Weibo Social Network Analysis

Wang Siyuan Xia Mengzhou Ma Yiqing
June 18, 2017

CONTENTS

1 DATA SCRAPING

1.1 DATA STRUCTURE

1. **Strategies.**
   Choose users relevant to big data.We thought of two scraping strategies for users.
   - i. Define 10 keywords
     “Big Data”，“Artificial Intelligence”，“Machine Learning”，“Statistics”，“Cloud Computing”，“”⋯. Do search queries in Weibo search page and scrap the users displayed in first 10 pages. (Around 1000 users)
   - ii.Scrape users who post relevant topic in the search page.
     For strategy I,Users are always relevant to the subject we choose based on Weibo’s official display sequence. And they are guaranteed to be influential in the network for the same reason. But this strategy sets the obstacle that we actually can’t find some really influential users relevant to big data like “李开复”automatically since he will never show up in the search list of the previous ten keywords.
     For strategy ii, we must set strict filters like the threshold for the number of followers to eliminate individual users who occasionally post one slice of relevant information to guarantee influence. Even if the users are of great influence, they tend to be from the category of public media like newspapers which are not actually quite relevant to big data.
     After trying both methods, we decided to adopt strategy I and to partly ease the problem mentioned above, we add some influential users like “小冰”by hand.

2. **Data fields**
   Data scraped consists of 4 forms.
   - i.User Information Form
     Items: _id, NickName, Gender, Province, City, Signature, Birthday, Num_Tweets, Num_Follows, Num_Fans, Sex_Orientation, Marriage, URL.
   - ii.Weibo Information Form
     Items: _id(user), ID, Content, PubTime, Co_oridinates(location), Tools, Like, Comment, Transfer.
   - iii.Follows Item
     Items: _id(user), follows, fansOfFollows.
   - iv.FansItem
     Items: _id(user), fans, fansOfFans.

3. **Graph structure**
   - i.Directed graph
     In this Project, we mainly construct a directed graph taking 1000 users as nodes and their following and followed relationships as directed edges. Later the influence problem and community clustering problem both need to be solved based on this structure.

- ii.Choose graph size
  Of course we know that the graph structure is better as bigger as possible for it will have more information to reflect the true Weibo social network. However, we only need to consider the 1000 user’ s influence ranking and for a better consideration, we finally choose 1000 users related to ‘Big data’ topic with relatively more fans so that it construct a more close relevance between these 1000 users. As a consequence, it less possible to expand the network structure use their fans and follows because many users have a large amount of fans especially one with more than 5000000 fans. So we finally only use these 1000 users as nodes.
- iii. Deal with sparse graph
  It’ s natural that our graph structure is quite sparse. It’ s a limit of our crawling data and project. To overcome this drawback to a certain extent and make full use of user information, we will take users’ fans and follows as weights to nodes and consider different influence comment algorithm from different angles. It will be expound later.

1.2 SCRAPING STURCTURE

The whole scraping structure is based on Python scrapy framework as displayed in figure 1.1 .

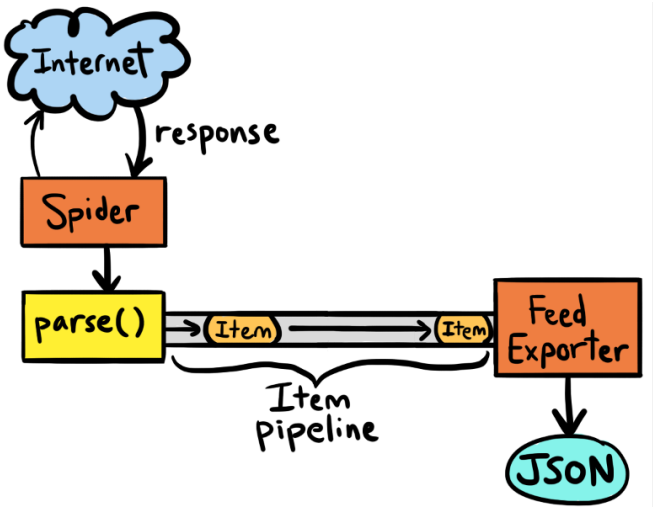


Figure 1.1: framework

- Selenium webdriver for unlocking verification code, as shown in figure 1.2.
  Implemented a simulation tester to unlock slider verification code.

