Recommendation System for Twitter Community

Jinfen Li, Heli Shah, Jiaqi Chen, Wenzhe Yang, Hamini Patel

1. Abstract

Twitter is a popular social media having 326 million active users every day. With such a huge amount of users, it is crucial to divide them to different communities for user recommendation.

In this paper, we build up a recommendation system to recommend users or hashtags based on some criteria. We use spectral, louvain clustering algorithm to do the community detection and evaluate the quality of community separation based on latent dirichlet allocation (LDA). Then we recommend twitter users who tweet nearby, who share the same topic, the popular people based on centrality or follower count or friend count. We also recommend hashtags based on the frequency so that users can choose from the automatic list of hashtags. All recommendations happen within the same community so that we can narrow the group and recommend the most likely related users.

2. Introduction

It is easy for two people in real life to become friends with someone if they share the same interest or disinterest, if they are popular among the community, if they tweet nearby the user's location or they belong to the same community. We decided to address the same constraints for our project to recommend friends to a Twitter user. It is interesting to find out that someone in your same community has the same hobbies and interests as you. So, our this project delivers the recommendation to a twitter user if another user is in the same community, tweeting on similar topics, popular in the community based on the centrality, popular based on the followers or friends count or if they tweet nearby the user's coordinates. At last, we also deliver the hashtag recommendations based on their frequency in the same community.

For delivering this recommendations, we start with collecting our data of users from the Twitter for whom recommendations would be made. We start with a user and mine his/her Friends and Followers and store their id, recent tweet, friend's count, follower's count and location in the dataset. We then apply various methods for community detection, popularity, finding location and hashtag recommendations on this collected data.

3. Data Collection

Data collection is defined as the process of collecting and measuring information of some targeted nodes or variables in an implemented system, so that the information can be used to evaluate some patterns, solve certain relevant questions or test a hypothesis. While building any system, data collection is the first and an important process that is performed because a system needs data to perform the evaluation.

Similarly, we also need information for performing data mining on it for our project. Thus, in our project we carry out data collection on twitter community to find some relevant user information on which evaluations can be performed to get the outcome.

As we are performing data collection for twitter community, we use twitter API and user functions to retrieve the user information and its attributes. For data collection, we take a starting user as the first node and crawl his/her friends or followers using crawl function in order to make them level-1 nodes. We do the same procedure on the level-1 nodes to find level-2 nodes and same for the other level nodes.

This way, we collected a bunch of users on whom we can apply various tics and tacs mentioned above through which the users and other data like hashtags would be recommended.

The attributes of users that we collect are as follow:

We used the following code for collecting our data:

```
def get_my_object(twitter_api, screen_names=None, user_ids=None):
    # Must have either screen_name or user_id (logical xor)
    assert (screen_names != None) != (user_ids != None), \
        "Must have screen_names or user_ids, but not both"

    items_to_info = []
    #Specifying the radius area till which boundaries a tweet should be searched for
    max_range = 10
    #Setting the maximum range of tweet results
    num_results = 3

    if screen_names:
        response = make_twitter_request(twitter_api.users.lookup,
                                        screen_name=screen_names)
    else:
        response = make_twitter_request(twitter_api.users.lookup,
                                        user_id=user_ids)
    # print(response)

    #print(response['location']['lat'])
    #tweets = twitter_api.search.tweets(q = "a", count=1000)

    for user_info in response:

        #print(user_info)
        items_to_info.append(str(user_info['id']))

        items_to_info.append(user_info['screen_name'])


        if 'status' in user_info:
            items_to_info.append(user_info['status']['text'])

        items_to_info.append(str(user_info['followers_count']))
        items_to_info.append(str(user_info['friends_count']))
```

```
        if user_info['location'] == '':

            items_to_info.append('')
        elif user_info['location']!=None:
            items_to_info.append(user_info['location'])

            print(user_info['location'])
            location = user_info['location']
            g = geocoder.osm(location)
            #print(g.bbox)
            if g.northeast != None and g.southwest != None:
                neLat=g.northeast[0]
                neLng=g.northeast[1]
                swLat=g.southwest[0]
                swLng=g.southwest[1]

                result_count=0
                num_results=1
                while result_count < num_results:
                    stream.filter(locations=[swLng,swLat,neLng,neLat])
                    print(screen_name1)
                    items_to_info.append(screen_name1)
                    result_count += 1

    return items_to_info
```

