Web Science

Network Based Social Media Analytics

Ibai Castells Bengoa, GUID: 2268745J

Github repository (includes sample data in tweets.json):
https://github.com/ibaiC/twitterCrawler

March 2020

1. Introduction:

The software developed for this coursework consists of a twitter crawler and analysis modules. The first part is a crawler that combines both the streaming API as well as the REST API of twitter developer services. The combination of both allows one to gather as many tweets as possible in a set timeframe. The second part of the work consists of data analysis. The first type of analysis is K-Means clustering which was performed using code from a previous work on k-means clustering of tweets (https://github.com/achyutb6/tweets-k-means). Slight modification of the code was required to work with my own setup, but this seemed to perform reasonably well. Data collection using the crawler script was done over the period of 1 hour, this resulted in approximately one hundred thousand tweets.

2. Data Crawling

a. Streaming API

For this part of the coursework the tweepy module from python was used. The consumer and API keys were used to authenticate and then the word “coronavirus” was tracked, and the language of tweets was filtered for English tweets only as this was most appropriate for the performance of clustering later. The streaming API consisted of a constant stream of 1% of the tweets found with that keyword. This stream did not return all the tweets with that keyword at a moment in time but rather a single random tweet that matched the filtering criteria. The limitations for this were only the number of keywords, user IDs and location boxes provided for tweet filtering. This was not really a limitation in my case as using the trending word of “coronavirus” I was already receiving a large volume of tweets constantly and using more filters did not increase the number of tweets I was receiving significantly.

b. Hybrid crawling

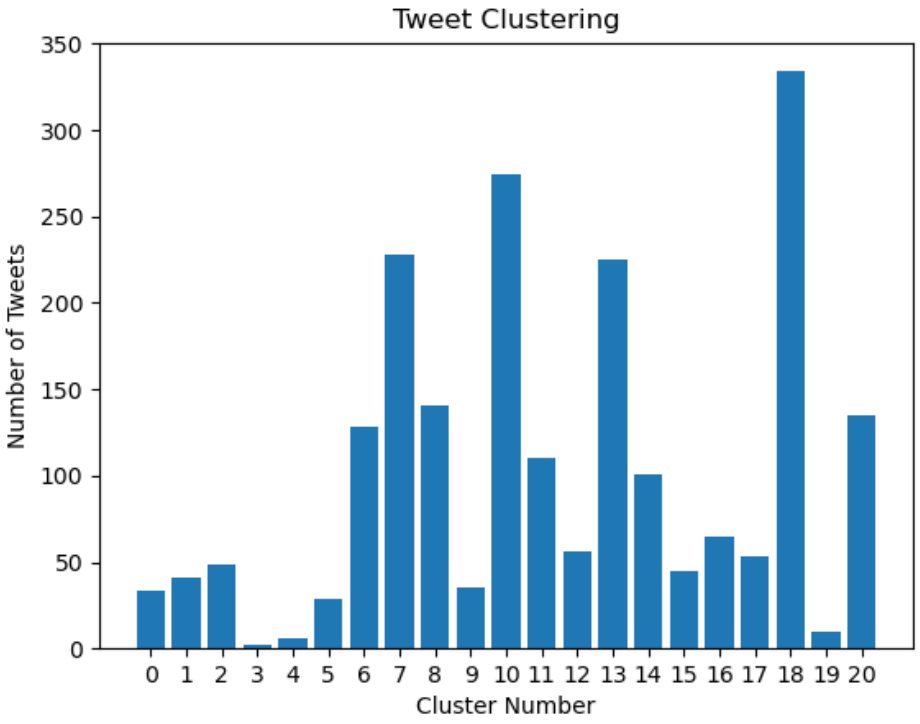
In order to improve the performance of the crawler, REST API probes were used to gather as many tweets on the trending topic as possible along with the streaming API. The REST probes were setup to filter on the same keywords and language as the stream as this appeared to gather the highest volume of tweets after trying different keyword combinations and user IDs. I found that following users returned the lowest tweets per second and following keywords and hashtags gave the highest volume of tweets per second. Overall using a combination of REST probes and Streaming API gave the highest volume of tweets per second. This was the ideal combination with the limitations of free developer accounts.