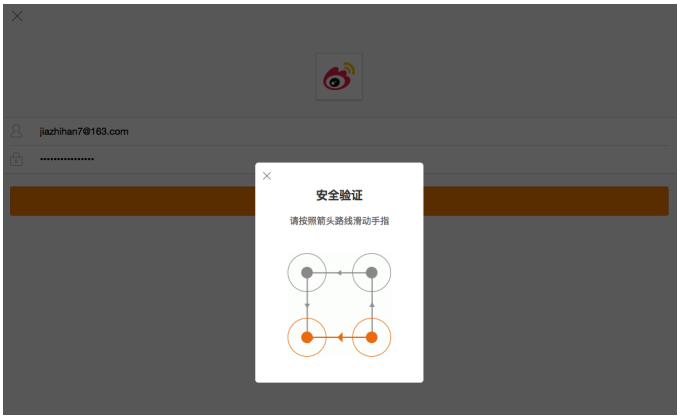


Figure 1.2: Selenium webdriver

- Middleware configuration for randomly choosing cookies and user-agents
- Scrapy selector xpath for analyzing html
- Mongodb for storing data configured in pipelines.

2 Statistical Analysis

We want to visualize the relationship and do statistical analysis based on the data we scraped from Weibo. After referring to the Twitter Case analysis diagrams in lecture handouts 11, we decided to use Echarts to visualize all of our results and display the following figures.

2.1 4 analysis

1. **The relationship between follows and fans**
   As shown in figure2.1, we analyzed the relationship between the number of fans and the number of the subscribers of the 1000 users from the weibo. We divided the users into two groups. The first group contains those who have more than 3000 fans and the second group contains those who have less than 3000 fans. We compare the average number of subscribers of these two group. They are 593 and 661.Based on randomness there is no much difference.

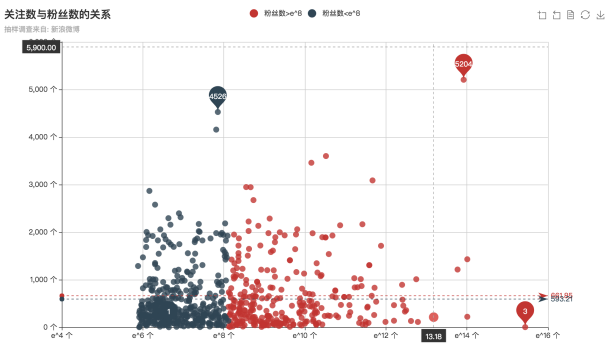


Figure 2.1: relationship between the number of fans and the number of the subscribers

2. **The relationship between fans and posts**
   As shown in figure2.2,we analysize the relationship between the number of followers and the number of Micro-blogs. We also divide theusers into two groups and compare the average number of Micro-blogs.They are 2606 and 882. So we discover that those who have more than 3000 tweets more than those who not.

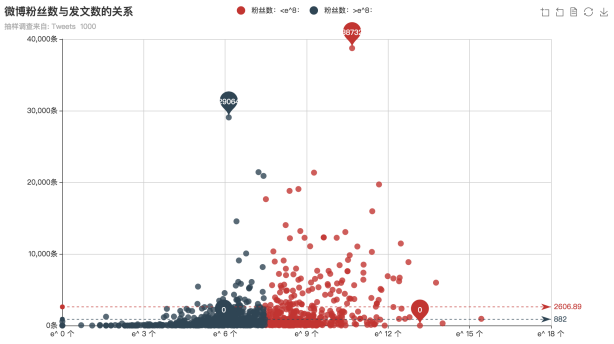


Figure 2.2: relationship between the number of followers and the number of Micro-blogs

3. **The relationship between fans and posts of users having fewer than 1000 fans**

   As shown in figure2.3, Based on the Figure[2],we choose those who have less than 1000 fans. We expected to get a linear relationship between the number of fans and the number of Micro-blogs. However the total number of users we scraping from the Weibo is rather small. We can not clearly see the linear relationship between these two variables.

