo is used to study the retweet network. It is a Chinese version of Twitter. And it shows all the intermediate users on one retweet link. On Sina Weibo, topic is shown by surrounding the keywords with "#". For example, "#Easter#" is a topic.

Related work

MicroBlog service is a hot area in social network study. Kumar et al. [3] designed TweetTracker to obtain crisis from near real-time tweets trending. To detect meme diffusion patterns in election-related microblogs, Ratkiewicz et al. [4, 5] used network analysis and visualization methods. Donghao Ren et al. [6] build a Sina Weibo Visual Analytic System to catch the weibo event. They collect the data by API and can only get the partial data.

My work

In this project, I build a offline Sina Weibo visualization and analyzing system. This system can collect the Sina Weibo data related to one topic by web crawler, store the data in the form of graph and calculate the centrality of the nodes. In this project, I study different kinds of centrality algorithm and implement them in the system. There is also a web interface to make it easier to understand the graph shape.

The report is organized as follows: Section II gives an overview about the related past work. Section II, the Sina Weibo Analytic system is introduced in detail. In Section III, graph structure and centrality analysis are discussed. Finally, conclusions are given in Section VI.

2 Weibo analytic system

2.1 Overview

Our system has four major components: a graph database for data storage and transaction solving, a web crawler for data collecting, a data analysis process to get the centrality and a web-based interface for data visualization.

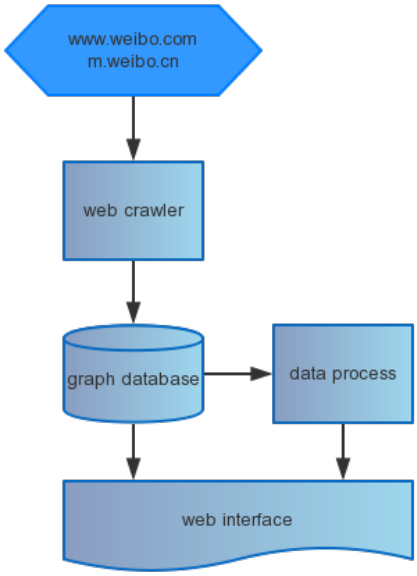

Figure 1 system architecture

The system architecture is shown in figure 1. The web crawler get the data from the www.weibo.com and m.weibo.cn. The data is stored in the graph database. The web interface can fetch the data directly from the database. And the data process can get the data from the database and calculate the centrality. The centrality can also be shown in the web interface.

2.2 Sina Weibo Crawler

2.2.1 Overview

Although Sina Weibo provides some open APIs for developers, these APIs are not enough for most developers. They have too much limitation on accessing numbers and time. In this project, we use web crawler instead of open APIs to fetch the data from Sina Weibo.

The web crawler technology is most used in search engine like Google and Yahoo!. They try to visit all the website on the internet and build indexes for them. In this project, web crawler copy the users' behavior to log in, search the Sina Weibo and read the content pages. When users visit the website, the bowser will send request to server for the HTML pages. In the program, the web crawler can send the same request to the server just like bowser. The overall process is shown in figure 2.

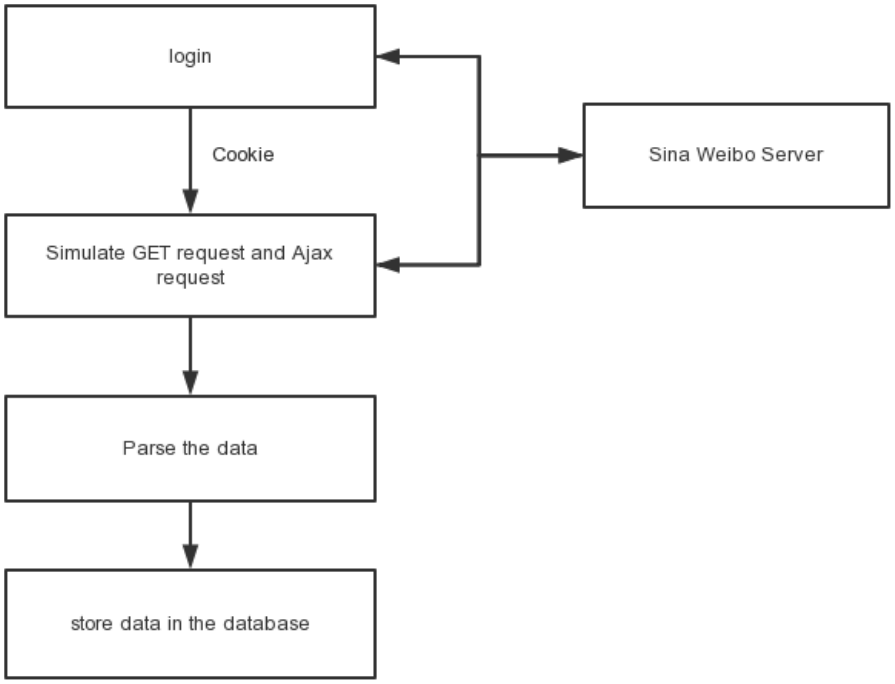

Figure 2 the overall process

2.2.2 Login

Cookie is the goal of the login process. It is widely used in nowadays website to keep a session active and to remember user state information. When user first visit one website, the server will also send back a piece of data, which is called cookie and stored in the browser. Every time the user loads the website, the browser sends the cookie back to the server to tell the website about the user's previous activity.