3. Tweet Grouping

The method for tweet clustering involved 3 steps. First, the tweets were tokenized using regular expressions. This was done to remove all stopwords, urls and redundant words such as “RT” for retweets and @username for user mentions. The reason for this is that this type of text found in tweets has no meaningful semantics which means that it would have made the clustering more unreliable and inaccurate. The second step was identifying centroids on which to base the clustering. These seeds were computed randomly with different k values. It was found that a higher number of clusters generally resulted in more consistent and precise clusters however with such a large dataset, this required a high number of clusters (greater than 10% of total tweets). This high number of clusters results in so many different groups that it stops being meaningful to the human mind and it becomes unclear how the clusters are different from each other. Additionally, the third-party k-means clustering software encountered errors when using a high number of clusters, therefore I used 1% of total tweets as the number of clusters which appeared to work well. The next step after generating the initial seeds was to compute the Jaccard distance of each tweet to the centroids. Each tweet was then clustered using the k-means algorithm based on its smallest Jaccard distance to the possible centroids. Finally, the sum of squared errors was computed to get a measure of reliability. It was found that with a high number of clusters, the SSE value was lower. The output of the clustering program is found in the output folder of the project and it is in the format of:

Cluster number [tweet id, tweet id,…]
SSE value

The clustering results for 2100 tweets with 21 clusters were plotted and an SSE value of 1501.8 was found.



Hashtag Grouping

I wrote a script saved as analysis.py to find the most frequent hashtags for each cluster. The method used consisted of taking the hashtags of each tweet in a cluster, adding them to an array and finally sorting the array by frequency and taking the set of that array to remove duplicates. This was time efficient and simple to do and yielded the following results for the data sample provided.

These results can be seen on the cluster_hashtags.txt file in the output folder. This file outputs the top 10 hashtags for each cluster and returns empty sets for clusters where no tweets come with hashtags.

The total number of hashtags is: 125

The average hashtags per clusters were: 5.9

The min number of hashtags per group was: 0

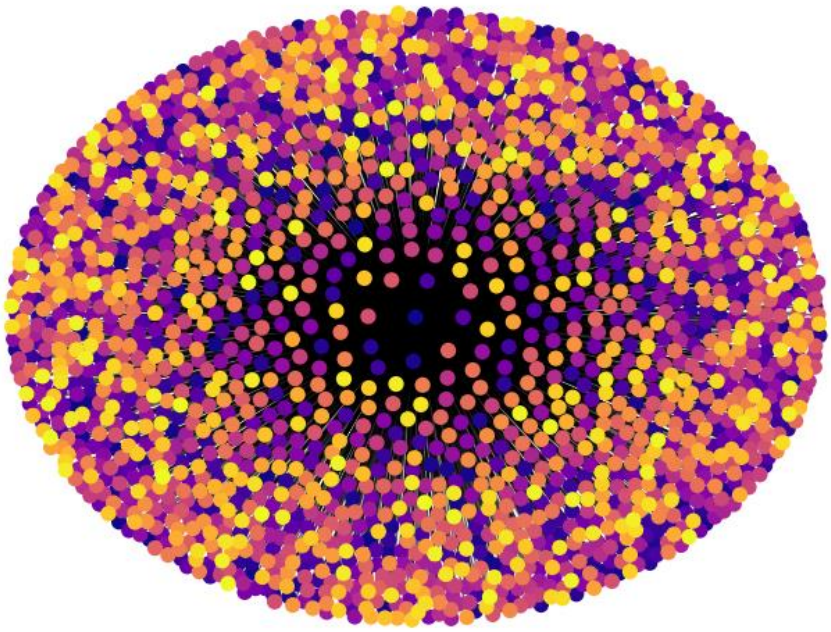
The max number of hashtags per group was: 10

These results are for the clustered data. For the individual tweets before clustering, the tweet with most hashtags was 7. Most tweets did not have any hashtag. This drove the average hashtags per tweet to 1 hashtag. The most frequent hashtag was “#coronavirus”.