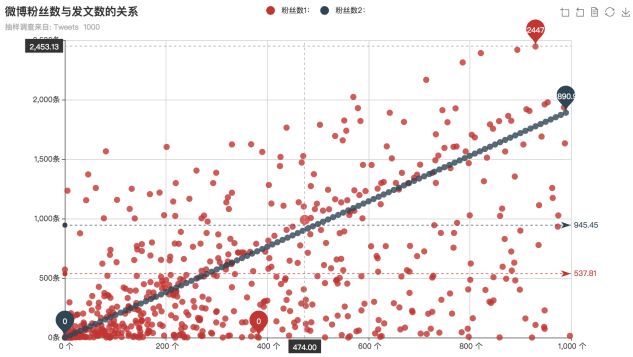


Figure 2.3: relationship between fans and posts of users having fewer than 1000 fans

4. **At what time do users post weibo?**
   As shown in figure2.4, We intend to analysize the micro-blog’ s users most active time.The result is that the user sends big data related micro-blog's mainly concentrates on 10:00 o’ clock in the morning

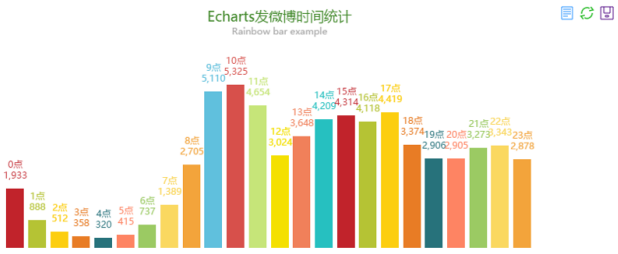


Figure 2.4: Micro blog users most active time

3 User Influential Analysis

It’ s obvious that user influence is a quite complex analysis related to many factors, including user active degree, covering rate, transmissibility and so on. So we can’ t use only one quota to analyze this problem. We should approach the problem all-round. Here we use these different methods.

3.1 PageRank

- **Introduction**
  PageRank (PR) is an algorithm used by Google Search to rank websites in their search engine results. It works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is

that more important websites are likely to receive more links from other websites. Here we take users instead of websites and following and followed relations as links and we can estimate the influence of users. As the same, we assume that more influential users have much more fans.

- **Why use PageRank**
  PageRank is also one method of centrality measurement. Compared to eigenvector centrality and Katz centrality, it is a high level implementation considering much more true situation. For example, when a node without any outer edge, it can't convey centrality. But pagerank will assume that the node has a fixed probability jumping to any other nodes. To implement this, we need to add edges for all nodes with 0 out degree. And then if a node with a high centrality has many outer neighbors, its high centrality will also convey to their neighbors, which is not reasonable. So in pagerank the centrality conveyed should be divided by the out degree of the neighbors.

$$PR(p_i) = \alpha \Sigma_{p_j \epsilon M_{p_i}} \frac{PR(p_j)}{L(p_j)} + \frac{(1-\alpha)}{N} \tag{3.1}$$

  L(pj) is the num of node pj's out degree, and alpha usually is 0.85. In the end, it will converge to the final pagerank value.

- **Improvement**
  In this project, we do not just use origin pagerank algorithm but also make some improvement. Because we only use a small size 1000 nodes graph, so the fans and follows information are significant too. We take these rate and give them different weights as the initial value and transition probability, that is random term is not 1/N for each node any more but the sum of weighted fans rate and follows rate. And 1/L(pj) also need to be changed to the rate during its all neighbors. We must recognize that fans are more important with a higher weight.

- **Result**
  In this well-known algorithm, we finally converge and compute the pagerank value and sorted in descending order, and first k is most k influential users. The result is shown in figure3.1, We can see these top 20 influential users are reasonable including [小冰，数据挖掘与数据分析，社会网络与数据挖掘] and so on. However, even the pagerank is a quite perfect method, but for that our network is not complete and too sparse, the ranking result may has a discounted accuracy. And unfortunately, there's no relationship between these nodes.

