KU LEUVEN

Analysis of Large-Scale Social Networks

Project

Twitter Network Analysis

Prof. Dr. Bart Thijs

Group members:

Name	Student ID	Course ID
Triguna Bangalore Narasaraj	r0648544	H0T27a
Swarnalata Patra	r0729319	H0T27a
Anthoula Mountzouri	r0736452	H0T27a

Disclaimer

The analysis in this report is strictly only for academic purpose only. The report mentions several twitter usernames, and we do not intend to release these outside academia. The authors of this report will not be responsible for any leakage of sensitive information.

Table of Contents

Overview

The project is about analysing a network of twitter users who have regularly tweeted mentioning a hashtag #Brexit [17] during the month of April 2019 in English language. Brexit is currently a hot topic among the twitter users and has an influence on the entire European Union. We have also done a sentiment analysis [27] of these twitter users about Brexit. The analysis conducted in this report is in a very broad perspective. It gives a platform for future analysis related to specific application. Future work could be done using our analysis for different applications like fraud detection, political campaigning, or detecting the source of fake news, etc.

The fundamental insight of a social network analysis is that a person's position in the network is an indicator of how influential they are. This is because influential people are more likely to acquire connections, but also because having more connections in itself makes one more influential.

It has been shown that all the users talking about Brexit are bound together in a web of follower-following relationships, despite not necessarily sharing anything else in common. Hence, we show in our analysis, how common interests connect the different users around the globe in similar social communities.

We have used the network data to get some more concrete insights in the network. For example, identifying influencers, fake profiles, segment audiences, detect different communities and understand what content is interesting to them, how similar are the users in a community w.r.t their sentiments.

We are using various algorithms in our project to show the different relationships and their influence in the network. This is helpful to identify the people and reward them based on having lots of followers or having relationship with others who are themselves well connected. These influential users play a crucial role in the society as they can spread news, and/or promote ideas and inventions across the world.

Data collection

Approach

The data is collected using tweepy [12] library in Python. In order to collect the tweets, it is required to have a twitter developer account. Once the account is set up, the twitter api would provide a set of keys using which it is possible to get the tweets for the specified hashtag (#). The data that is collected using tweepy are about the twitter users who has tweeted about #Brexit in the English language for the whole month of April 2019. Only the original tweets are considered, meaning retweets are not collected. But the amount of retweet counts for a given tweet is recorded.

The attributes of the collected data are:

- **created_at**: (Timestamp) that indicates date & time of creation of the tweet.
  E.g.: 2019-05-23 23:59:58
- **screen_name**: (Text) is the twitter account id name of a user.
  E.g.: screen_name of Donald Trump is 'realDonaldTrump'.
- **location**: (Text) is geographic location of users. This is set by users themselves.
- **followers_count**: (Int) indicates number of twitter users who are following the given user.
- **friends_count**: (Int) indicates how many twitter users the given user follow.
- **retweet_count**: (Int) indicates the number of times the given tweet is retweeted by others.

- **text**: (Text) contains the actual tweet.
  E.g.: '#May has failed in delivering #Brexit #EU'
- **tags**: (List of Strings) indicates other hashtags present in tweet (text).
  E.g.: For the tweet, '#May has failed in delivering #Brexit #EU', the 'tags' attribute contains the value ['#May','#Brexit','#EU']
- **mentions**: (List of Strings) indicates other twitter users who are mentioned in tweet (text).
  E.g.: For the tweet, '@theresa_may has failed in delivering #Brexit. @Nigel_Farage is to blame', the 'mentions' attribute contains the value ['@theresa_may','@Nigel_Farage']

We've collected all 324,352 original tweets in the English language for the given hashtag. The number of unique #Brexit tweeters are 99,229. Some of them have tweeted only once, and others have tweeted more than once in the month of April 2019. In this study, the aim is to capture the network of twitter users who showed that they care deeply about Brexit by expressing their opinion on twitter for each of the four weeks of April 2019. There are 3,427 different twitter users who has tweeted about Brexit for each of the four weeks. Among them, the aim is to capture the main influencers about Brexit on Twitter. Hence, we select only the twitter users whose tweets are retweeted at least 10 times in each of the four weeks of April. By this criterion, there are 240 twitter users who have tweeted about Brexit every week during the month of April 2019 and whose tweets are retweeted at least 10 times in each of the four weeks.

Sentiment classification

The next step is to find the sentiment polarity for each of the tweets. For that we used textblob [13] package in Python. Textblob library comes with the best of NLTK library [14] and pattern library [15], both of which are extensively used in the field of Natural Language Processing [16]. The first step for sentiment classification is to clean the tweets, where all the emojis, hyper-links, and user-mentions are removed. Then the text is normalized and given to textblob package that does the sentiment classification. It generates two outputs:

i) *sentiment polarity* which ranges between -1 & +1. This shows the emotion about the tweets, where +1 indicates strong belief about the hashtag and -1 indicates the opposite, and

ii) *sentiment subjectivity* which ranges between 0 & 1. 0 indicates that the tweets are factual, and 1 indicates the tweets are very subjective. In this analysis, sentiment subjectivity is not used.

Relation extraction

The next step is to find the relations (following & follower) between these 240 twitter users. To calculate this, we used tweepy library again. For 240 nodes, there are (240 * 239)/2 = 28680 possible (undirected) edges in the network. The data for this is collected in the below format:

- **source_screen_name**: (Text) is the twitter account id name of a user 1.
- **destination_screen_name**: (Text)) is the twitter account id name of a user 2.
- **has_mutual_following**: (Bool) indicates if both the source and destination users follow each other.
- **source_follow_dest**: (Bool) indicates if the source user follows destination user.
- **dest_follow_source**: (Bool) indicates if the destination user follows source user.

In the network, we found that there were only 2349 mutually following undirected edges and 7264 directed edges (both follower & following edge).

Key challenges

- We are using a free version twitter api account. Due to this, there are some limitations (wait rate limit) on the number of tweets that could be collected at a time. On an average it takes about 1 hour to collect 10k tweets. Since we collected over 320k tweets, it took us more than 32 hours to collect the data.
- Also, using the free version, we can only collect tweets for the past seven days, meaning we regularly collected tweets during the month of April.

- In order to find the follower/following (edges) between the twitter users (nodes), we use the free version of twitter api again. We were able to find the follower & following relations between users using *api.show_friendship(source_twitter_user,destination_twitter_user)*. Due to the wait rate limit, we could only find the relations for about 800 combinations of twitter users per hour. Since, we needed to find the relations among 28680 combinations (240*239/2), it took us more than 36 hours to find the edges. This is the *main* reason as to why we only choose 240 nodes for this project. Any further increase in the number of nodes will drastically increase the time to find the (possible) edges.

Network components

Nodes (vertices)

Nodes are the twitter *users* in the network. In this report the analysis is mainly carried out for 240 nodes. These are the users who have regularly expressed their interest in Brexit by tweeting about it every week during the month of April 2019. The opinion of these users are also valuable as their tweets are retweeted at least 10 times for each week. The weight of the node can be represented by either the number of tweets a user made that week, number of retweets by other users on the given user's tweet for that week, or the node weight could simply be the number of followers.