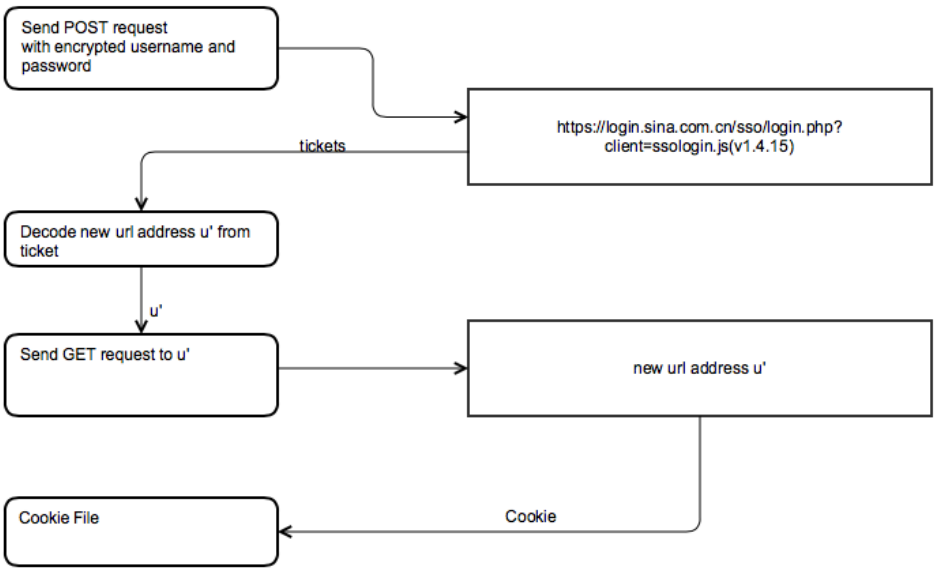

Figure 3 login proces

Sina Weibo login process has two steps as shown in figure 3. At first, Sending POST request with encrypted username and pass word to the login authentication address. The server will send back a ticket include another new url address. Secondly, Sending GET request to the new url address. Then the cookies will be sent back.

2.2.3 Fetch and decode data

In this project, tweets on one topic is collected to build the graph. To get all the tweets, Sina Weibo search function is utilized. Sina Weibo provides this function for users to search by keywords. If we search by "#topic#", we can get the tweets related to the topic.

There is a limitation on this function, which is set by Sina Weibo. At most 1000 tweets can be shown in the results. But the function support search under some time interval, which can be used to get all the tweets related to one topic. The smallest time interval is 1 hour. It means that we can search hour by hour to collect all the data.

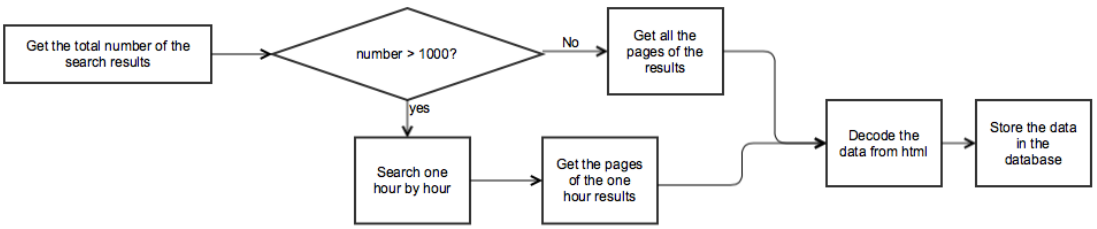

Figure 4 fetch and decode data

The overall progress is shown in figure 4. The crawler gets the first page of the search results and decodes the total number of search results. If the number of search results doesn't exceed 1000. The crawler can crawl all the results. Otherwise, the system needs to know the start time and end time to crawl the topic. Then it will search the tweets in one hour interval from start time to end time.

The decode part is mainly dealing with HTML page because the response is in the form of HTML. The page is the same as what we see when browsing the website. What we focus is the retweet link, user name and time. To decode the HTML, Jsoup library is used. It is a open source Java library to parse the HTML into DOM tree. Then we can select and modify the DOM tree as what we do with Javascript and CSS.

2.2.4 Sina Weibo anti-crawler strategy

Sina Weibo is a large website and has critical anti-crawler strategies. Those strategies are not known. To avoid those strategies, the crawler needs to be designed carefully and tested multiple times. After multiple tests to the Sina Weibo anti-crawler strategy, following strategies are found:

(1) One account cannot access the server too frequently. If the account visit the Sina Weibo per 3 second, then the account is required to enter the validation code after about 37 visits. The account can visit the website per 30 seconds with unlimited times.

(2) One account cannot log in to frequently. Otherwise, the account is required to enter the validation code when logging in.

In our system, we use following methods to walk around those anti-crawler strategies:

(1) Use random User-Agent for each request to the Sina Weibo server. In the HTTP request, User-Agent provides browser information and system details to the server. If one User-Agent access the server too frequently, the server will find it out.
(2) Use 10 accounts to balance the access intervals. The safe interval for one account is 30 seconds. To get the data faster, 10 accounts can be used alternately. So the interval can be reduced to 3 seconds.
(3) Store cookies into local file. Each time the user log in, one cookie will be assigned to the crawler. These cookies can be valid for a long time. So before logging in, check if the local cookies are still valid. If they are valid, use them directly. Logging in too often may lead to the account being blocked.

2.3 Data modeling

The data collected from the web is the retweet information. It is in nature a directed graph about social network including nodes and relationships.

2.3.1 Node

In the retweet network, each node is one user. When creating a node in the graph, it means the information has been spread to that user. The node has properties as Table 1.

Property	Description
id	user unique id in Sina Weibo
name	user screen name shown in Sina Weibo.
time	optional. Time to tweet.
betweenness	optional. Betweenness centrality.
pagerank	optional. Pagerank centrality.
closeness	optional. Closeness centrality.
outdegree	optional. Outdegree centrality.

Table 1 Node properties

There are several features for the node modeling:

(1) The time property is optional. When the user express their own idea about the targeted topic, the time will be recorded under the user node. If the user just retweet other users, the time will be recorded under the relationship.

(2) The centrality property is optional. These properties are stored under the node to show in the web interface.