To collect user information, we call the *get_my_object()* function with the screen_name and the user_id as arguments. The screen_name and the user_id are of the user that is currently considered as the node i.e. starting node, level-1 nodes, level-2 nodes and so on.

In this function, we make a twitter request using twitter API for *users.lookup* which is a standard GET request for fetching the basic information of user like user's id, name, location, recent tweet, number of friends and followers of the user i.e. friends count and followers count. Then, we store the values of these attributes in an array for each user.

After fetching this data, we store it in a 'csv' file for further use i.e. mining on that data. Following is the code for storing the gathered data:

```
def store_data(screen_name,limit):

    premium_search_args = oauth_login()

    response = make_twitter_request(premium_search_args.users.lookup,
                        screen_name=screen_name)
    user_id = ''
    for r in response:
        user_id = r['id']
    friends_ids, followers_ids = get_friends_followers_ids(premium_search_args, screen_name=screen_name,
                                                           friends_limit=10,
                                                           followers_limit=10)
    connection = []
    connection.extend(friends_ids)
    connection.extend(followers_ids)
    edge,nodes = crawl_followers(premium_search_args, screen_name, connection, user_id,limit=limit)
    all_user_info = []
    with open('data.csv', 'w') as file:
        for e in set(nodes):
            user_info = get_my_object(premium_search_args, user_ids=e)
            all_user_info.append(user_info)

        df = pandas.DataFrame(all_user_info,columns=['id','name','text','follower_count','friend_count','location',
                                                     'recommendation_1'])
        pandas.DataFrame.to_csv(df,file,index=None)
    with open('edge.pickle','wb') as file:
        pickle.dump(edge, file)
```

Using the *store_data()* function with the *screen_name* as an argument, we store the user node's data in the output file. A separate field with the column headings of the attributes is stored for each user that is traversed.The user's 'location' field can be used to get recommendations of friends for that on the basis of tweets' location of other twitter users, the 'friends_count' and 'followers_count' field can be used to find the popular user to recommend him/her to our targeted user, the 'text' field can be used to analyze the topics of tweets based on which a twitter user can be recommended who has interest in the same topic or who tweets about the same topic.

This information in the output file is then used further for performing various data mining algorithms in order to get the output as per given requirements which is recommendations of friends based on popularity, location of tweets, same community, topic of interest and hashtag recommendations in our case.

4. Community Detection

After we build up the network having edges connected users and their followers or friends, we divide them into different communities using Louvain and spectral clustering algorithm. Then we compare the quality of the community based on the comprehension of the topics provided by LDA.

4.1 Louvain

According to the official definition from Neo4j Graph Algorithms library, the Louvain method of community detection is an algorithm for detecting communities in networks. It maximizes a modularity

score for each community, where the modularity quantifies the quality of an assignment of nodes to communities by evaluating how much more densely connected the nodes within a community are, compared to how connected they would be in a random network.

The Louvain algorithm is one of the fastest modularity-based algorithms, and works well with large graphs. It also reveals a hierarchy of communities at different scales, which can be useful for understanding the global functioning of a network.

According to wikipedia, modularity is a scale value between -1 and 1 that measures the density of edges inside communities to edges outside communities. The modularity matrix could be computed as $B = A - dd^T/2m$ where m is the number of edges, A is the adjacent matrix, d is degree matrix. The modularity is defined as

$$Q = \frac{1}{2m} \sum_{l=1}^{k} \sum_{i \in C_l, j \in C_l} \left(A_{ij} - \frac{d_i d_j}{2m} \right)$$

In order to maximize the value, Louvain has two phases.