4. User Interactions

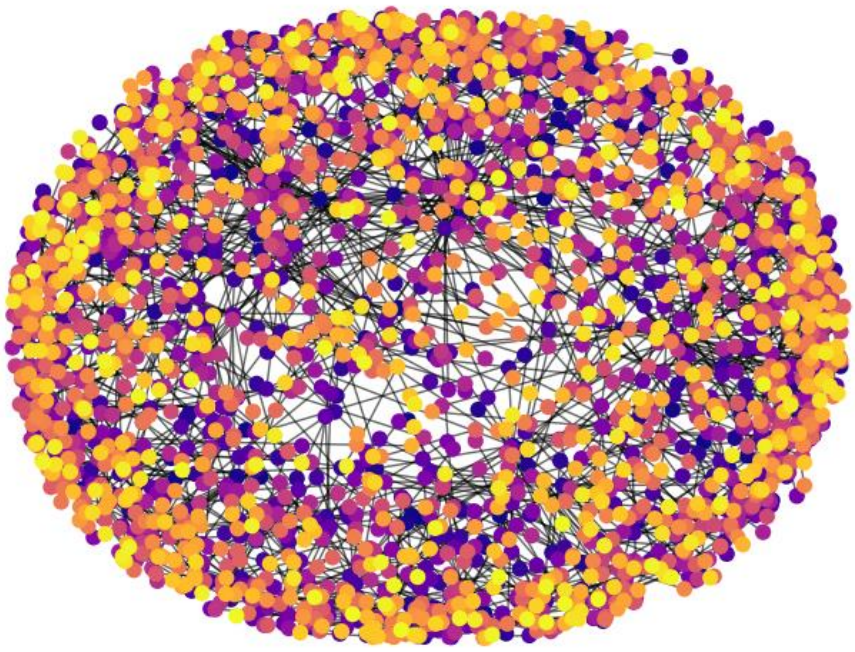
User interactions were mapped using network, a python library used for network mapping. Using the different interaction attributes of each tweet, a graph was generated for each type of interaction. These four different interaction types recorded and graphed are quoted tweets, replies, retweets and user mentions. The graphs can be seen below as well as in the output folder of the project.

User mentions:



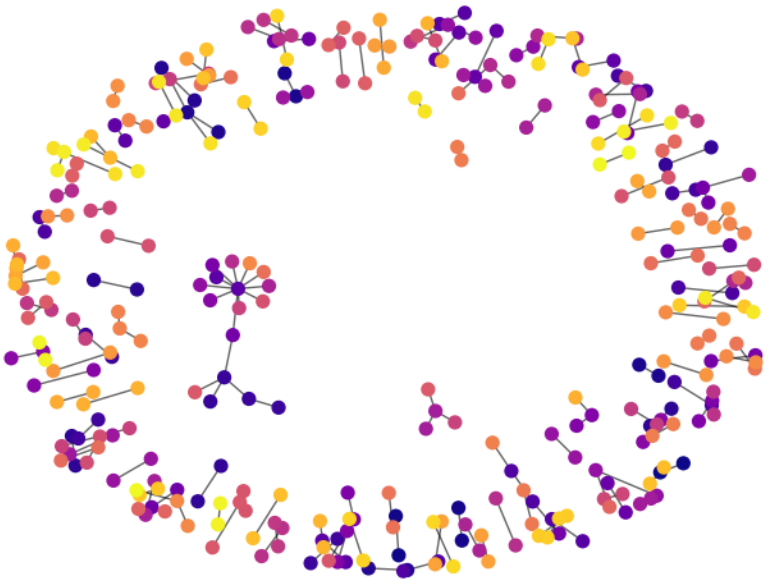
There appears to be a central user being mentioned constantly and then many instances of different users being mentioned few times. The closer to the centre a user is, the more mentioned it is and more connected to other users.

