

# Twitter User Recommendations

Mohan Varma Gottumukkala (mg1265), Rutgers, The State University of New Jersey  
Rasagna Veeramallu (rv358), Rutgers, The State University of New Jersey

---

Social Networking is gaining popularity and has now become a part of everyday life. It helps people to be connected to their family and friends. Social Networks like Twitter can be represented as graphs, where each user is represented as a vertex and each edge represents one user following another user of interest. Our work focuses on suggesting probable users to follow based on the certain user attributes related to each user. We are using Apache Spark to analyze the dataset and to create user graph that models the social network. We are using machine learning techniques to recommend users to follow, based on users he/she is currently following.

Categories and Subject Descriptors: **[Recommendation Systems]**: Machine Learning—*Spark, Python*

---

## 1. INTRODUCTION

With the advent of social media and social networking services, it is now easy to keep up with activities of people of interest. Twitter is one of the many social networking services available today. Twitter is an online news and social networking service where users interact with each other through messages called 'tweets'. The users of this social networking service can follow other registered users of their interest and will be able to read all the tweets posted by users they follow. This kind of relationships can be modelled as a directed graph. Twitter users are modelled as vertices of the graph and the edges represent one user following another user.

Since going through twitter feed takes our time, it is very important to find users whose tweets are interesting to follow. Otherwise, our feeds would be filled with a lot of uninteresting tweets. Our project tries to solve this problem by suggesting who to follow based on users' interests. We find these user interests based on users he/she is following currently and other related attributes like the popularity of user.

## 2. DATA

The data used for modelling the social network graph is collected using Twitter API - TwitterStream and Tweepy. The attributes for each user were collected using Tweepy and the list of users that each user follows is retrieved using TwitterStream.

TwitterStream provides methods to sample a large collection of random tweets, using which the user-ids of the users posting the tweet are collected. Using this user-id, the other attributes related to the user are extracted using Tweepy.

To generate the graph, we carefully started with a user who follows a large number of users and used the list of users followed by this user to get other users and the users they follow. We continued this process to collect a large number of Twitter registered users and their followers.

1:2 •

## 2.1 Twitter Users - Vertices

The attributes extracted for each user include:

- (1) user-id: A unique number to identify each user.
- (2) screen-name: A pseudo name for each user.
- (3) name: The actual name of each user.
- (4) followers\_count: The number of users following this user. Larger followers count indicate a popular user.
- (5) listed\_count: Number of lists a user is part of. Higher listed\_count value represents popular user.
- (6) statuses\_count: Number of tweets posted by a user. Higher statuses\_count indicate a active user.
- (7) friends\_count: Number of different users this user follows and those who follow the user back.
- (8) favorites\_count: Number of user's tweets marked as favorite.
- (9) location: the geographical location of the user

All there attributed extracted are written to a comma-separated values (csv) file.

## 2.2 Relationships - Edges

Each edge in our directed social network graph represent one user following another user. This list of users each user is following is extracted using TwitterStream library provided by Twitter and these follower-followee relations are written to a comma-separated values (csv) file.

## 2.3 Graph

Using the information extracted above, a directed graph is constructed. The vertices of the graph represent users and the directed edge represents a follower-followee relationship. Our graph consists of 23585 users or vertices and 1048575 edges.

A snap-shot of the whole graph is shown below. It is generated using Gephi. The black spots represent each user node in the network graph.

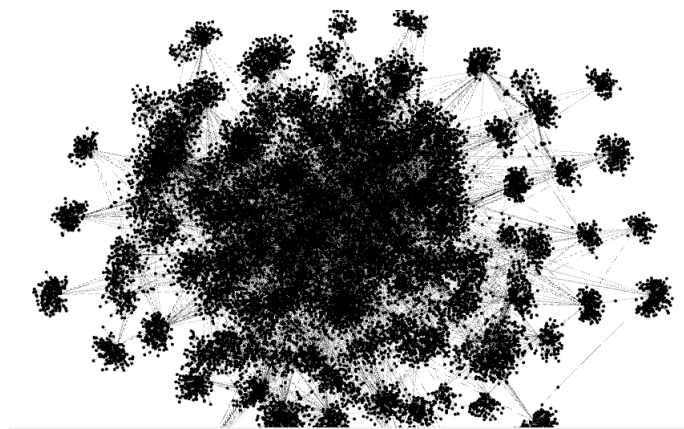


Fig. 1. User Graph

### 3. ANALYSING DATA

Since the real world social network graphs are sparse, we validated our data selection method (mentioned in DATA section above) by analyzing various attributes of the user graph. We used Apache Spark for the analysis.