2.3.2 Relationship

In the retweet network, each relationship is one retweet. The relationship has properties as Table 2.

Property	Description
Type:Repost	every relationship is a repost.
Time	Time to retweet

Table 2 relationship properties

2.3.3 Graph build rules

When building the graph, there are several rules:

(1) Each node is unique. One user only maps to one node.

(2) When user retweet himself, it means nothing for information propagation. So this self link is ignored in the graph.

(3) User A may retweet multiple times from User B. In the graph, we only have one edge from B to A which records the earliest retweet time.

(4) If the original tweet is deleted, then ignore this link.

2.4 Data storage

2.4.1 Graph database

Relational databases such as Mysql and Oracle have been used widely since born. They are used to store the structured data in the tables with columns of some predetermined type. When referring to data in other rows and tables, foreign keys are used. And the database will do join work at the query time to match primary keys and foreign keys and to combine those data from different table. These operations are CPU-intensive and cost a lot of time.

For the data like graph-structured or other specific structure data, relational database is not so good because of the efficiency. NoSQL is developed for these special uses. NoSQL means Not Only SQL. It is a group of new databases include mongoDB, CouchDB, HBase and etc. And the graph database used in this project is one kind of NoSQL databases.

When looking at the graph database, there are about 16 databases to choose[7]. For example, FlockDB is the graph database developed by Twitter to store their own graph data. It cause horizontally and is designed for online, low-latency, high throughput environments such as web-site. But it is not capable of doing graph traversal and analyze. So it is not for this project. After comparing and searching on web, we decided to use Neo4j graph database as the system database.

2.4.2 Neo4j database

Neo4j database is a graph database built by Java. It is open-sourced and is free to use. It has following features[8]:

(1) true ACID transactions and high availability.
(2) process billions of nodes and relationships.
(3) high speed querying about traversal.
(4) use Property Graph Model. This model allows both nodes and relationships to have property.

2.4.3 Database operation

The database can be operated directly from the terminal. In directory /neo4j-community-2.1.6/bin, using "./neo4j-shell -path pathToLocalDatabase" to connect to the database. This method can be used to learn the content of the database and debug.

The database also has a web interface to use and see the graph directly. In directory /neo4j-community-2.1.6/bin, using "./neo4j start" to start the neo4j database server. Then open "http://localhost:7474/" to query data and see the result directly as shown in figure 5.

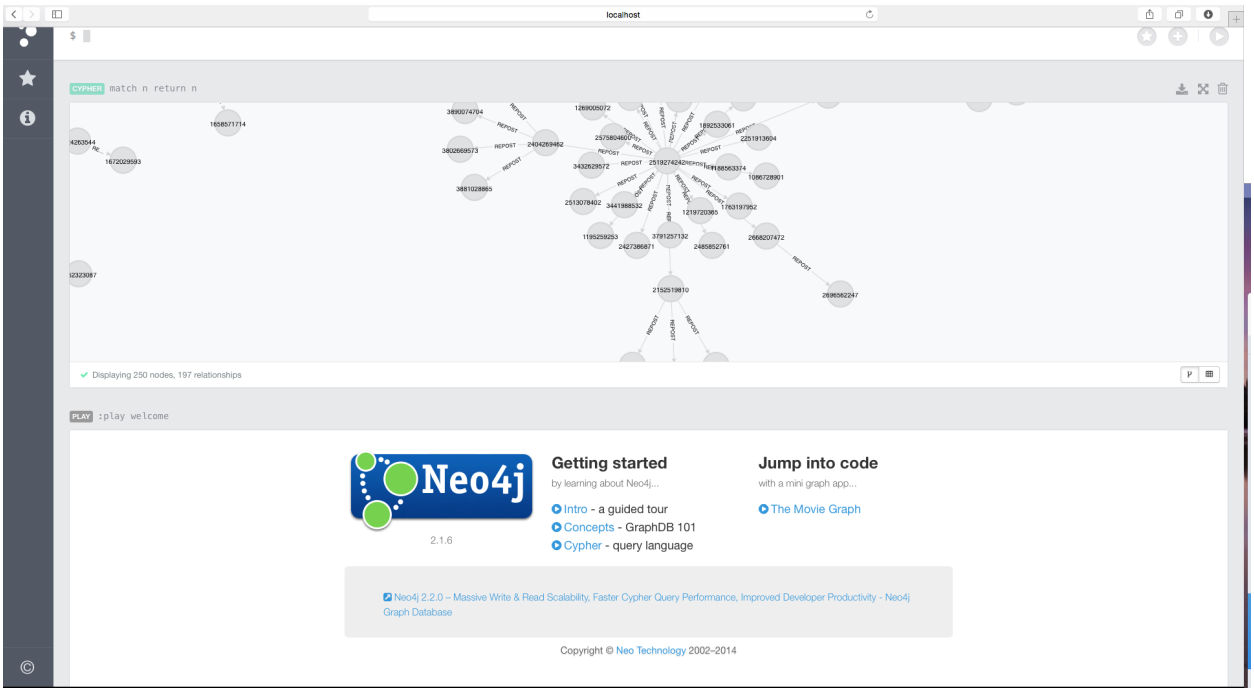

Figure 5 database web interface

2.4.4 Database helper functions

The database is operated in Java in the project. The original method to access database is complexed. Every transaction needs to be surrounded by exception processing code. Some same logical operations such as creating nodes are needed multiple times. So it is necessary to encapsulate the codes into helper functions to simplify the codes. EmbeddedNeo4j.java has all the helper functions inside. The helper functions are shown in Table 3.