Edges (links)

Edges are the *relationships* between the twitter users such as follower, following, or mutual following. In the network that we are going to analyse there are a total of 7264 follower/following edges (directed) with 2349 mutual following edges (undirected). For undirected edges, the weight of the edges is set to unity. For directed edges the weight is proportional to the number of times a source user mentions the destination user in their tweets.

Algorithms

Centrality measures

Nodes in a social network have different roles and structures within the network. Various centrality measures are used here to identify important nodes or communities in the network.

We focussed on below centrality measures:

Degree centrality

Definition: Degree centrality assigns an importance score based purely on the number of links held by each node. It is the simplest measure of node connectivity. [3]

What it tells us: How many direct connections each node has to other nodes within the network.

When to use it: For finding very connected individuals, popular individuals, individuals who are likely to hold most information or individuals who can quickly connect with the wider network.

Interpretation in the network: For our twitter network, for the undirected graph we are calculating the degree centrality. For the directed graph we are calculating in-degree (followers count) and out-degree (following count) as distinct measures. The in-degree centrality for a node v is the fraction of nodes its incoming edges are connected to. The out-degree centrality for a node v is the fraction of nodes its outgoing edges are connected to. The figure below shows a plot between the users with

different retweet thresholds in x axis and in degree of these nodes(users) in the y axis. The plot in blue is for the users whose tweets have been retweeted atleast 100 times. The green plot indicates the users whose tweets are retweeted atleast 10 times. We can observe in the plot that the blue users have higher number of followers compared to the green ones. So, we can infer that the people having more no. of followers(higher in degree) have more influence, since their tweets have been retweeted atleast 100 times.

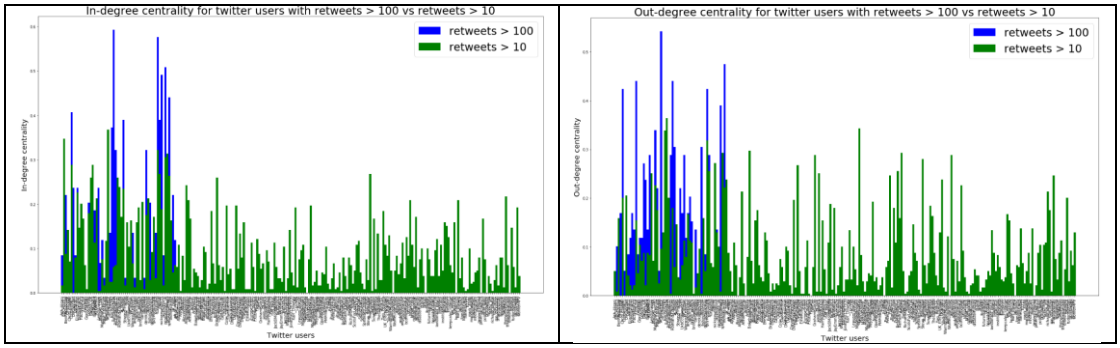

Figure 1 In-degree & out-degree centrality

In our analysis, for the undirected network, it was found that twitter user @spaceangel1964 had the highest degree centrality with a score of 0.35. This user has highest centrality degree as she as almost equally high number of followers (33k) & following(30k). Clearly, the twitter user need not be a highly influential person to have high degree centrality measure as demonstrated by @spaceangel1964.

For the directed network, twitter user @RCorbettMEP has the highest in-degree centrality score of 0.46, and twitter user @spaceangel1964 has the highest out-degree centrality score of 0.53. Twitter users having a high in-degree centrality are considered highly influential in the network and @RCorbettMEP is one among the highest influencers with regards to Brexit, although he is not that relatively popular in twitter compared to other twitter users in the network such as @BBCPolitics or @SkyNews.

Follower/following ratio

Definition: It is the ratio of in degree (followers) to out degree (followings) for any node(user).

What it tells us: It compares the number of users who have subscribed to the updates of user A with the number of users that user A is following. The higher the result, the more people are interested in the user's status updates without the user needing to show interest in their status updates first. If the result is smaller than 1, the user is likely to be considered a mass-follower who follows other users for the sole purpose of gaining more users himself. If the result is much less than 1, it might not even be a real user. It could be a web bot or spider collecting information about trending topics.

When to use it: For finding influential users who can spread their opinions in the network faster than other users.

Interpretation in the network: In Figure 2, we have plotted the sorted result across all the nodes in a network and compared to see how much more or less it is than 1. The red line represents the line that corresponds to ratio = 1. We observe that, among the users who tweet about Brexit, there are very few users have this ratio less than 1. Most of the users in this network are having a great influence as the ratio is quite high as compared with 1.

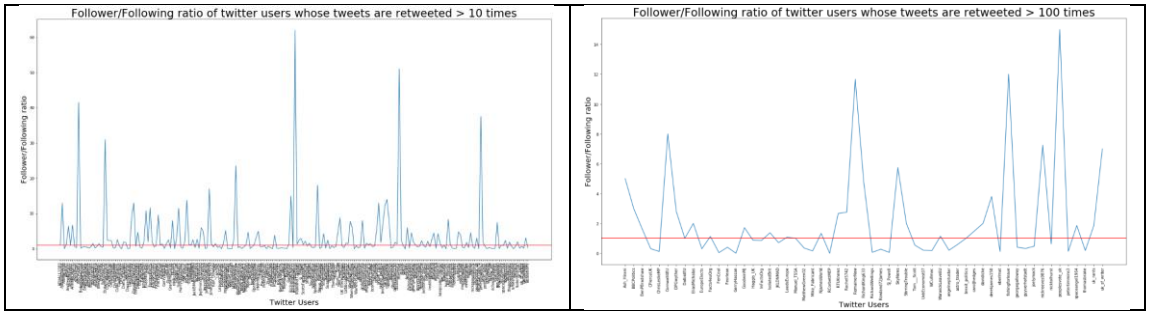

Figure 2 Follower/following ratio

In our analysis we found that twitter user @RichardWellings has the highest following ratio among the network of twitter users who consistently tweets about #Brexit. This shows that other people view @RichardWellings opinions about Brexit, even though this user is not popular having only 26k followers.

Network density

Definition: Network density describes the portion of the *potential* connections in a network that are *actual* connections. A *"potential connection"* is a connection that could potentially exist between two "nodes" – regardless of whether or not it actually does. [3]

What it tells us: The density of real graph refers to the proportion of links that exist in the graph and the maximum number of possible links in the graph. For example: A family reunion has high network density, but a public bus has low network density. Network Density equals to 0 for a graph without any link between nodes and 1 for a completely connected graph.

When to use it: For finding how dense or sparse is the network being analysed. If a network is fully connected, it gives us an idea that all the members in the network are connected to each other, like in a family.

Interpretation in the network: For n users, the number of maximum links are n(n-1) for a directed graph and n(n-1)/2 for an undirected graph. The densities that we observe in the social interaction network of 60 and 240 users are 0.17 and 0.085 for the directed, and 0.34 and 0.171 for the undirected graphs, respectively. This implies the network with more users in twitter is less dense. This is because twitter is a social network where it is highly unlikely that everyone is connected to each other. However, they do share similar interest and thus, they tweet about Brexit even if they are in a very sparse network.