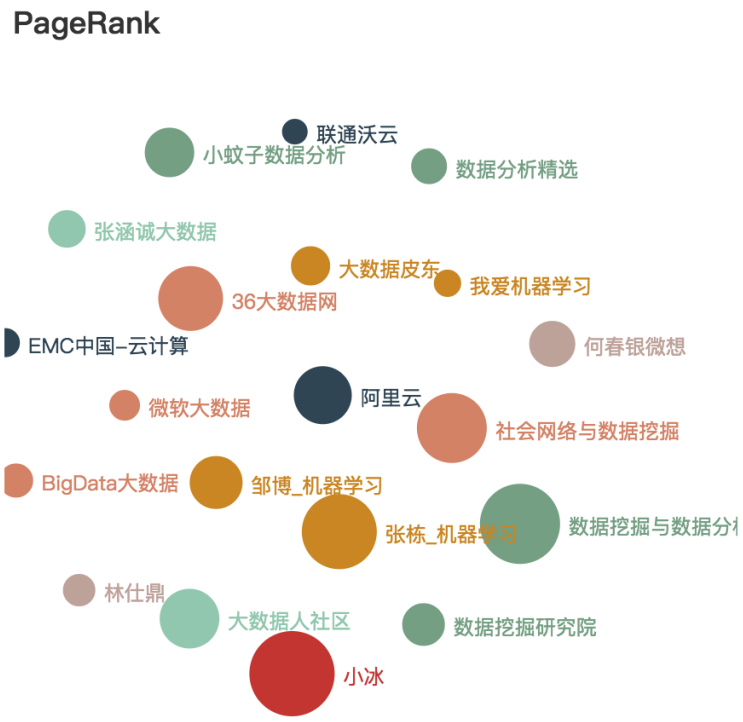


Figure 3.1: pagerank

3.2 Degree Rank

- **Why still degree rank**
  Because our graph structure is just part of true network, so we choose another simple but intuitive algorithm degree centrality which make full use of users' fans and follows information.

- **Implementation**
  The degree centrality is

$$C(v) = d_{in} + d_{out} \tag{3.2}$$

  din represent prestige computed by fans number while dout represent sociability computed by follows number. However, to appraisal users' influence, we know prestige is more vital. So we do not use the origin degree centrality formula but give din and dout different weights.

$$C(v) = \partial d_{in} + \beta d_{out} \tag{3.3}$$

  In project we set $\partial = 0.85$ while$\beta = 0.15$

- **Result**
  In this algorithm, we sorted users' degree centrality C(V) in a descending order and

also first k are most k influential users. Figure3.2 is the our top 20 users. And here are a closer relation between them.