#### 3.1 In-Degree:

In the graph modelled based on the follower-followee relations extracted using Twitter API, in-degree of a vertex or a user represents number of other users following this specific user or vertex.

We analysed the in-degree of all the users extracted for our project and constructed a histogram analysing the number of users vs number of their followers.

The histogram obtained is shown below (Fig. 2):

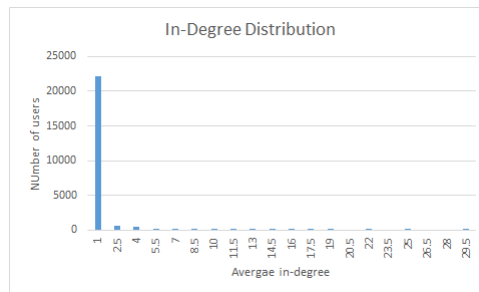


Fig. 2. In Degree Distribution

As the histogram above shows, there are few users with large number of followers and large number of users with small number of followers. Users with high in-degree are considered popular users.

#### 3.2 Out-Degree:

In the above graph, out-degree of each user represents number of other users that this user is following. We analysed the out-degrees of all the users and constructed a histogram that shows the number of users vs number of their followers.

The histogram is shown below (Fig. 3).

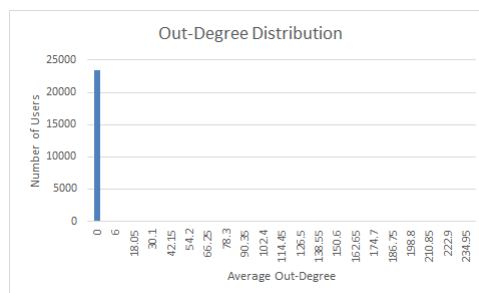


Fig. 3. Out Degree Distribution

Users with high out-degrees are following many users.

3.3 Page Rank:

Page rank for each vertex is calculated based on the in-coming and out-going links for each vertex in the graph. Page rank represents the popularity and prominence of each user in our social network. The histogram representing the page rank distribution of different users is shown below (Fig. 4):

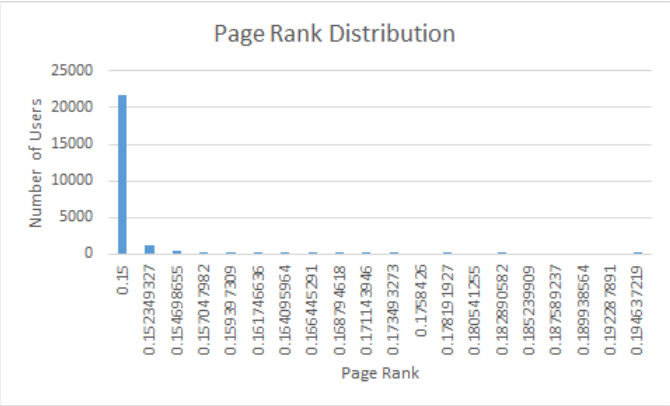


Fig. 4. Page Rank Distribution

The table below (Fig. 5) shows the number of vertices or users for different values of page rank.

AveragePageRank	UsersCount
0.15	21788
0.1523493272911164	1194
0.15469865458223275	320
0.15704798187334915	141
0.15939730916446554	58
0.1617466364555819	37
0.1640959637466983	23
0.1664452910378147	8
0.16879461832893106	5
0.17114394562004745	7
0.17349327291116384	1
0.1758426002022802	0
0.1781919274933966	1
0.18054125478451297	0
0.18289058207562936	1
0.18523990936674575	0
0.18758923665786215	0
0.1899385639489785	0
0.19228789124009488	0
0.1946372185312113	1

Fig. 5. Page Rank Table

After this analysis using Spark, we are confident that our graph structure is good for training and testing using Machine Learning techniques.