First, each node is a community. Then the change of the modularity is calculated by removing node i from its own community and moving it to the community of each neighbor j of i. Node i is placed in the community that maximize the change of the modularity score. This process is applied repeatedly and sequentially to all nodes until no modularity increase can occur. Once this local maximum of modularity is hit, the first phase has ended.

The second phase is that the modularity information of the new formed community is recalculated, and the new formed community is treated as a single node for further merge, and repeat phase 1.

4.2 Spectral Clustering

Network-centric level community consider user connection within the entire network globally. It aims to cut the connected network into several disconnected sets. Spectral clustering is one of the network-centric community detection algorithm. It is proposed by Luxburg and it derived from the graph partition. Basically, it needs to find out a partition such that the cut (the total number of edges between two disjoint sets of nodes) is minimized.

However, the minimum cut often return the imbalanced partitions. As is shown in figure 1, one partition has four nodes while the other has only one node.

So we change the object function to consider the size of the community.

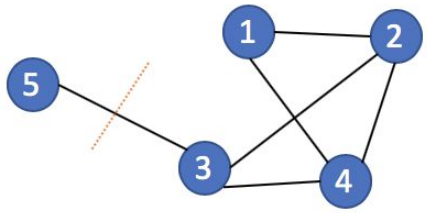


Figure 1: graph partition

$$Ratio\ Cut(\pi) = \frac{1}{k}\sum_{i=1}^{k}\frac{cut(C_i, \overline{C_i})}{|C_i|}$$

$$Normalized\ Cut(\pi) = \frac{1}{k}\sum_{i=1}^{k}\frac{cut(C_i, \overline{C_i})}{vol|C_i|}$$

Where C_i is a community, $|C_i|$ is the number of nodes in C_i, $vol(C_i)$ is the sum of degrees in C_i. Both ratio cut and normalized cut prefer a balanced partition.

Spectral clustering transform the above formula into a more efficient way. It includes three steps.

First, it calculates the adjacent matrix A based on KNN. If one node is in the k nearest neighbor of another node, they are connected.

Second, we project the data into lower dimension space. We notice that data points in the same cluster may also be far away–even farther away than points in different clusters if we using k-means algorithm. For spectral clustering, it aims to project high dimension vectors into lower dimension space so that the two closed nodes can be divided into the same cluster. For this, we compute the Graph Laplacian, which is just another matrix representation of a graph and can be useful in finding interesting properties of a graph. It can be computed by $L = D - A$ where D is the degree matrix. This L is used to calculate the eigenvalue and eigenvector by $Lv = \lambda v$ where v is the eigenvector and λ is the eigenvalue.

Third, we use the eigenvector corresponding to the eigenvalue to assign values to each node.

To get k distinct clusters, we use k-means to cluster eigenvector into k clusters.

4.3 Evaluation

After we applied the two above community detection algorithms, we need to evaluate the quality of the community. Notice that in each community, users are more likely to talk about similar topics within the same community than between different communities. Users in the same community have strong connections than users among different communities, and they would like to follow others' tweets and retweet on the initial tweets, which improves the probability of similar topics in the same community. We

use Latent Dirichlet Allocation (LDA, will be discussed in section 5) as our topic modeling model, and apply it on the cleaned tweets and get the top 20 salient words from each topic.
We found that for louvain, the topic words extracted from each topic are less comprehensive than those with spectral clustering. So we used spectral clustering as our community detection algorithm.

5. Latent Dirichlet Allocation (LDA)

latent Dirichlet allocation (LDA) is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. After we find the communities, we recommend friends to users depend on the similarity between their tweeter topic.
We extract 1023 users' tweets. For each user, we extract 200 most recent tweets and make it into one document.
Then, we generate 10 topics of the whole community which contains all of the documents. After that we categorize each user's twitter document by the 10 topics depend on the probability of its topic distribution. If users' documents are categorized to the same topic, that means they share similar topic in their tweets and they may share similar interests.