Clustering coefficient

Definition: Clustering Coefficient is a measure of the degree to which nodes in a graph tend to cluster together. It can be calculated w.r.t a single node, called local clustering coefficient and w.r.t the entire network, called global clustering coefficient.

What it tells us: If local clustering coefficient of a node A is 1/3 means 1/3rd of all the possible pairs of friends of A who could be friends, are actually friends. For nodes having less than 2 friends, network assumes the clustering coefficient to be 0. Global clustering coefficient can be calculated by averaging the local coefficients over all the nodes in the graph. Transitivity is another way of calculating the global clustering coefficient, where the nodes with higher degree are weighted higher. So even one node having very high degree will have the influence on the clustering coefficient value compared to many other nodes with lower degree.

When to use it: Nodes with high clustering coefficient should be considered as there is a scope for getting more data of mutual friends from the network.

Interpretation in the network: In our network we observed that the highest clustering coefficient of any node is 0.5. This means at max, only half of the friends of any user are actually friends among themselves. This implies that the network is not completely connected network. However, people who do not know each other, share similar interests and tweet based on the topic which are related to them.

Betweenness centrality

Definition: Betweenness centrality measures the number of times a node lies on the shortest path between other nodes. [3]

What it tells us: This measure shows which nodes act as 'bridges' between nodes in a network. It does this by identifying all the shortest paths and then counting how many times each node falls on one. A high betweenness count could indicate someone holds authority over, or controls collaboration between, disparate clusters in a network; or indicate they are on the periphery of both clusters.

When to use it: For finding the individuals who influence the flow around a system.

Interpretation in the network: We found the top 5 nodes in the network having highest betweenness measure. The profiles of these networks show that they are either some news channels responsible for the breaking news or someone in direct connection with the topic. Ex- Brexit Coordinator for EU. Twitter user @brexit_politics has the highest betweenness centrality measure with a score of 0.11.

Closeness centrality

Definition: This measure scores each node based on their 'closeness' to all other nodes within the network. [3]

What it tells us: This measure calculates the shortest paths between all nodes, then assigns each node a score based on its sum of shortest paths. Closeness centrality can help find good 'broadcasters', but in a highly connected network we will often find all nodes have a similar score. What may be more useful is using Closeness to find influencers within a single cluster.

When to use it: For finding the individuals who are best placed to influence the entire network most quickly.

Interpretation in the network: We sorted the nodes with highest closeness value and observed that the users whose tweets are retweeted 100 times atleast, have higher value of closeness compared to those with 10 times retweeted tweets. This implies that the former ones have more direct connections compared to the later. It is easier for these users to spread the breaking news or create fake news and broadcast to all the users in direct connection and make an impression of it as real. These sources, if detected accurately, can be used to prevent fake news. Twitter user @Rowland72James has the highest closeness centrality measure in our network with a score of 0.51. This user is actually a known debater about the topic and the centrality measure reflects the same.

Page rank or eigen vector centrality

Definition: Like degree centrality, Eigen Centrality measures a node's influence based on the number of links it has to other nodes within the network. It also considers how well connected a node is, and how many links their connections have, and so on through the network.

PageRank is a variant of Eigen Centrality, also assigning nodes a score based on their connections, and their connections' connections. The difference is that PageRank also takes link direction and weights into account – so links can only pass influence in one direction and pass different amounts of influence.

What it tells us: By calculating the extended connections of a node, Eigen Centrality can identify nodes with influence over the whole network, not just those directly connected to it. PageRank uncovers nodes whose influence extends beyond their direct connections into the wider network.

When to use it: Eigen Centrality is a good 'all-round' SNA score, handy for understanding human social networks, but also for understanding networks like malware propagation. Because it factors in directionality and connection weight, PageRank can be helpful for understanding citations and authority.

Interpretation in the network: In our network we have used Page rank to detect the most important nodes in the network and used this ranking for community detection, by using mapequation.org. This will be explained in detail in the later part of the report. In our network, twitter user @spaceangel1964 has the highest eigen vector centrality measure with a score of 0.17.

In the figure below, all the locations of the centrality measures are highlighted.