## 4. CANDIDATE SELECTION

In order to generate a score for the likelihood of the existence of a connection or edge between two nodes, i.e predict probable edges, we need to prune the set of candidates to run our prediction algorithm. We used an approach based on random walks to prune our potential candidates among all the users.

### 4.1 Personalized Score Propagation

Each node is assigned an initial score of 1.0. In the first iteration, the user or the node retains half of its score and propagates the other half of it's score equally to all its neighbors, both the followers and the followees. The followee nodes in-turn continues the process of retaining half of their score and propagating the other half of their score to their neighbors in the subsequent iterations.

This process is followed for each node in our graph to select the candidate nodes or users to run the prediction algorithm. The users with top scores generated in the above explained process are considered as the potential candidates to run the machine learning algorithm to predict the probable edges or connections.

## 5. FEATURE SELECTION

To represent the existence and non-existence of edges, i.e. the followee-follower relationship between two users, we extracted multiple attributes related to both the concerned users and the existence of a relation between them. For each edge in the graph, the follower is called the source and the node that it follows or is connected - the followee is called the destination node. The features extracted for each edge are as follows:

- (1) Does the source user follow the destination user(Does the edge exist?)
- (2) Does the destination user follow the source user
- (3) number of followers for the source user
- (4) number of followers for the destination user
- (5) number of followees for the source user
- (6) number of followees for the destination user
- (7) ratio of number of followers to number of followees for source user
- (8) ratio of number of followers to number of followees for destination user
- (9) score for the source user calculated using score propagation
- (10) score for the destination user calculated using score propagation
- (11) number of followers in common between source and the destination users
- (12) number of followees in common between the source and the destination users
- (13) percentage of followers who are friends for source user
- (14) percentage of followers who are friends for destination user
- (15) percentage of favorite tweets out of all tweets for source user
- (16) percentage of favorite tweets out of all tweets for destination user
- (17) page rank score of source user
- (18) page rank score of destination user
- (19) katz similarity score for source user
- (20) katz similarity score for destination user

The above mentioned features are extracted for each user node against the set of candidate nodes selected using the personalized score propagation method mentioned above. These features are fed to a machine learning algorithm to generate the likelihood of the existence of an edge between a pair of nodes. The above generated features are used for training our machine learning algorithm

## 6. PREDICTION

For predicting the probable users to follow for a particular user, we pruned the potential candidates using the above score propagation algorithm mention above, and extracted the features for that particular user against all the potential candidates. We used the clustering algorithm - called K-Means clustering to predict the likelihood of the existence of the edge between each pair of users generated above.

The K-Means algorithm clusters data by trying to separate samples into different groups of equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-squares. We trained our classifier to segregate the pair of users into two clusters, one cluster which indicates potential edge between the pair and the other cluster that indicates the absence of the edge.

We used the K-Means clustering algorithm provided by Scikit-Learn library in Python to classify our edges or connection between potential users.

## 7. VISUALIZATION

We are using Python's NetworkX and Matplotlib libraries to visualize the User Graph. Since its very difficult to visualize the whole graph, we are displaying two level neighbours of source node (node for which we are predicting the probable edges). Source node is colored in green. All the nodes within two edges are colored in yellow. Edges from source node to these yellow nodes are colored in red. The recommended nodes are colored in cyan and edges representing follow suggestions are colored in blue.