5.1Data Preparation

We used regular expression to clean the url links, emoji and references of some other users' name which have no relevance to topics.
Then we use NLTK Tokenize and filter the stop-words, which are not critical for generating topics. After that we use NLTK. Lemmatization Stemming to reduce the words redundancy because the words share the same meaningful base with their inflexion and derivation forms. Lemmatization can efficiently remove the affixes and obtain the correct base form of the words.
146 of user's documents contain 'nan' after data processing. So, we dropped the documents which values are considered missing, and use the remaining 877 documents to do the topic modeling.
Before doing the topic modeling, we use Tf-idfVectorizer to vectorized the documents to emphasized the important words in the document. According to the size of the document, we choose 2000 features to do vectorizing.
The parameter of Tf-idfVectorizer is:
No_features = 2000
(max_df=0.90,min_df=2, use_idf=True,analyzer="word",max_features=no_features,stop_words = None)

5.2 Result of Topic Modeling

Topic 0: Politics

trump music vine good thank vote want presid go taught awesom know congress watch would thought countri actor need new

Topic 1: Portuguese User

bought que não de eu actorslif com uma photo dia look go green sunni film brooklyn show da awesom gojimgo

Topic 2: Vacation

outland thank like one get day go amp happi time see look new watch peopl today good know make great

Topic 3: Vacation

outland new amp day like get go see happi thank great today time season good amaz year look photo one

Topic 4: Podcast

outland get exclusive ya vine netflix amp podcast lol girl drought land roger outlander final saturday like sweet say pray tiger one

Topic 5: Academic

prek student grade school day great que histori ckla read houston fcp lme dear teacher group job grader babi internet

Topic 6: Spanish User

de que la el en lo es por se un para una mi che con si não del todo da

Topic 7: Emotion

happi batb sale zone view road inform ticket roswellnm en bday work weird bu like bitch team heart know readi

Topic 8: Job

thank see impeach job congrat need recent readi still reach gift kudo notredam man note play open post timeless outsid

Topic 9: French User

de le je et la pour pa il que en ça du ce est un une sur memphi fait qui

The explanation of interpreting topic words:

Topic 0 : The topic words contains the name of politicians , the government agency and political activity like “vote”, so it might be reliable to categorized them as politics.

Topic 1: This topic contains Portuguese words, it may indicate a group of people within the community use or at least interest in Portuguese. Topic words also contains words relate to entertainment like film, actors.

Topic 2&3 : Topic2 and Topic3 are quite similar according to the topic words.

Topic 4 : This group of topic words refers to Netflix and podcast, so the topic may refer to TV show or podcast on internet which is relate to going outside, here we summarize this group as podcast.

Topic 6: The topic words in topic6 are all pronouns and conjunctions, but we can group them as one topic which indicate the users who may interest in Spanish or they are Spanish speakers.

Topic 7 : This group of topic words are not obviously have a main topic, it seems like expressing personal emotions because the topic word list contains slangs like “bday” which means bad day, “bitch” and “batb” which is an abbreviation of “beating around the bush”.

Topic 9: The topic words are French words and most of them are pronouns and conjunctions. Only one words “memphi” indicate the topic may refer to a location. However, we prefer to group it as French User which also reflect the users’ interest.

5.3 Grouping users by topics

Here is the result of grouping users by the topic of their tweets. Actually, every user’s tweets have probabilities of belonging to every topic. We chose topic that the user’s tweets most likely belongs to by the highest probability. We recommend the users whose tweets are under the same topic.

12 users’ tweets belong to topic9

	user_id	topic
48	795953028209774592	9
92	2594578615	9
121	1374656754	9
292	907339111341707266	9
303	110236326	9
442	899176169177841664	9

833 users’ tweets belong to topic2

	user_id	topic
0	836645421971820544	2
1	735082409864089601	2
2	709808773863501826	2
3	1032413840561106944	2
4	734891283001409536	2

31 users’ tweets belong to topic6

	user_id	topic
23	313620507	6
28	135600161	6
67	790878846694395904	6
131	1032583364505755648	6
133	922874042264313856	6

1 user’s tweets belong to topic1

	user_id	topic
606	1025217760744419328	1

6. Nearby Users

Today, networking has become a necessary socioeconomic activity in order to earn better opportunities. Recommending a user who tweets nearby our considered user can help the user to grow his/her network. This feature of our recommendation system allows a user to find another user who tweeted near the user's location.