function name and parameters	description
Node createNode(String id,String screen_name)	Create a node with user id and user screen name. Return the node created.
Node createNode(String id,String screen_name,String time)	Create a node with user id, user screen name and weibo create time. Return the node created.
void nodeAddTime(Node node,String time)	Add time property to a node.
Node findById(String id)	Find and return the node with the id. If node is not existed, return null.
Node findByScreenName(String screenName)	Find and return the node with the screen name. If node is not existed, return null.
Relationship findRelByNodes(Node start, Node end)	Find and return the relationship between two nodes. If the relationship is not existed, return null.

function name and parameters	description
void addRelWithTime(Node start, Node end, String time)	Add or update time property for a relationship.
String getRelTime(Relationship relationship)	Get the time property of the relationship
String getNodeName(Node n)	Get the screen name of the node
List<Node> getAllNodes()	Get all the nodes that have in-degree or out-degree. The return value is in form of list.
Set<relationship> getAllRels()	Get all the relationships in form of set.
Set<Node> getAllRootNodeWithRel()	Get all the nodes that generate the information and spread it out.
boolean hasNodeProperty(Node node,String p)	If the node has the property p, return true. Otherwise, return false.

Table 3 Neo4j database helper functions

2.5 Web-based interface

2.5.1 Overview

This part is the interface for data visualization and interactive. It includes three parts:

(1) Data querying. Neo4j database provides Restful API to access the data. The API end point is “http://localhost:7474/db/data/transaction/commit”. In the html, Ajax is used to query data. And the return result is in JSON format.

(2) Data pre-processing. The JSON response needs to change the format to be accepted by D3.js.

(3) Data visualization. This part is implemented by D3.js.

2.5.2 D3.js and forced directed graph

D3.js is a Javascript library for data visualization. It has a lot of powerful tools. One of them is forced directed graph. The nodes inside the graph have repulsive charge force to separate the nodes. In addition to that, a pseudo-gravity force keeps nodes centered in the visible area and avoids expulsion of disconnected subgraphs, while links are fixed-distance geometric constraints. The final interface is shown in figure 6.

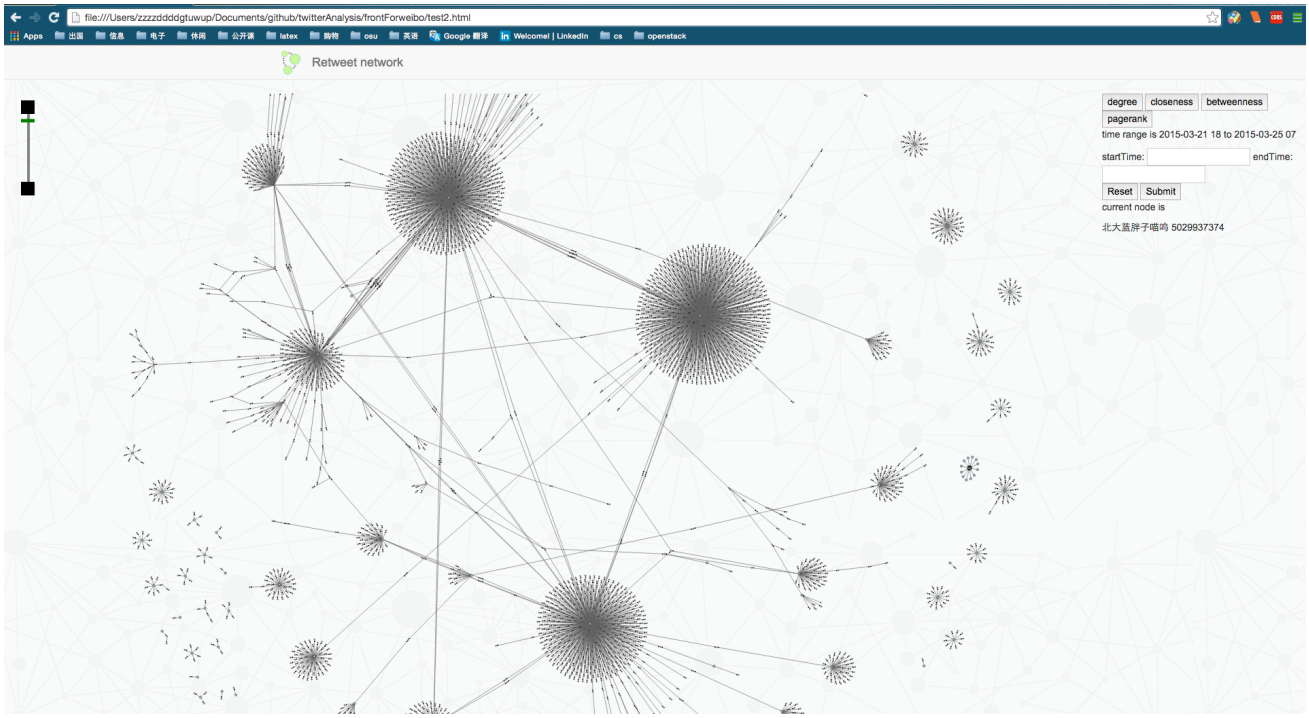

Figure 6 web-based interface

2.5.3 Functions

All following functions are implemented by JavaScript.

(1) The graph view supports zoom in, zoom out and drag.

(2) When mouse moves over one node, the user name, user id and the time will be shown above the nodes and under the 'Submit' button as shown in figure 6.

(3) When clicking one node, the data flow from that nodes will be shown as figure 7.