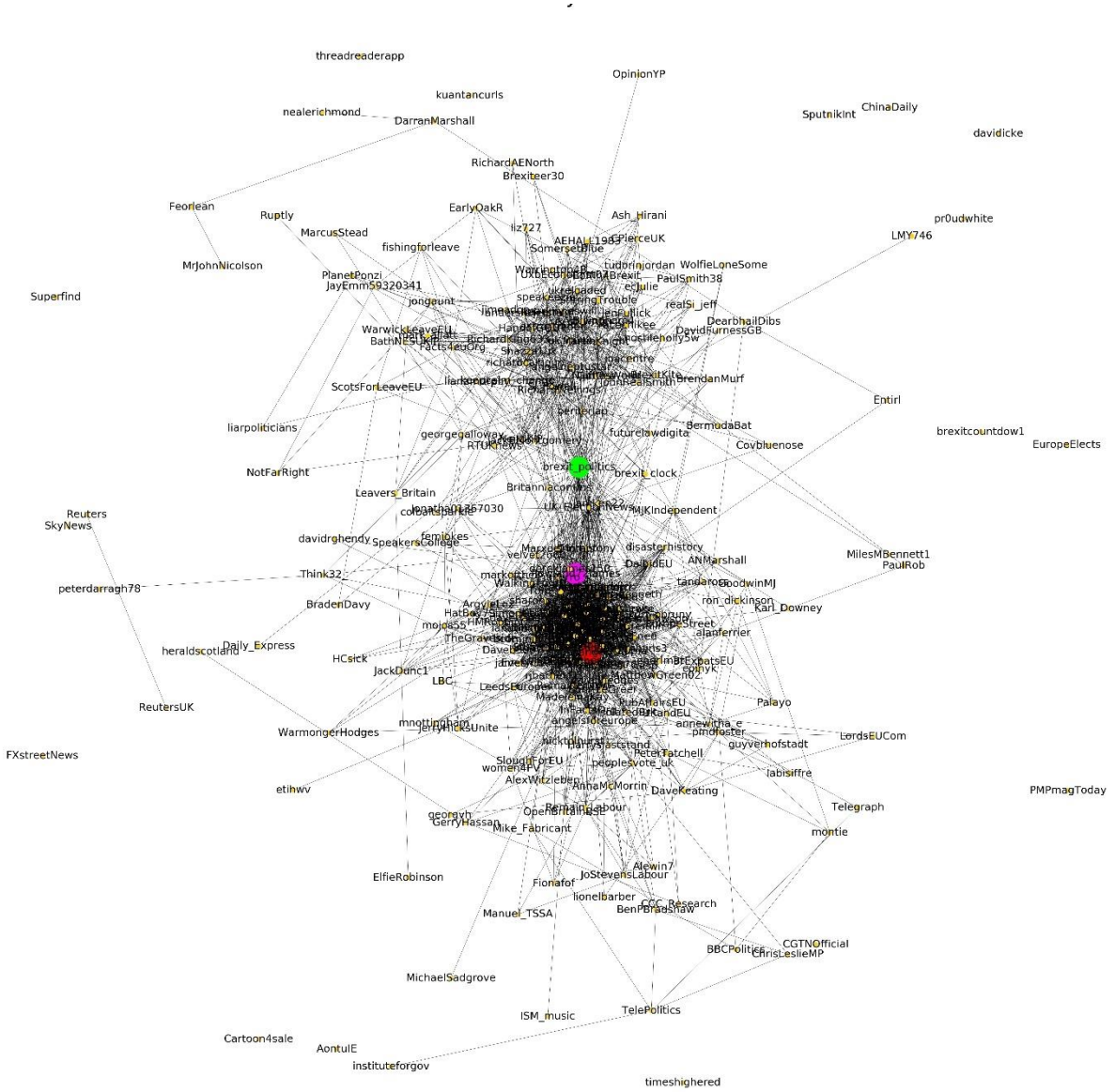

Figure 3 Location of users of high centrality

In the above graph: The red coloured node represents twitter user (@spaceangel1964) of high degree centrality & high eigenvector centrality. The green coloured node represents twitter user (@brexit_politics) of high betweenness centrality. The magenta coloured node represents twitter user (@Rowland72James) of high closeness centrality.

Community detection

Definition: The process of discovering the cohesive groups or clusters in the network is known as community detection. It forms one of the key tasks of Social network analysis. A number of community detection algorithms and methods have been proposed and deployed for the identification of communities. [5]

What it tells us: A community is formed by individuals such that those within a group interact with each other more frequently than with those outside the group. The communities in social networks are analogous to clusters in networks. Communities which have common nodes are called overlapping communities.

When to use it: The detection of communities in social networks can be useful in many applications where group decisions are taken, e.g., multicasting a message of interest to a community instead of sending it to each one in the group or recommending a set of products to a community.

Nowadays, the increasing demands for community detection in large-scale social networks necessitate the use of distributed and scalable methods to detect communities in an effective and efficient manner.

Different Algorithms of community detection and their interpretation in the network:

Label Propagation

Label propagation works by propagating labels throughout the network and forming communities based on this process of label propagation. The idea is, within a cluster, all nodes connected to each other, will eventually converge to the same label. The intuition behind the algorithm is that a single label can quickly become dominant in a densely connected group of nodes but will have trouble crossing a sparsely connected region. The algorithm stops when every node has a label that the maximum no. of their neighbour has.[9]

In comparison with other algorithms label propagation has advantages in its running time and amount of a priori information needed about the network structure (no parameter is required to be known beforehand). The disadvantage is that it produces no unique solution, but an aggregate of many solutions. It tends to merge smaller communities into larger ones.

Label propagation algorithm partitioned the graph into 22 communities. For simplicity, only the communities having more than 3 members are highlighted in the figure below.

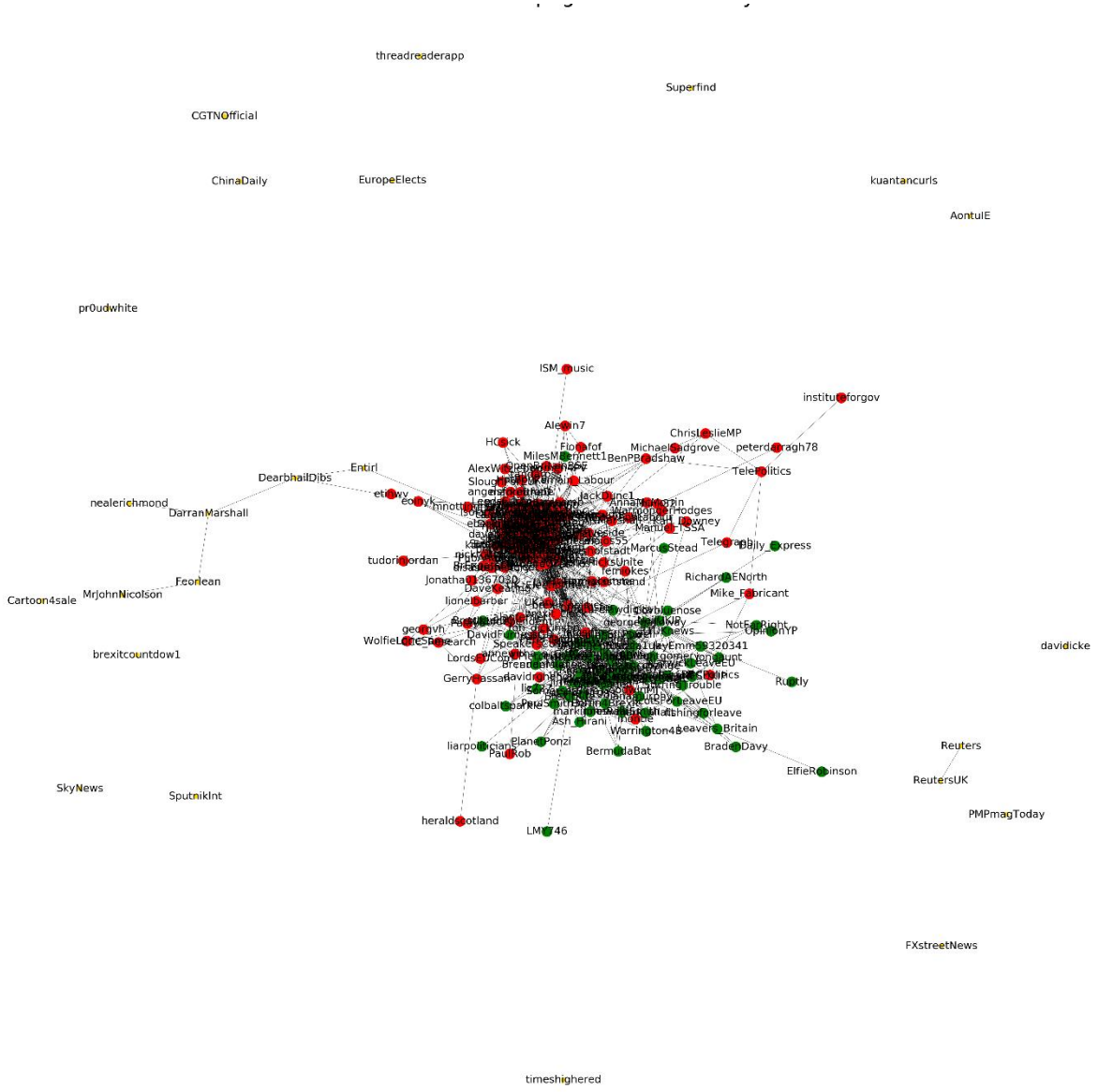

Figure 4 Community detection using label propagation

Kernighan-Lin

Graph partitioning is the process of partitioning a graph into a predefined number of smaller components with specific properties. Kernighan-Lin is a graph partitioning based community detection algorithm. It partitions the nodes of the graph with cost on edges into subsets of given sizes so as to minimize the sum of costs on all edges cut. A major disadvantage of this algorithm is that the number of groups have to be predefined. The algorithm however is quite fast with a worst-case running time of $O(n^2)$. [10]

In the figure below, two communities detected by Kernighan-Lin algorithm is displayed.