In this section, we find the location of the user that is under consideration for recommendation. Based on that location, we find all the recent tweets that are made at that particular location. Then, we extract three tweets from a list of tweets on that location. And finally, we recommend the profiles of theses tweets to our current user in consideration.

The process of data collection can help us here. We collected some fields or attributes of the users in consideration. These fields contain a 'location' field which has the value of the current user's location. We fetch that information of 'location' field for each user from the data we stored in the output file earlier during data collection and do the processing using that location to get the desired output.

To know the processing and mining in detail, we can discuss the following code snippet which is used in our project to recommend the users who tweeted nearby our user node:

```
if user_info['location'] == '':

    items_to_info.append('')
elif user_info['location']!=None:
    items_to_info.append(user_info['location'])

    print(user_info['location'])
    location = user_info['location']
    g = geocoder.osm(location)
    #print(g.bbox)
    if g.northeast != None and g.southwest != None:
        neLat=g.northeast[0]
        neLng=g.northeast[1]
        swLat=g.southwest[0]
        swLng=g.southwest[1]

        result_count=0
        num_results=1
        while result_count < num_results:
            stream.filter(locations=[swLng,swLat,neLng,neLat])
            print(screen_name1)
            items_to_info.append(screen_name1)
            result_count += 1
```

In the function *get_my_object()*, when we retrieve the 'location' attribute of a user, at first we find the bounding box coordinates of that location using the 'geocoder' library of python. The bounding box coordinates are: north-east latitude, north-east longitude, south-west latitude and south-west longitude.

The values of these four coordinates are used as parameters for finding the tweets having location of these coordinates. Here, we have used the Streaming API to fetch the real-time tweets from twitter. We filter the streamed tweets by giving the parameter 'locations' which takes the value of the four coordinates we derived from the user's location. This way, it compares the user's location with the location of current three tweets. if the location turns out to be same, then the profile of user who tweeted the corresponding tweet is retrieved and this profile is recommended to our node user. Also, the recommended user's *screen_name* is stored as 'recommendation' in the output 'csv' file with other attributes of the node user.

7. Popular Users

Popular is a very complex concept in social media.Several indices can influence a person's popularity in the community.We should consider about the following two points:

1) The status or influence of a person in the community

2) The ability of a person to produce creative content

The reason for these two points is that a person will be attracted by an attractive person or attractive content especially in a social media.We have already talked about how to find a creative topic in a community.In this chapter,we will discuss how to find a person have more influence in one's community.In our program,we try two way to find such a person and recommend to the user.

7.1 Recommend based on followers or friends

In a Twitter community,it is not a hard work to get the number of friends and followers.These two data are the most intuitive data that reflects one's popularity.As we know,a star often has thousands of followers.A person who have a large quantity of followers usually have the characteristics below:

1) The person always share some attractive content or he has great comment on the event.

2) This person is very eye-catching.He could be a celebrity or the leader of a community.

A person who have a large quantity of friends have the characteristics below:

1) The person is interested in socializing so that he try his best to make more friends.

2) The person take part in several activity in the community,so he knows more people in the community

3) The person has malicious target.For example,the one wants to advertise to more people.