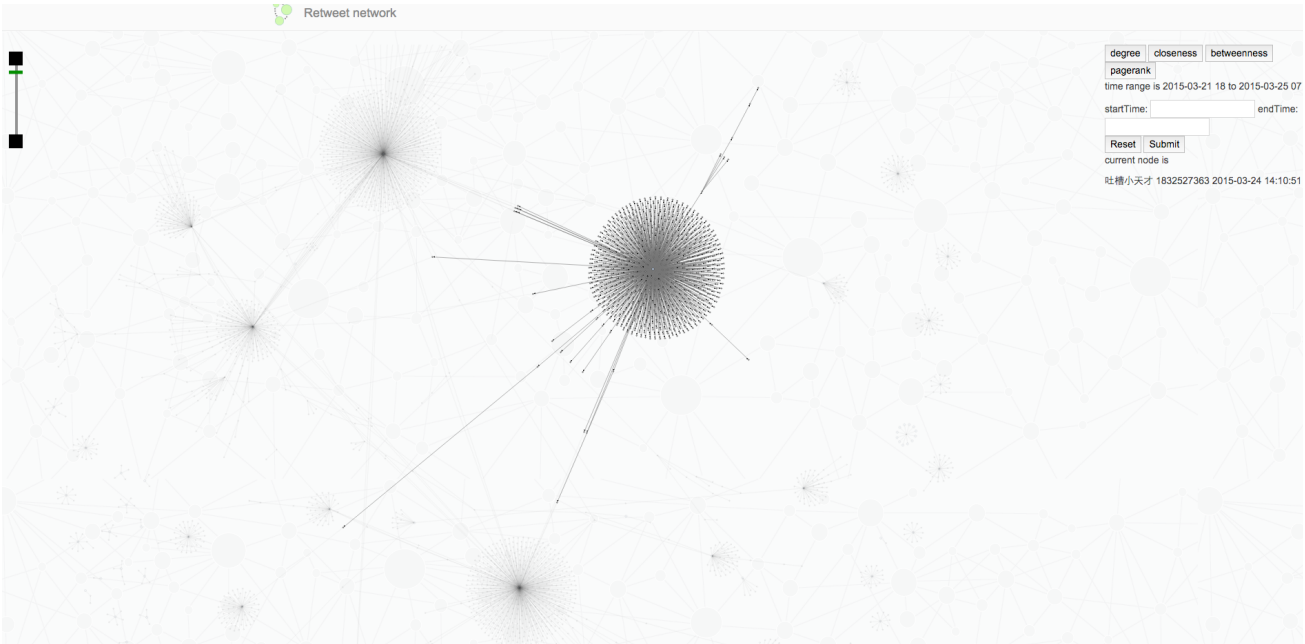

Figure 7 data flow of one node

(4) The time range of the graph is calculated when pre-processing the data and shown on the right side of the web.
(5) The graph support query under some time period. As shown in figure 8, when querying graph between 2015-03-24 15:00:00 to 2015-03-24 16:00:00. The tweet and retweet before the time interval will be in color pink. Those in the time interval will be in color green. Those not happening will fade out.

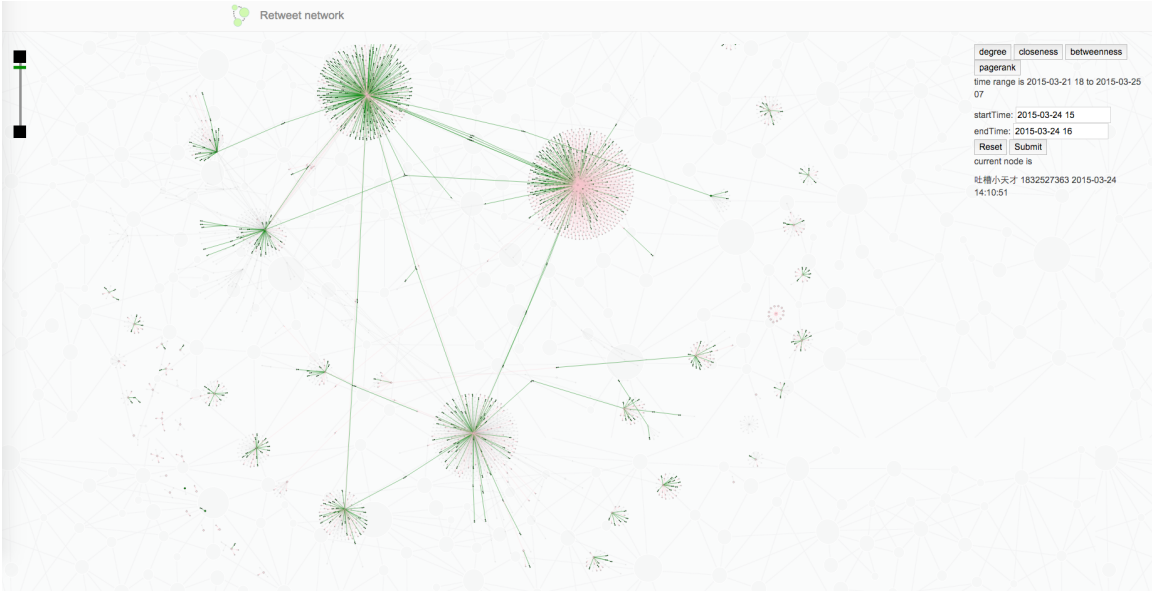

Figure 8 Querying data under time period

3 Analysis

In this section, analysis on graph will be discussed. We will study different types of graph and different network centrality.

3.1 Graph structure

By observing many different topic, three modes of graph are found:
(1) Single polar mode. This is often related to topic like news. Some media account send news and other users retweet that news. One example is topic “Barcelona 2:1 Real Madrid C.F.”. The graph is shown in figure 9. There is single polar in the graph, which is the source of the news. Most of the tweets get information from that single polar.