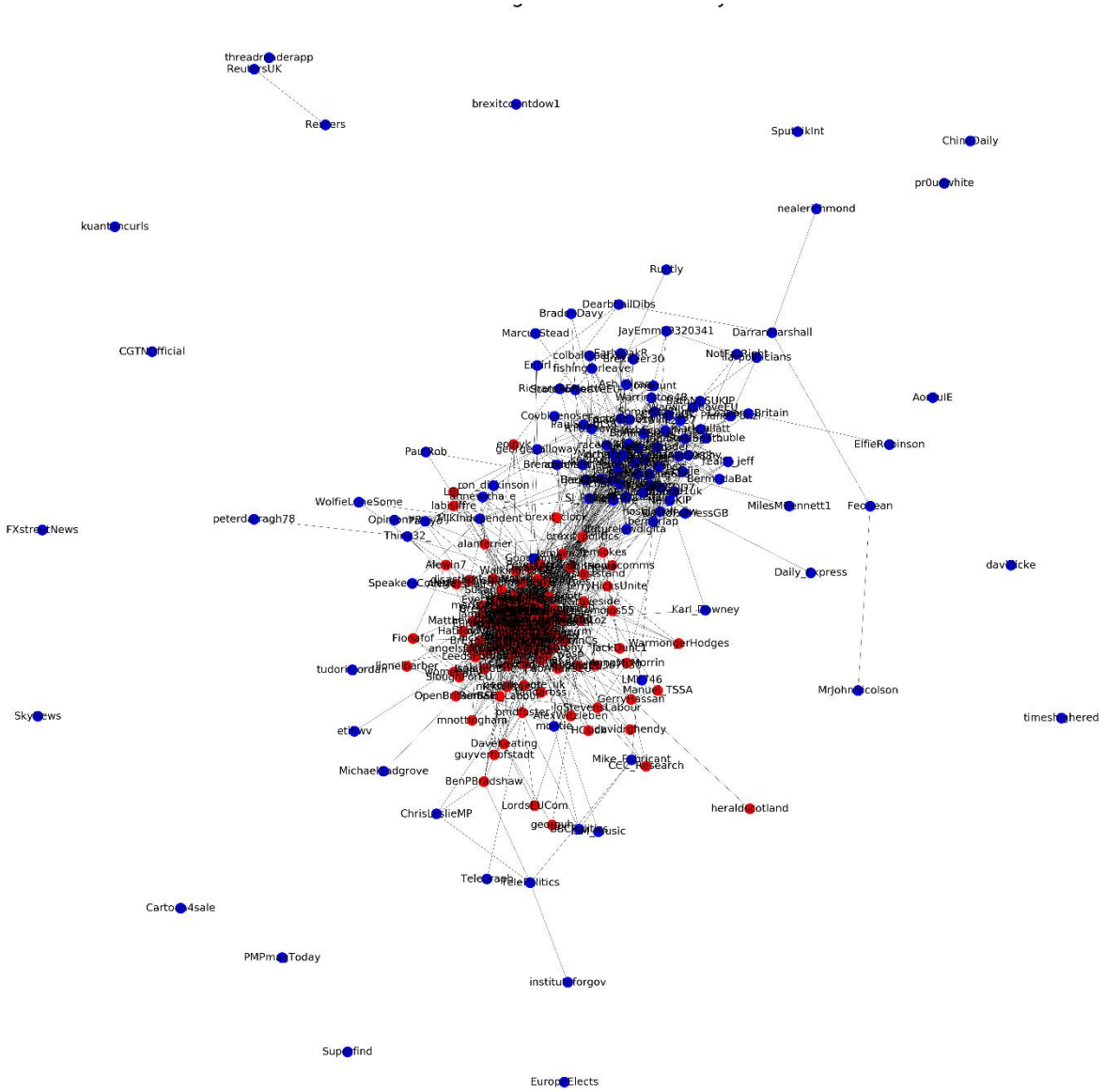

Figure 5 Community detection using Kernighan-Lin

Girvan and Newman

This algorithm is based on the iterative elimination of edges with the highest number of shortest paths that pass through them. This is a divisive hierarchical clustering method based on edge-betweenness. The worst-case time complexity of the edge-betweenness algorithm is O(m^2n). and is O(n^3) for sparse graphs, where m denotes the number of edges and n is the number of vertices. [10] Newman and Girvan first defined a measure known as **'modularity'** to judge the quality of partitions or communities formed, such that high modularity corresponds to a better community structure. Newman also generalized the betweenness algorithm for weighted networks.

The algorithm partitioned the graph into 19 different communities. For simplicity, only the groups having at least three members are displayed in the graph. Only two communities have at least three members. From the plot, the partition looks visually imbalanced, meaning most of the members are in a single group represented by blue colour.