In this situation,it is hard for us to avoid the third possibility.However,these two variable are still very valuable for us.We check every person in the community to get the number of friends and followers.We choose the person who have more friends or followers to recommend to the user.We also give the detail information of these people.Here is an example of the running result in our program:

```
Most friends
No.1
User_ID=336445332
closeness=0.34748641304347827
ScreenName=SamHeughan
friends=510505
followers=1017
His text=Ah the Big Yin! https://t.co/wL1HDHrM3v
No.2
User_ID=1510918164
closeness=0.34748641304347827
ScreenName=Outlander_STARZ
friends=447419
followers=909
His text=The start of production on #Outlander Season 5 is just around the corner, clan! (: @caitrionambalfe) https://t.co/WanwlU
No.3
User_ID=174972753
closeness=0.34748641304347827
ScreenName=Writer_DG
friends=281538
followers=188
His text=RT @ScottJKyle1: A few years ago (2013) I produced a show called "How to make a killing in Bollywood" we took it to the

Most followers
No.1
User_ID=1004075138
closeness=0.34748641304347827
ScreenName=MatthewSlivar
friends=250395
followers=113935
His text=https://t.co/0Z4I31kOS8
No.2
User_ID=2975420093
closeness=0.34748641304347827
ScreenName=edappadvice
friends=51904
followers=32669
His text=9 Things You Should Say to Your Kids Every Day https://t.co/4H54j101QW
No.3
User_ID=1703282845
closeness=0.47186346863468637
ScreenName=outlanderpod
friends=38805
followers=12446
His text=RT @outlanderpod: Love the #Outlander series on Starz? So do we! Join in the conversation and let us know what YOU think
```

The result of friends recommended by "friends and followers"

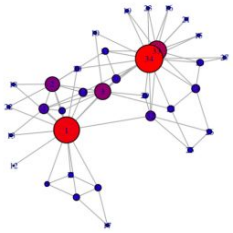
7.2 Recommend based on centrality analysis

The definition of Centrality analysis is about identifying the most "important" nodes in a network.A person has a high number of the centrality means it is closer to the centre of the community.The three most common methods to calculate the centrality are like below:Degree Centrality,Betweenness Centrality,Closeness Centrality.We use one of these three kinds of method to recommend friends to users.

7.2.1 Degree centrality

Degree centrality measure ranks nodes with more connections higher in terms of centrality.In the twitter situation,degree centrality means the sum of numbers of friends and followers in the community.The formula to calculate the degree centrality is like below:

$$C_d^{Norm}(v_i) = \frac{d_i}{n-1}$$



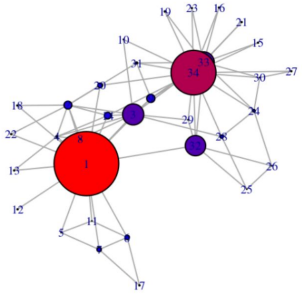
An example of Degree centrality

The example of Degree centrality shows that we can find a person with more friends and followers.However,this kind of work has already done in the chapter 7.1 in our program.So we only use it to ensure the correctness of chapter 7.1 in our program.

7.2.2 Betweenness Centrality

Node betweenness counts the number of shortest paths in a network that will pass a node.In the twitter situation,it means the number of the relation those who can be connect with each other through the person simply.The formula to calculate the betweenness centrality is like below:

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



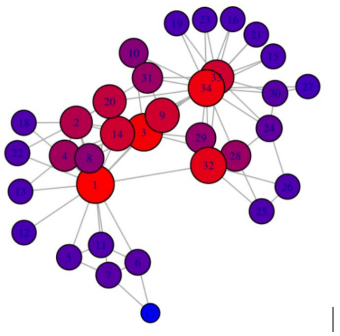
An example of Betweenness Centrality

The example of betweenness centrality shows that this kind of centrality focus on the bridge function in a graph.If our target is to find more friends through the person,it will be a better choice.In this situation,we want to find the centre person we can get more interesting information.So we choose the Closeness Centrality.