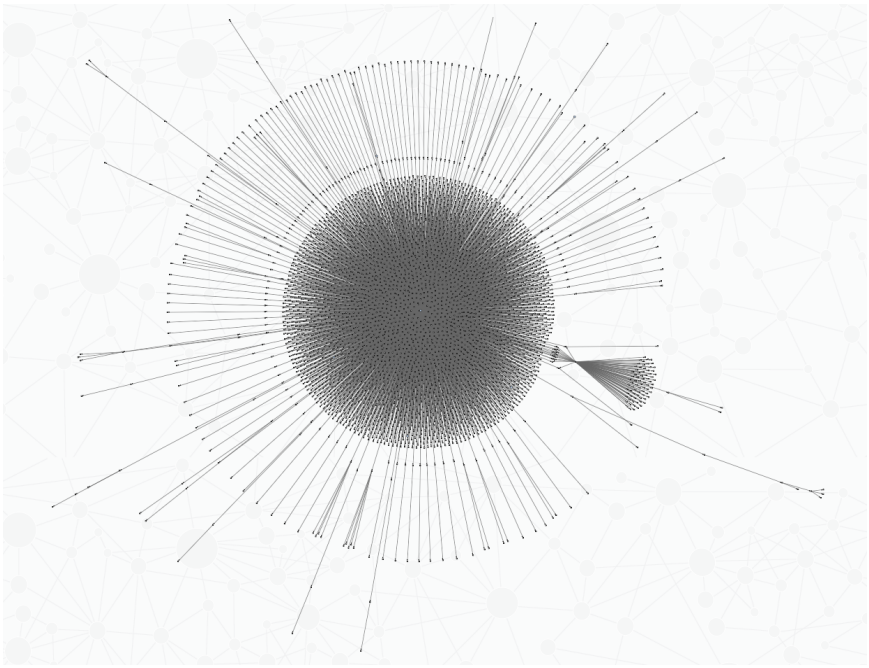

Figure 9 Single polar mode

(2) multiple polar mode. This topic is widely found when observing the data. One example is “Tang Yan and Wu Yifan falls in love” as shown in figure 10. Many users expressed their own ideas and got retweeted by other users. We can also see that some users retweet from different sources.

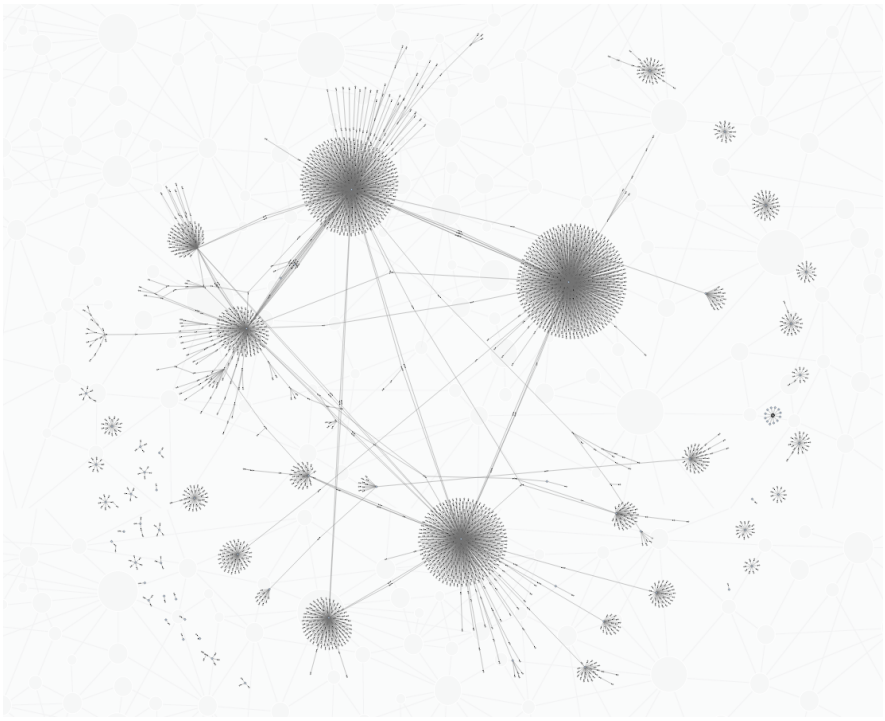

Figure 10 multiple polar mode

(3) Isolated subgraph mode. In this case, there are a lot of small subgraph under this topic as shown in figure 11. This example is topic “2015 ChongQing International Marathon”. This mode often happens when topic is not hot. Maybe only hundreds of users attend the topic. And there is no central and important user in the graph.