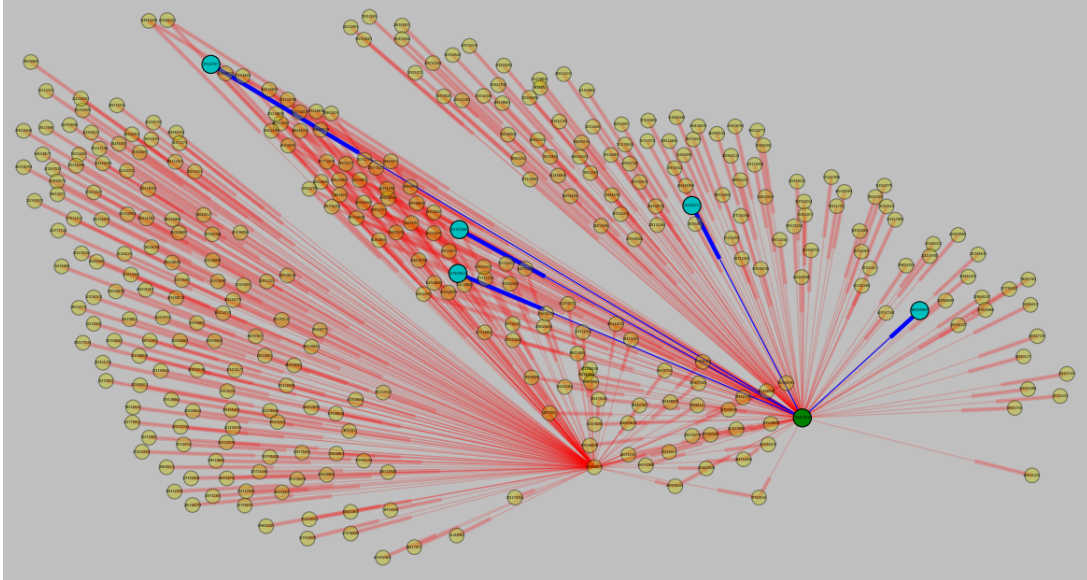


Fig. 6. Visualization of existing and recommended connections

Graph generated using these libraries can be zoomed in for greater details. Here is the graph zoomed in to examine a particular section of the graph.

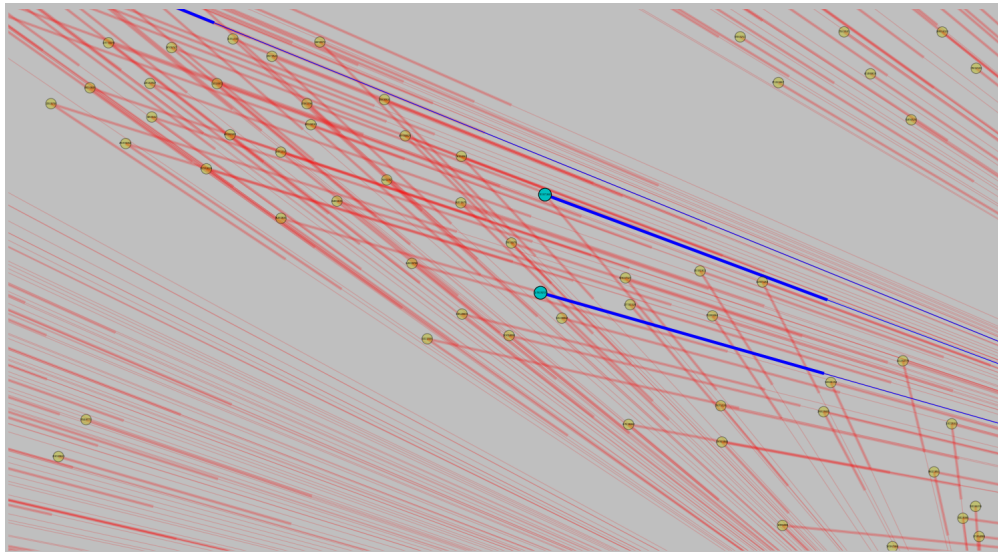


Fig. 7. Examining the graph in more detail

On observing the predicted edges, users connected by users that source node is connected to are also shown. This illustrates that our prediction algorithm is working efficiently as there is a high probability of such users being similar to the source user and are probable connections.

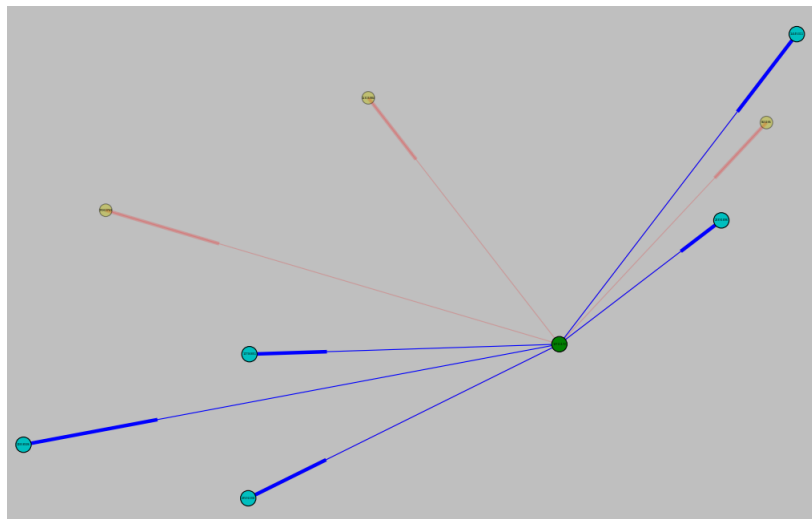


Fig. 8. User following very few users

For users who follow only quite users or users with very few number of followers, we recommend users who are followed by large number of followers or popular users, as shown in figure 8.