rank.png

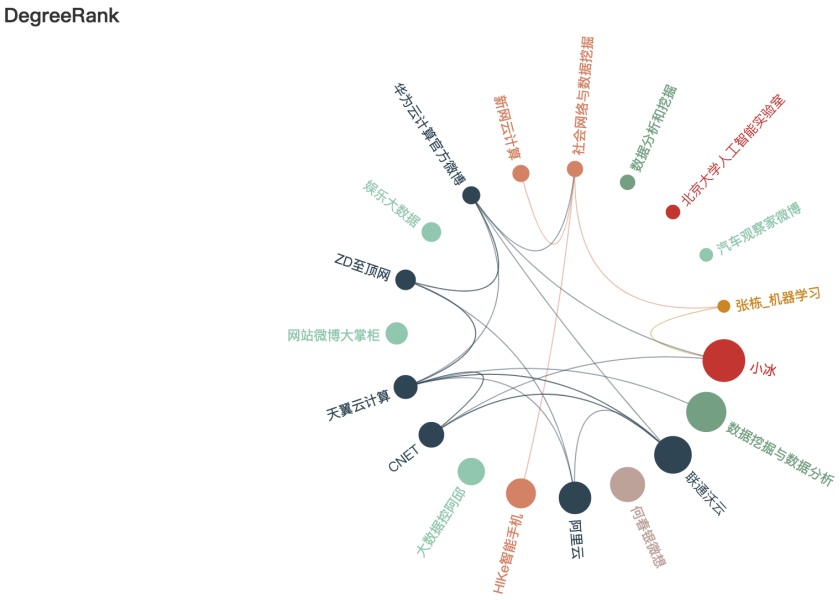


Figure 3.2: degree rank

3.3 Weibo Behavior Rank

- **Why need Weibo behavior**
  For we know, Weibo users' influence is still related to users' Weibo transmissibility. If a user pub many influential Weibos which have been commented, forwarded and liked, he will be more influential.

- **Implementation**
  So we still use weighted algorithm to achieve this method. The formula is

$$D(v) = \partial x + \beta y + \gamma z \tag{3.4}$$

  Where x is comment number, y is forward number while z is the like number. All of these numbers we compute by averaging user' s all tweets. We know that comment, forward and like has a descending influence order. So we set$\partial = 0.5, \beta = 0.35, \gamma = 0.15$

- **Result**
  In this part, we have a big different result as shown in figure3.3 which have liitle consistency with the former two for these are two definitely different evaluation ways.

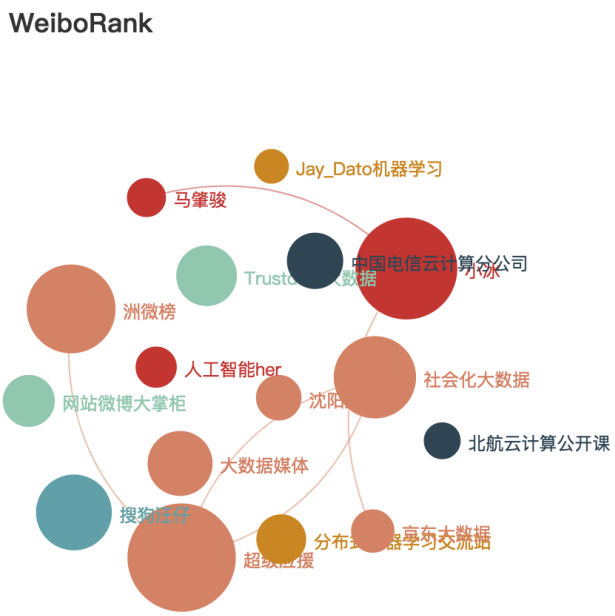


Figure 3.3: weibo rank

3.4 Combination

To conclude all these three algorithm, we can generate an intersection on these three sets. Figure3.4 is the result of a Venn Graph. And we can see pagerank and degree have more same influence users while WeiboRank not. And the intersection users must be very influential users.

Figure 3.4: Venn Graph

4 KEYWORD EXTRACTION

Texts are our target users' most recent 100 posts, around 1 million pieces in total. The whole text file is 14.6 MB. We implemented TF-IDF and TextRank to do keyword extraction.

4.1 TF-IDF

- **introduction**
  TF-IDF algorithm is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. Tf refers to (term frequency) and can be calculated by the following formula:

$$tf(t,d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{max f_{t',d} : t' \epsilon d} \tag{4.1}$$

  And IDF refers to (inversed document frequency) and can be calculated by the following formula:

$$idf(t,d) = log \frac{N}{|d \epsilon D : t \epsilon d|} \tag{4.2}$$

  Then TF-IDF value is the product of these two statistics. And following is the simplest ranking functions computed by summing the tf – idf for each query term.

$$tfidf(t,d) = tf(t,d) \cdot idf(t,f) \tag{4.3}$$

- **problems and solutions**
  Actually we can only get term frequency since we only have one document to describe and unless we get a much bigger word corpus, otherwise it is meaningless to calculate the IDF since every word's IDF is exactly the same. We've thought of 2 solutions to tackle the problem.