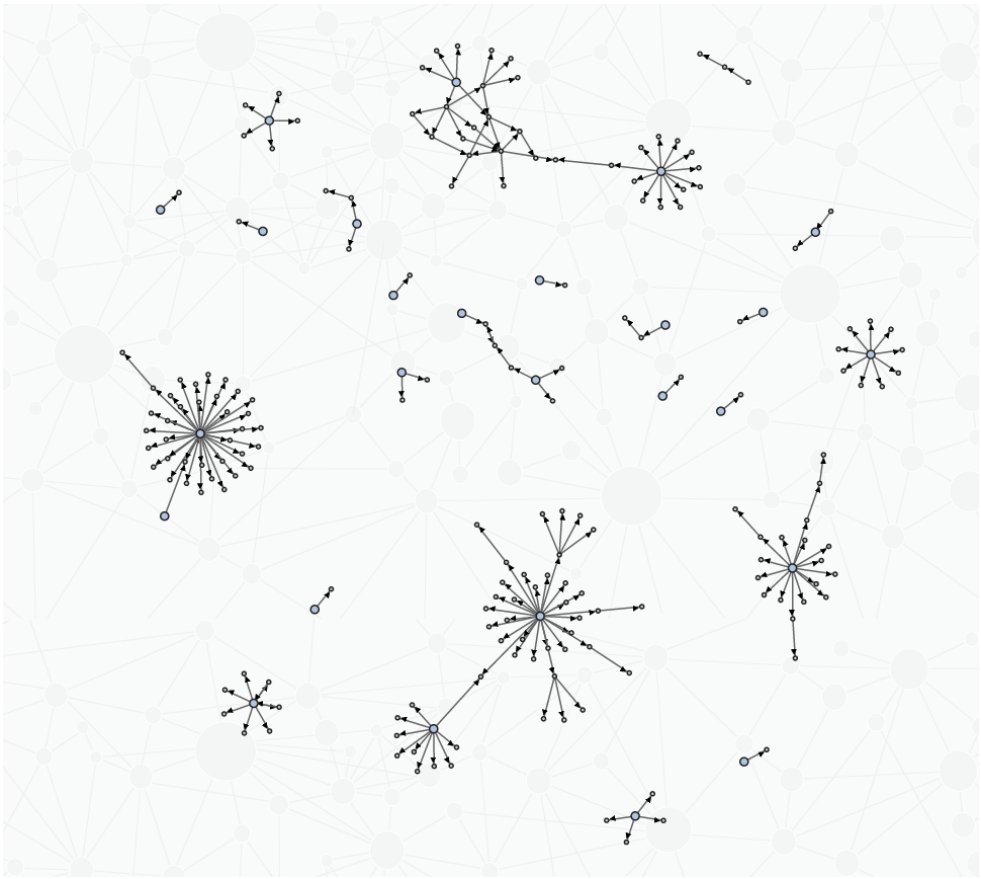

Figure 11 Isolated subgraph mode

3.2 Centrality

Centrality is important when analyzing the structure of the social network and data flow. In this section, we will discuss degree centrality, betweenness centrality, closeness centrality and page rank.

3.2.1 Degree centrality

Degree centrality is the most direct way to measure the importance of the nodes. Out degree is used in the measuring. In the system, there is a degree centrality button. After pressing it, the degree centrality will be shown as figure 12.

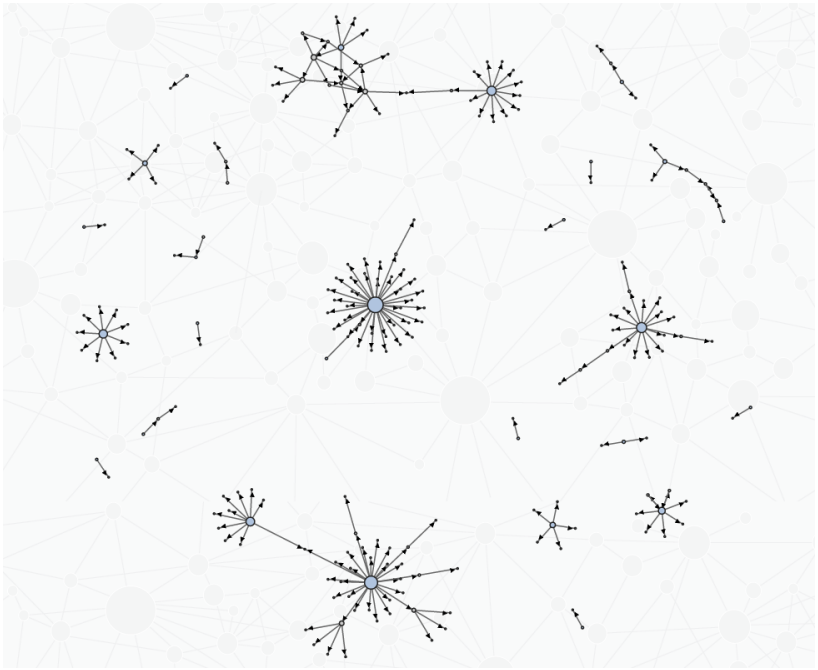

Figure 12 degree centrality

3.2.2 Betweenness centrality

Betweenness centrality counts the times of shortest path which pass the node. This method mainly focuses on the bridge function of the nodes. The betweenness can be presented as[4]:

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where σ_{st} is total number of shortest path from node s to node t through the node v.

From figure 13, we can find that the nodes with high degree don't guarantee high betweenness centrality. Because when count the path, the source and the target are not counted. So this measurement only finds out important bridge nodes.

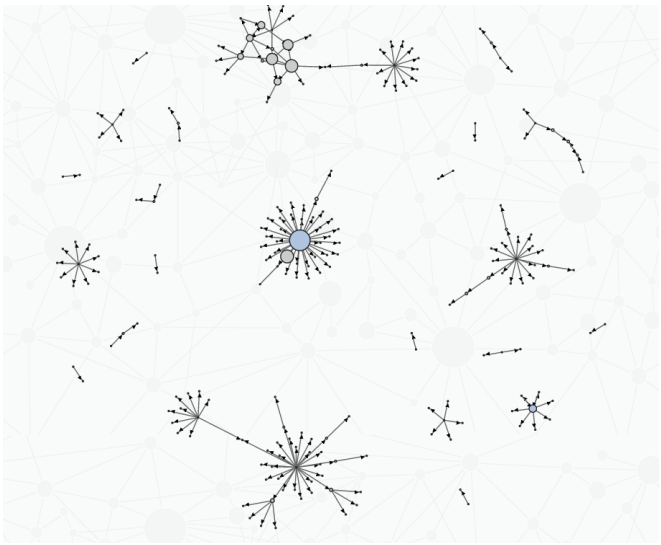

Figure 13 Betweenness centrality

3.2.3 Closeness Centrality

Closeness centrality is the reciprocal of the farness[9] while farness is the sum of the shortest path length to all other nodes. This method is mainly for connected graph. So it is used in the connected components instead of global graph. As shown in figure 14, the closeness is high for nodes near the out side of the connected components. Some source nodes is not high as the outer nodes.