Retweets



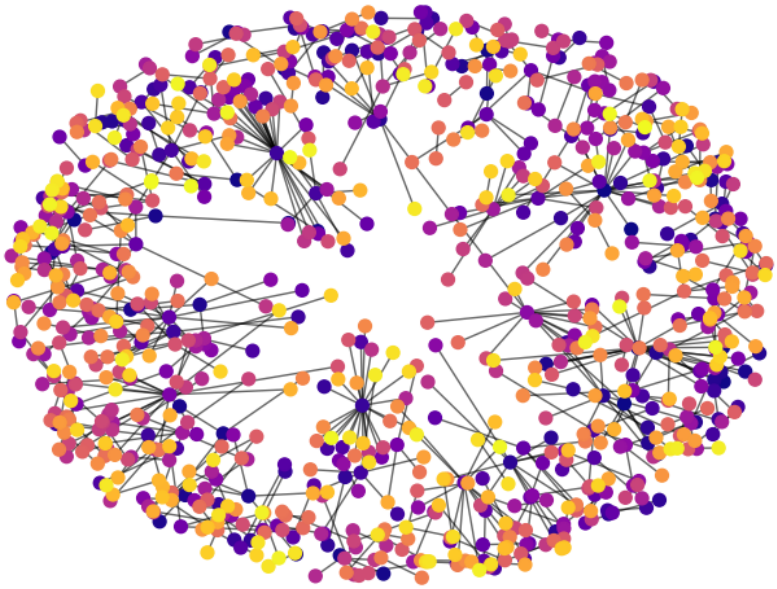
The retweet network appears to be more diverse than the user mentions as there is a bigger spread closer to the centre. Retweets with more frequency are more frequent and the range of retweet frequency is smaller than the range of user mention variation.

Replies



The small number of points in the graph show that replies are an infrequent action for the tweets in this topic. There appears to be many different connections with more links per node than in other interaction types.

Quoted



The nodes on this graph appear to be more evenly spaced out than in other interactions. Showing that the frequency of quoted tweets is relatively uniform. Additionally, the nodes appear to have many edges suggesting that many people are quoting one person and one person is quoting many people.

In terms of the hashtag data, the cluster_hashtags.txt output file clusters the hashtags into the groups that they are often found in and the hashtags that they are frequently found with. This was presented earlier in this report and covers both that section as well as this one.

5. Network Analysis

For this section data of each graph was found and possible triad combinations were generated using networkx triad functions. The data can be seen below. The methods used to get the data for each graph were functions from networkx library.

User mentions:

Nodes: 2012
Edges: 2011
Max Degree: 2011
Min Degree: 1
Mean Degree: 2
Connected: Yes
Triadic census:
{'003': 1353433165, '012': 0, '102': 0, '021D': 0, '021U': 0, '021C': 0, '111D': 0, '111U': 0, '030T': 0, '030C': 0, '201': 2021055, '120D': 0, '120U': 0, '120C': 0, '210': 0, '300': 0}

Retweets

Nodes: 2614
Edges: 1696
Max Degree: 29
Min Degree: 1
Mean Degree: 1.2
Connected: No
Triadic census:
{'003': 2969066309, '012': 0, '102': 4423358, '021D': 0, '021U': 0, '021C': 0, '111D': 0, '111U': 0, '030T': 0, '030C': 0, '201': 3297, '120D': 0, '120U': 0, '120C': 0, '210': 0, '300': 0}

Replies

Nodes: 68
Edges: 35
Max Degree: 3
Min Degree: 1
Mean Degree: 1
Connected: No
Triadic census:
{'003': 47809, '012': 0, '102': 2304, '021D': 0, '021U': 0, '021C': 0, '111D': 0, '111U': 0, '030T': 0, '030C': 0, '201': 3, '120D': 0, '120U': 0, '120C': 0, '210': 0, '300': 0}

Quoted

Nodes: 934
Edges: 628
Max Degree: 29
Min Degree: 1
Mean Degree: 1.3
Connected: No
Triadic census:
{'003': 134777451, '012': 0, '102': 581570, '021D': 0, '021U': 0, '021C': 0, '111D': 0, '111U': 0, '030T': 0, '030C': 0, '201': 1863, '120D': 0, '120U': 0, '120C': 0, '210': 0, '300': 0}