## 8. RESULTS

After training the K-Means classifier, we input the user ID of the user for which we want to predict the potential users to follow. A resultant graph as shown in the visualization above is displayed to the user.

To calculate the accuracy of our prediction process, we created a validation set using 10% of the data initially extracted. When measures on the validation set, the accuracy reported was 84.86%.

This accuracy has been achieved by re-training the classifier based on the results and probable predictions in each step of the prediction process. The training has been carried out in 3 iterations.

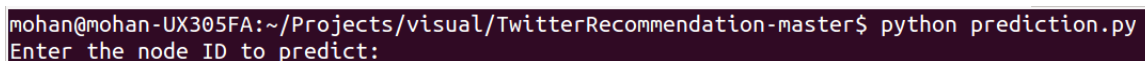
Among all the features extracted to train the K-Means clustering classifier, few features played a prominent role in the probable edges. These features are as follows:

- (1) page rank of the source and destination nodes
- (2) katz similarity measure of the source and destination nodes
- (3) ratio of number of followers to number of followees for source and destination nodes

The above mentioned features carried the highest weight in deciding the probable edges.

## 9. RUNNING THE CODE

All the required python packages (scikit-learn, networkx, matplotlib) need to be installed to run this project. Run predict.py file to input node-id and once its entered, the visualization is generated with recommended node-ids to follow.



```
mohan@mohan-UX305FA:~/Projects/visual/TwitterRecommendation-master$ python prediction.py
Enter the node ID to predict:
```

Fig. 9. Running the code

We are operating on node-ids rather than twitter handles as some of the handles are found to be missing from the data.

## 10. TECHNICAL CHALLENGES

### 10.1 Twitter API request rate limits

Twitter imposes a limit on the rate at which we request their data. It is 15 per request per hour per account. We wrote a script that runs hourly to pull the required data. We also used multiple twitter account credentials and ran this script with multiple credentials.



## 10.2 Constructing a graph that is close to real-world social network

Since real-world social networks graphs are sparse, we made effort to construct such a graph by pulling the right data as described in the data section. We also validated this data using Apache Spark by generating various statistics about the graph.

## 11. SOURCE CODE

Source code for this project is hosted on Github at <https://github.com/RasagnaVeeramallu/TwitterRecommendation>

## 12. REFERENCES

<https://www.kaggle.com/c/FacebookRecruiting/discussion/2082>  
<https://gephi.org/users/tutorial-visualization>  
<http://blog.echen.me/2012/07/31/edge-prediction-in-a-social-graph-my-solution-to-facebooks-user-recommendation-contest-on-kaggle/>  
<http://socialmedia-class.org/twittertutorial.html>  
<https://github.com/tweepy/tweepy/tree/master/tweepy>  
<https://spark.apache.org/docs/1.1.0/graphx-programming-guide.html>  
<http://networkx.readthedocs.io/en/networkx-1.10/>  
<https://en.wikipedia.org/wiki/Twitter>  
<http://scikit-learn.org/stable/modules/clustering.html#clustering>  
<https://www.cs.umd.edu/class/spring2008/cmsc828g/Slides/link-prediction.pdf>  
<https://link.springer.com/book/10.1007/978-3-319-22735-1>  
[https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.algorithms centrality.katz\\_centrality.html](https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.algorithms centrality.katz_centrality.html)  
[https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.algorithms link\\_analysis.page\\_rank.html](https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.algorithms link_analysis.page_rank.html)  
<https://networkx.github.io/documentation/networkx-1.10/reference/algorithms.html>  
<https://www.cs.umd.edu/class/spring2008/cmsc828g/Slides/link-prediction.pdf>