  - Divide the document into 10+ documents thus IDF is effective. And then combine top words of these sub-documents. However, this method has got the problem that these divided documents are homogeneous. Calculating TF-IDF on each document will be very much likely to filter words like "数据" since it is a word frequently appears in each sub-document.
  - Simply calculate term-frequency and eliminate all the stop words. In the real implementation, we chose the second method and get the following results.

- **results and evaluation**
  The top-30 words are as follows.
  数据, 中国, 安全, 计算, 技术, 企业, 手机, 公司, 应用, 学习, 用户, 互联网,发表, 分析, 平台, 发展, 百度, 分析, 数据挖掘, 市场, 产品, 红包, 服务, 用户, 问题, 专业版, 人工智能, 世界, 时代
  The result is actually quite reasonable, but in order to verify the correctness of the result, we implemented another algorithm TextRank to do keyword extraction again.

- **introduction**
  TextRank[2] is a graph-based ranking algorithm to decide the importance of a vertex within a graph, based on global information recursively drawn from the entire graph. The specific ranking details are quite similar with PageRank mentioned above thus will not be fully described in this section.
  In TextRank, each effective word is a node in the graph and the edge is determined by the distance between words. If two words appear in the same window of length k, then these two nodes are connected by an edge (Co-occurrence Links). An example is shown as the following figure 4.1

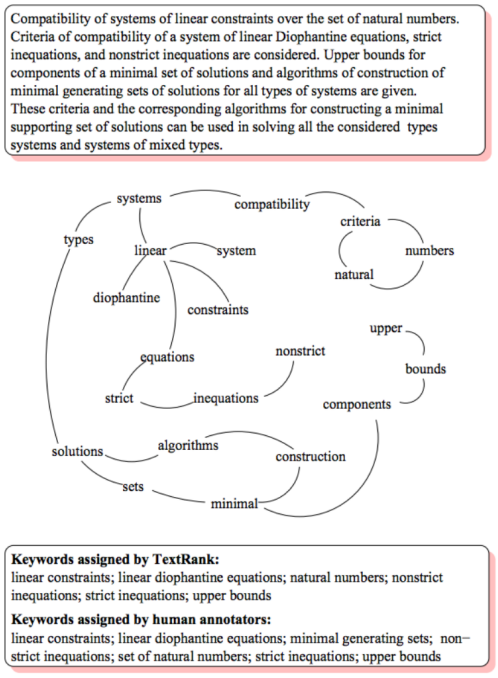


Figure 4.1: example

- **Why we use TextRank**
  TextRank does not only rely on the local context of a text unit (vertex), but rather it takes into account information recursively drawn from the entire text (graph). Through its iterative mechanism, TextRank goes beyond simple graph connectivity, and it is able to score text units based also on the “importance” of other text units they link to.
  The underlying hypothesis is that in a cohesive text fragment, related text units tend to form a “Web” of connections that approximates the model humans build about a given context in the process of discourse understanding.

- **Process**

- Jieba[3] for Chinese text segmentation
- Stop words list to eliminate irrelevant words.
- Self-implemented TextRank algorithm to iteratively calculate the importance of each node.
  Note: After submitting our code, we discovered that Jieba also offered TextRank keyword extraction. Thus we did experiments again both on the full-text and each community's text files to get more accurate results. The results below may be a bit different from the those we showed in presentation.