7.2.3 Closeness Centrality

Closeness centrality measures how close a node is to all the other nodes. In the twitter situation,it means one can hear another through least friends.The formula to calculate the closeness centrality is like below:

$$C(x) = \frac{1}{\sum_{y} d(y,x)}$$



An example of Closeness Centrality

The example of Closeness Centrality shows that the person with a higher closeness centrality value can get valuable information from anywhere in the community quickly.Once follow him,we can quickly know the events happen recently.That is better for us to recommend to users.The picture below shows the result of our program.The program find the nodes with the three highest value of closeness centrality.

```
Centrality
No.1
User_ID=3315380664
closeness=0.8926701570680629
ScreenName=BraedenClarke
friends=889
followers=24
His text=https://t.co/r5zoZbuQqt
No.2
User_ID=186704655
closeness=0.47186346863468637
ScreenName=kenpriebe
friends=457
followers=825
His text=Just watched MAGICAL MYSTERY TOUR for the first time. Man, the Beatles were so high when they made that.
No.3
User_ID=2279874295
closeness=0.47186346863468637
ScreenName=zGibbyy
friends=70
followers=0
His text=YYC
```

The result of friends recommended based on "centrality analysis"

8. Hashtag Recommendation

Hashtag is an entity in twitter that shows the events which the user mentions in the tweet.The form in twitter is those words after hashtag signal "#".So It is easy to find the number of hashtag in the community.The number is meaningful.If the frequency of a word is highest in the community.It means that it is the most popular topic recently.

We have already complete the community detection in chapter 4.So the topic based on the hashtag could attract the user with a high probability.For example,the most frequency hashtag is "Trump".The user's

community is considered interested in politics.When we recommend “Trump” to the user.He could be interested in it too.The result in our program is like below,we recommend three most popular hashtag to the users.

```
The most popular three words=
['#outlander', '#highlands', '#gameofthrones']
```

The result of hashtag recommended

9. Conclusion

In this paper,we designed a system to recommend users or hashtags to a user in twitter.What we have done is like below:

(1)Used spectral and louvain clustering algorithm to do the community detection

(2)Evaluated the quality of community separation based on latent dirichlet allocation

(3)Recommended twitter users based on the location

(4)Recommended twitter users based on the popularity

(5)Recommended the hashtag based on the frequency

As a result,the user will obtain the most professional recommendation benefiting from our program.Whether he wants to find a friend nearby or he wants to find a celebrity in his community,he will reach the goal.

In the future,we can improve our program by giving weight to every feature.The recommendation could be more intelligent. However,we need to work hard to find the suitable weight of every feature Including but not limited to location and popularity of the user.

References

U. v. Luxburg. A tutorial on spectral clustering. Statistics and Computing, 17(4):395–416, 2007.

Tang, Lei, and Huan Liu. Community Detection and Mining in Social Media. Morgan & Claypool,2(1):13-16, 2010.

Zafarani, Reza. Social Media Mining: an Introduction.,3(1):73-85,2014.

De Meo, Pasquale, et al. "Generalized louvain method for community detection in large networks." 2011 11th International Conference on Intelligent Systems Design and Applications. IEEE, 2011.

Rajasingh, Indra, Bharati Rajan, and Florence Isido. "Betweeness-centrality of grid networks." 2009 International Conference on Computer Technology and Development. Vol. 1. IEEE, 2009.

Okamoto, Kazuya, Wei Chen, and Xiang-Yang Li. "Ranking of closeness centrality for large-scale social networks." International Workshop on Frontiers in Algorithmics. Springer, Berlin, Heidelberg, 2008.

van Gennip, Yves, et al. "Community detection using spectral clustering on sparse geosocial data." SIAM Journal on Applied Mathematics 73.1 (2013): 67-83.

Opsahl, Tore, Filip Agneessens, and John Skvoretz. "Node centrality in weighted networks: Generalizing degree and shortest paths." *Social networks* 32.3 (2010): 245-251.

Sunghwan Mac Kim, Stephen Wan, Cecile Paris, Brian Jin and Bella Robinson. "The Effects of Data Collection Methods in Twitter." Proceedings of 2016 EMNLP Workshop on Natural Language Processing and Computational Social Science, pages 86-91, November 5, 2016.

Liu, Yan, Alexandru Niculescu-Mizil, and Wojciech Gryc. "Topic-link LDA: joint models of topic and author community." proceedings of the 26th annual international conference on machine learning. ACM, 2009.