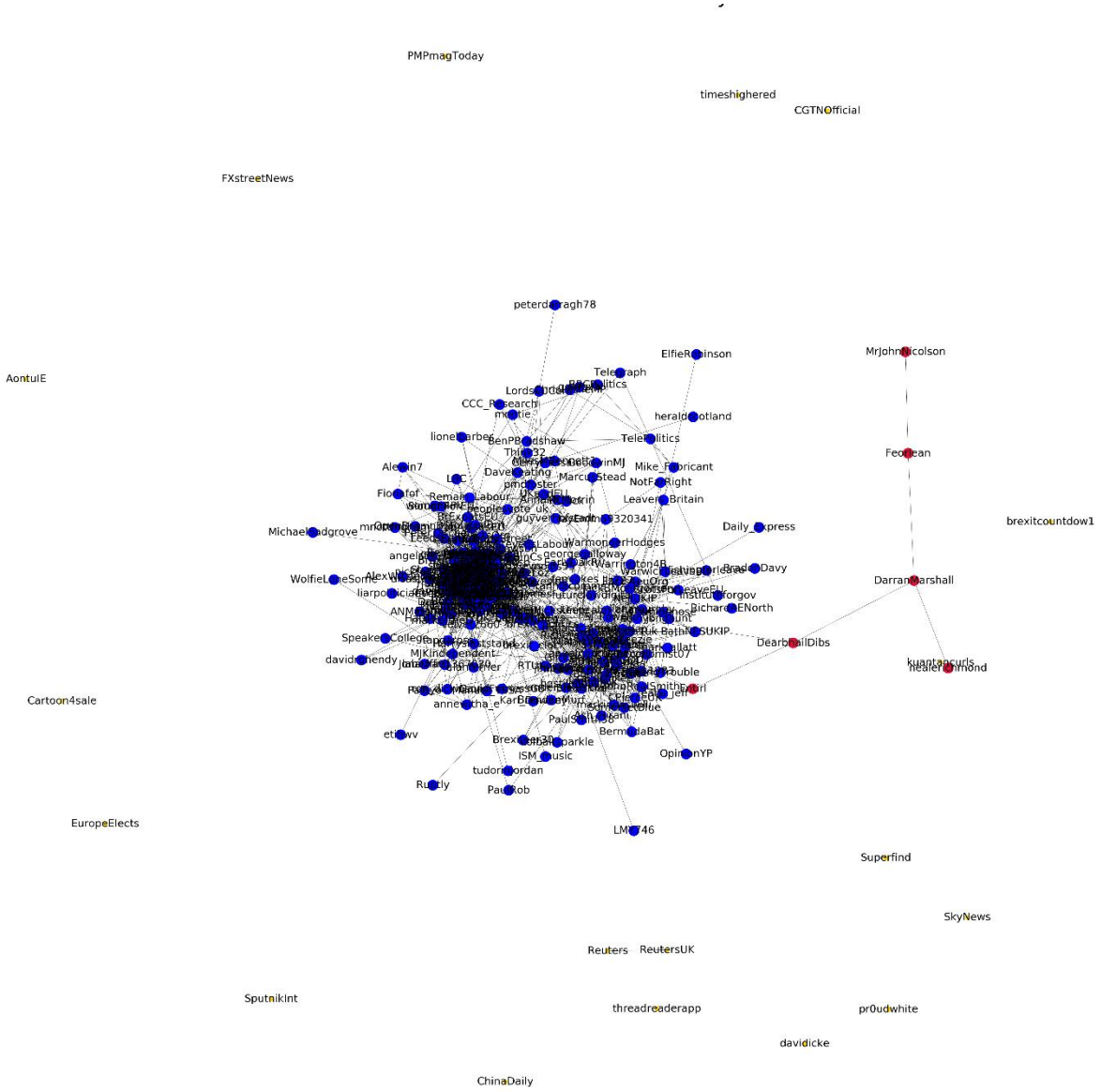

Figure 6 Community detection using Girvan and Newman

K-Clique

This community detection algorithm is mainly used to detect overlapping communities in a network. k-clique is a clique with k nodes.[8] K-clique community is a union of all k-cliques that can be reached from each other through a series of adjacent k-cliques. Two k-cliques are said to be adjacent if they share k-1 nodes. [6] We usually consider maximal cliques for this algorithm. In our network, we got the optimal results with k=5. The different communities are marked with separate colours.

Unlike other community algorithms, k-clique returned the community membership for only 144 out of 240 nodes in the network. Meaning, the did not assign a community to 96 members. For 144 members that are assigned a community, the algorithm partitioned these nodes into three communities and is shown in the figure below.

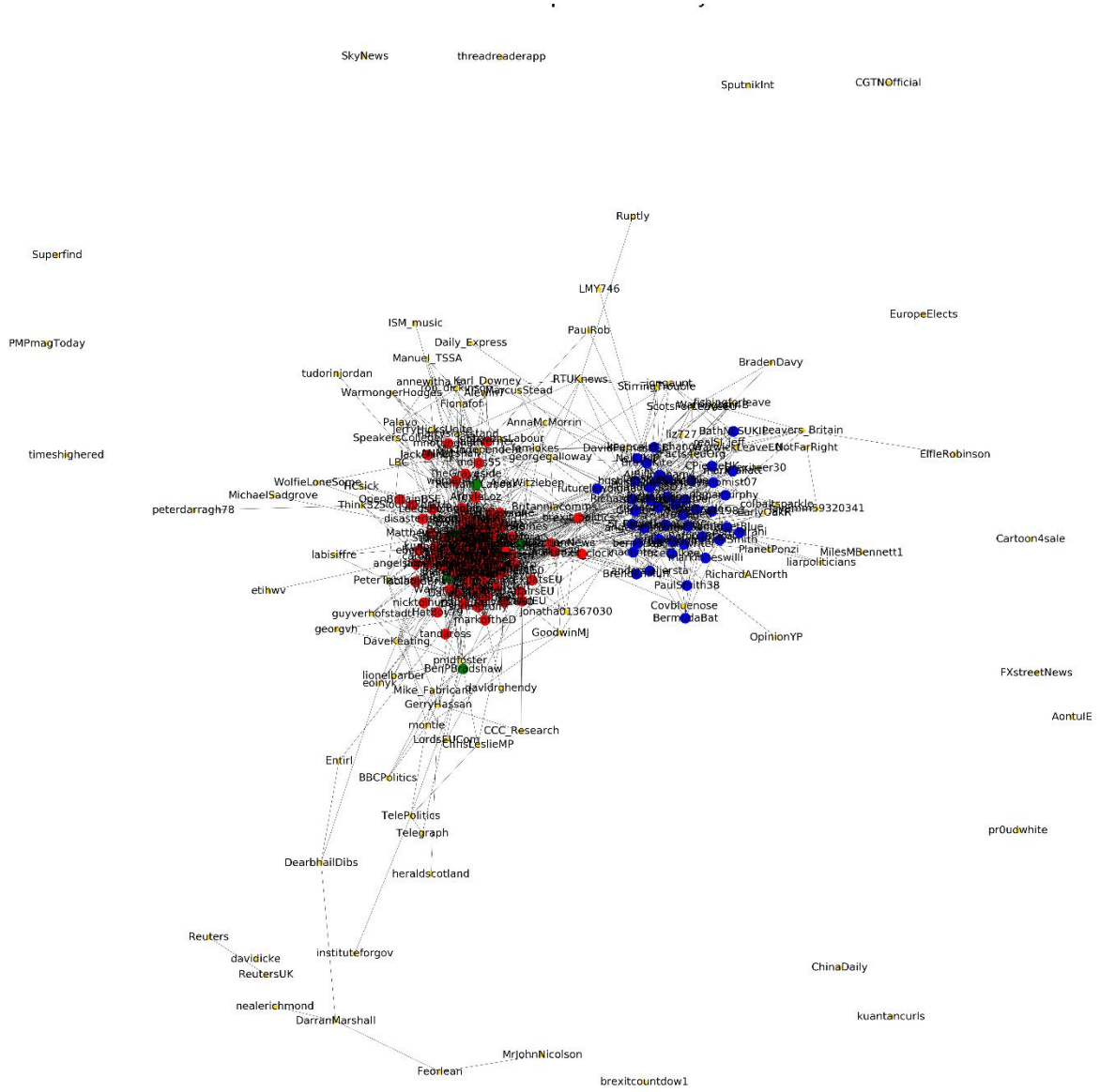

Figure 7 Community detection using k-clique

Modularity based approaches

Modularity has more to do with internal structure of the community rather than with graph partitioning. 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. Basically, it looks for patterns that occur more often than at random.[2] We used 2 types of community detection algorithms based on modularity method.

Greedy modularity method

The algorithm finds communities in graph using Clauset-Newman-Moore greedy modularity maximization. It starts with a state in which each node is in its own community and the algorithm repeatedly joins pairs of communities together to obtain larger ones. At each step, the joined communities are selected by considering the largest increase (or smallest decrease) in modularity. It produces a set of community structures organized hierarchically, with increasing granularity. The one obtaining the maximal modularity is considered as the best. This algorithm is relatively fast; however, the result can be very coarse.[2]

The greedy modularity algorithm partitioned the graph into 23 communities. In the figure below, only those communities are displayed that have at least three members in it. There are five such communities with at least three members. This is the only algorithm that partitioned the network in such a way that five communities has at least three members.