- **Results and Evaluation**
  For the full-text, the result is as follows. Words of top 50 are displayed in significance descending order.
  '数据', '中国', '计算', '学习', '企业', '技术', '互联网', '分享', '发布', '公司', '分析', '大家', '文章', ' 没有', '发展', '应用', '市场', '人工智能', '用户', '智能', '时代', '平台', '行业', '北京', '需要', '开始', ' 产品', '手机', '移动', '全球', '时间', '机器', '服务', '世界', '推荐', '头条', '系统', '机器人', '视频', '问题', '使用', '美国', '研究', '网络', '成为', '进行', ' 时候', '信息', '工作', '科技'
  The result is quite reasonable, including words like "数据", "计算", "互联网" and so on, which are in accordance with what our presumption. But still it contains words like "进行", "时候",which do not offer too much information. Therefore we updated the stop words and eliminated verbs in jieba and did the experiment again to get the final ranking list of top 50.
  '数据', '企业', '中国', '服务', '技术', '发展', '平台', '互联网', '公司', '市场', '用户', '全球', '行业', '产品', '业务', '管理', '网络', '智能', '软件', '系统', '移动', '科技', '时代', '数据中心', '领域', '客户', '信息', '产业', '美国', '工作', '北京', '问题', ' 城市', '大会', '建设', '解决方案', '手机', '中心', '传统', '世界', '视频', '国家', '分析', '峰会', '设计', ' 能力', '设备', '研究', '模式', '服务器'
  TextRank' keyword extraction seems to be more accurate than TF-IDF but indeed does not have too much difference from term-frequency. These two algorithms both generate effective keywords after filtering stop-words.

4.3 Word-Cloud Visualization

5 Community Detection Algorithm – Fast Unfolding (Louvain)

5.1 introduction

Fast Unfolding[4] is a modularity-based community detection algorithm, which incorporates the following 5 steps.

- i.Firstly regard each node in the community as an independent community, the number of communities is the same as the number of nodes.

Figure 4.2: Word-Cloud

- ii.For each node i, assign each node i to the community its neighbor node belongs to, calculate the change of modularity before and after the assignment and keep record of the neighbor node that maximized the modularity change. If the maximum value is larger than 0, then assign node i to the corresponding community, otherwise nothing is done.
- iii.Repeat ii until all nodes' communities don' t change.
- iv.Compress the graph to make all nodes in one community to one new node, the weights of edges inside the community is transformed to the loop weight of the new node and the weights of the edges between communities are now weights of new nodes.
- v. Repeat i until the modularity of the whole graph keeps constant. In the paper the change of modularity is defined as ...

$$\triangle Q = [\frac{\Sigma_{in} + k_{i,in}}{2m} - (\frac{\Sigma_{tot} + k_i}{2m})^2] - [\frac{\Sigma_{in}}{2m} - (\frac{\Sigma_{tot}}{2m})^2 - (\frac{k_i}{2m})^2] \tag{5.1}$$

5.2 Why we use Louvain

- **Least computation time**
  Unlike other community algorithms, Louvain produces hierarchical communities. The bottom level takes the most time, and after compressing communities, the nodes and edges will be cut tremendously. And the modularity change is relevant only with the communities these two nodes belong to. Thus the computational speed is phenomenal. It only took around 10 seconds on our dataset.
- **Effective on sparse graph**
  The community detection algorithm is quite effective in detecting communities of a

sparse graph with large number of nodes and relatively fewer edges. The graph in our case conforms to this specification thus the detection achieves great performance.

5.3 Results and Evaluation

- **Communities**
  We successfully partition all nodes into 8 communities, with each contains more than 100 nodes. The result is shown in figure 5.1. Each color represents one community.

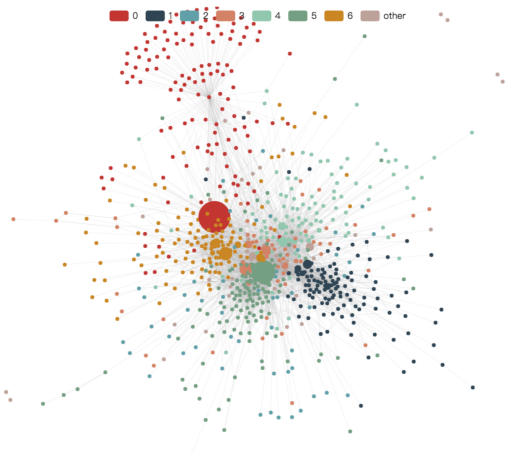


Figure 5.1: Communities

- **Analysis of the results**
  In order to gain a deeper understanding of the community detection results, we further analyzed the texts we’ ve gathered of the corresponding users in each community. Mainly we repeat the steps in Part IV and get the following results.