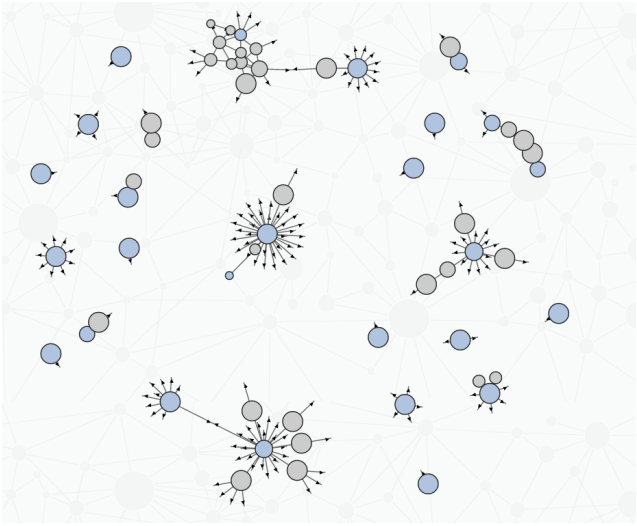

Figure 14 Closeness centrality

3.2.4 Page rank

Page rank algorithm[10][11] is a link analysis algorithm. It is developed by Larry Page, who is one of the founder of Google. It is mostly used in the search engine to determine the importance and the rank of the website. It can also be used to analyze other kinds of network such as social network.

In this project, page rank iterative algorithm is used. Java is slow in calculating matrix. So we output the graph adjacent matrix to a file and use matlab to calculate the page rank value. From figure 15, we can find that page rank has a good balance for source nodes and bridge nodes.

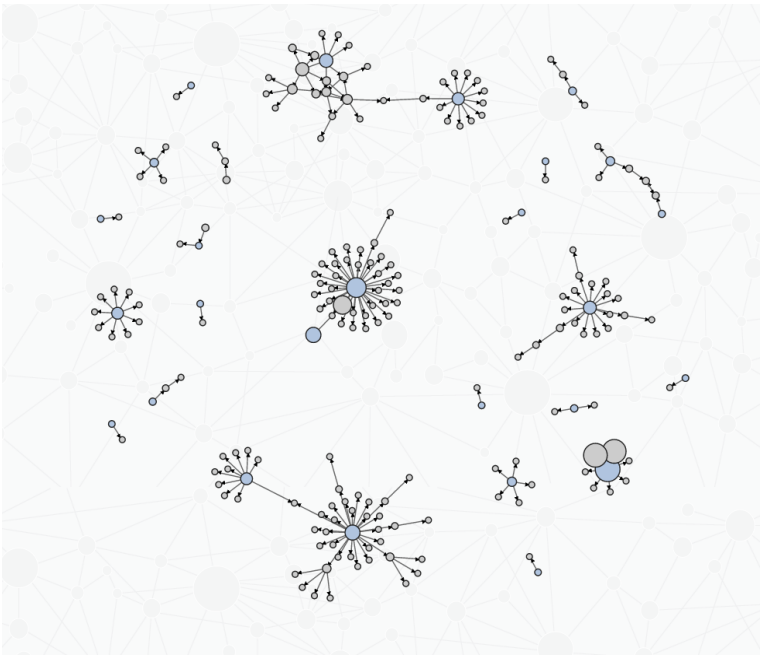

Figure 15 Page rank centrality

4 conclusion

In this project, we presented a new Sina Weibo analytic system for collecting, analyzing and visualizing data. This system allow users to view the retweet network conveniently and directly. It provides many useful functions to catch the data flow and graph in specific time window. Different centrality is also included in the system to offer different analysis method for node importance.

Based on the system, we find three retweet network modes. We also compare different centrality methods. Degree method focuses on the source node. Betweenness Centrality

has more weight on the bridge nodes. Closeness centrality gives more weight to outside nodes. And page rank method is pretty balance.
In the future, we plan to add more functions to the system. There are a lot of zombie users in the Sina Weibo. They should be recognized and deleted from the network.
Furthermore, the time aspect should also be added in the centrality. Those nodes who spread data fast should also be important. At last, the content of the tweet can also be used to study the retweet network. we can separate the network according to the different ideas about same topic.

Reference

[1] Kumar, Shamanth, Morstatter, Fred, and Huan Liu. Twitter Data Analytics. Springer, 2013.
[2] http://baike.baidu.com/view/2762127.htm
[3] S. Kumar, G. Barbier, M. A. Abbasi, and H. Liu. TweetTracker: An analysis tool for humanitarian and disaster relief. In Proceedings of the 5th International AAAI Conference on Weblogs and Social Media, 2011. note: (ICWSM-11).
[4] Ratkiewicz, Jacob, et al. "Truthy: mapping the spread of astroturf in microblog streams." *Proceedings of the 20th international conference companion on World wide web*. ACM, 2011.
[5] Ratkiewicz, Jacob, et al. "Detecting and tracking the spread of astroturf memes in microblog streams." *arXiv preprint arXiv:1011.3768* (2010).
[6] Donghao Ren, Xin Zhang, Zhenhuang Wang, Jing Li, and Xiaoru Yuan. WeiboEvents: A Crowd Sourcing Weibo Visual Analytic System. IEEE Pacific Visualization Symposium (PacificVis 2014), Visualization Notes, Yokohama, Japan, 2014.
[7] https://docs.google.com/spreadsheet/ccc?key=0AlHPKx74VyC5dERyMHlLQ2lMY3d FQS1JRExYQUNhdVE#gid=0
[8] Brandes, Ulrik (2001). "A faster algorithm for betweenness centrality" (PDF). Journal of Mathematical Sociology 25: 163–177. doi:10.1080/0022250x.2001.9990249. Retrieved October 11, 2011.
[9] Sabidussi, G. (1966) The centrality index of a graph. Psychometrika 31, 581–603.
[10]Arasu, A. and Novak, J. and Tomkins, A. and Tomlin, J. (2002). "PageRank computation and the structure of the web: Experiments and algorithms". Proceedings of the Eleventh International World Wide Web Conference, Poster Track. Brisbane, Australia. pp. 107–117.
[11] Massimo Franceschet (2010). "PageRank: Standing on the shoulders of giants". arXiv: 1002.2858