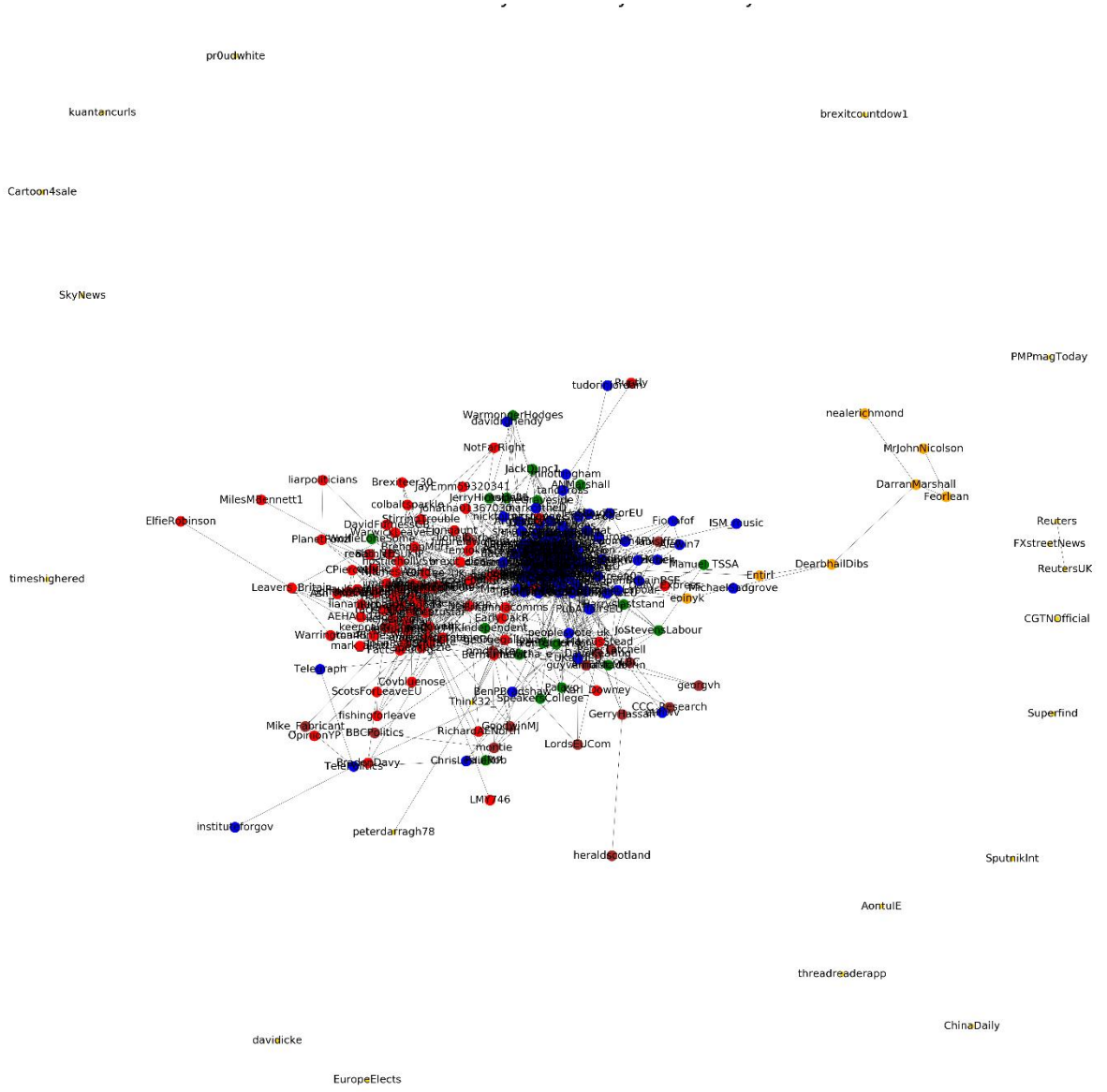

Figure 8 Community detection using greedy modularity method

Louvain method

A very popular heuristic algorithm used for community detection. It maximizes modularity score for each community. The algorithm first assigns each node to its own community and then goes through each node and evaluate the modularity gain from removing the node from its own community and placing to its neighbouring community. The procedure is repeated until the modularity score keeps increasing. When clusters cannot be improved further by moving individual nodes, the Louvain algorithm aggregates the network, so that each cluster in the original network becomes a node in the aggregated network. In the aggregated network, the algorithm then starts to move individual nodes from one cluster to another. By repeating the node movement and aggregation, the Louvain algorithm is able to find high-quality clusters in a short time. Since the Louvain algorithm keeps moving nodes from one

cluster to another, at some point it may move the crucial node to a different cluster, thereby breaking the connectivity of the original cluster. Perhaps surprisingly, the Louvain algorithm cannot fix this shattered connectivity. Also, this method reaches to a local maximum modularity based on the order of nodes chosen. And hence results in different final distributions of communities each time. The issue with this algorithm is they have trouble detecting small communities in large networks.

The Louvain method partitioned the network into 22 communities. In the figure below, only the communities are displayed that have at least three members.