  1. Group 0 : [’人工智能’, ’机器人’, ’中国’, ’智能’, ’数据’, ’技术’, ’实验室’, ’公司’]
  2. Group 1 : [’计算’, ’云计算’, ‘企业’, ’中国’, ’ 服务’, ’技术’, ’应用’, ’平台’, ’发展’]
  3. Group 2: [‘数据’, ‘中国’, 互联网’, 企业’, 技术’, 数据分析’, 公司’, 发展’]

  We can see from the results that these 3 groups do manifest focus on different topics specifically “人工智能”, “云计算” and “数据分析”, reinforcing the idea that users tend to follow those who they share the topic with, eg. users related with AI are more likely to follow other users related with AI rather than cloud computing. This result can be more clearly illustrated in our front-end web pages

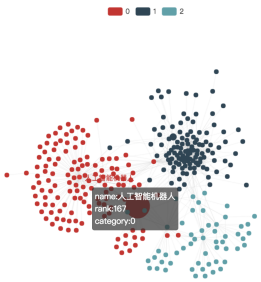


Figure 5.2: "人工智能"topic

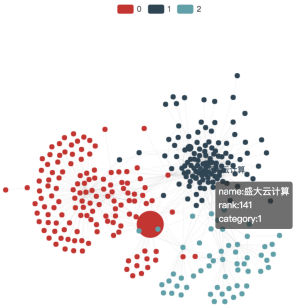


Figure 5.3: "云计算"topic

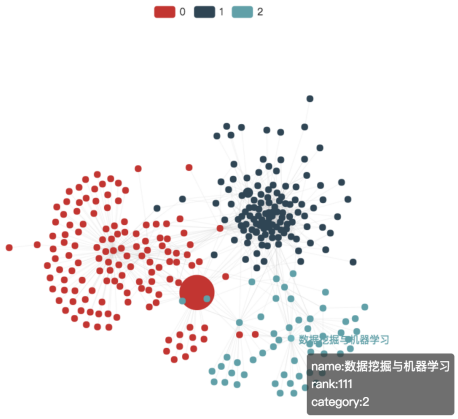
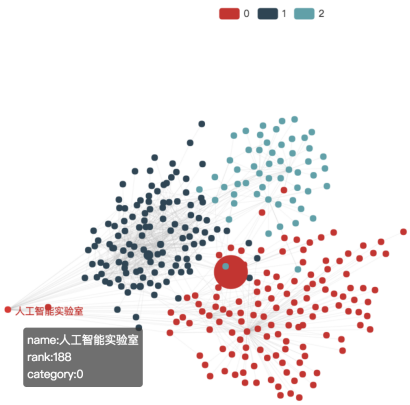


Figure 5.4: "数据分析"topic

Inside a certain group, the relationship between users are also various. Some groups display a complex follow pattern in which the edges are crossed in a mixing way while some groups' follow pattern is much easier, usually a group of users connected by one certain node. Also, quite interestingly, we can find users like "人工智能实验室" acting as a connecting role between two communities, as displayed in figure 5.5.



11.png

Figure 5.5: “人工智能实验室”connecting

6 References

1. https://yuantw.gitbooks.io/dataservice/content/scrapy.html
2. Mihalcea R, Tarau P. TextRank: Bringing order into texts[C]. Association for Computational Linguistics, 2004.
3. https://github.com/fxsjy/jieba
4. Blondel V D, Guillaume J L, Lambiotte R, et al. Fast unfolding of communities in large networks[J]. Journal of statistical mechanics: theory and experiment, 2008, 2008(10): P10008.

7 Thoughts and reflection

Through the whole process of the project, we’ ve got to do real practical work on a real network, which enforces what we learned in class. But there are still much room for us to improve.

- More users can be added to the graph to form a relatively accurate network.
- Text analysis could be more accurate by training models like word2vec to better describe users.
- Post’ s propagation path can be taken into consideration in user influential analysis. (We did not do it because the information we gathered is not of much use, most forwards are insignificant)
- More community detection algorithms can be implemented to guarantee accuracy.