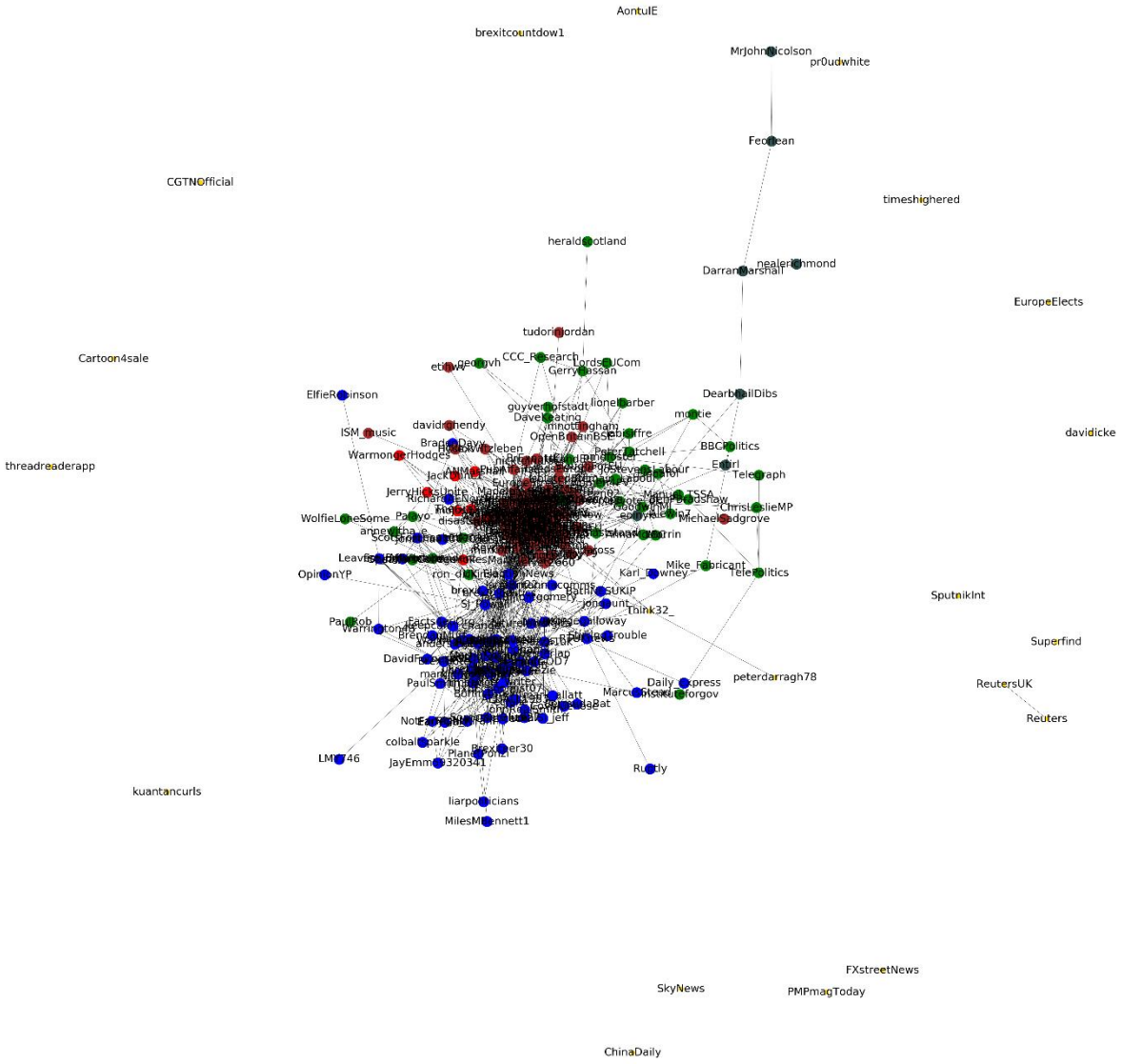

Figure 9 Community detection using Louvain method

Leiden method

The problem of shattered connectivity in Louvain algorithm is fixed in the Leiden algorithm. The Leiden algorithm is able to split clusters instead of only merging them, as is done by the Louvain algorithm. By splitting clusters in a specific way, the Leiden algorithm guarantees that clusters are well-connected. It is impossible to improve the quality of the clusters by moving one or more nodes from one cluster to another. This is a strong property of the Leiden algorithm. It states that the clusters it finds are not too far from optimal. Finally, rather than continuously checking for all nodes in a network

whether they can be moved to a different cluster, as is done in the Louvain algorithm, the Leiden algorithm performs this check only for so-called *unstable nodes*. As a result, the Leiden algorithm does not only find higher quality clusters than the Louvain algorithm, it also does so in much less time.[11]

The Leiden method partitioned the network into 22 communities. In the figure below, only the communities are displayed that have at least three members.

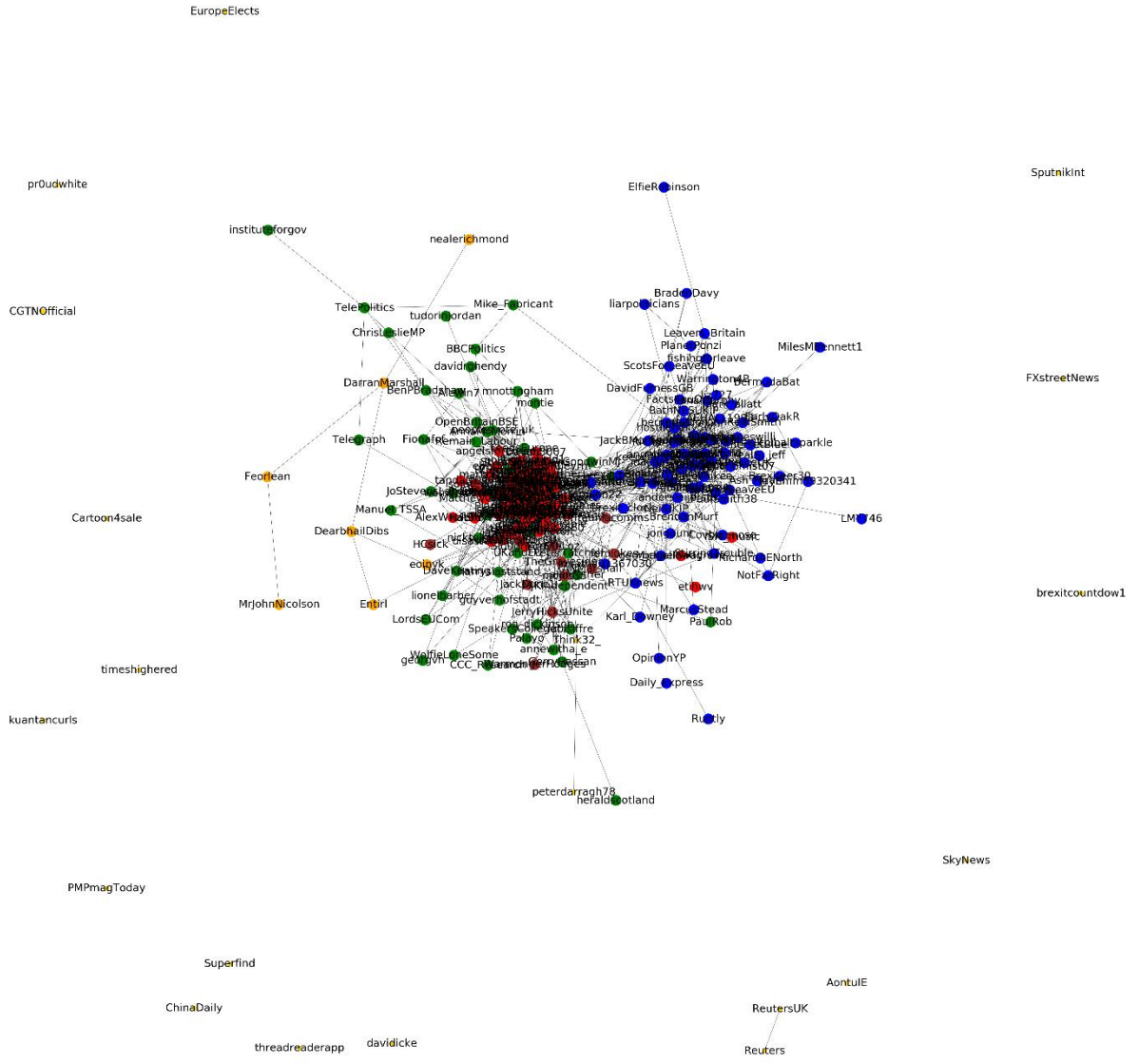

Figure 10 Community detection using Leiden method

Map equation:

We also experimented with map equation for community detection. Page rank is used as the basis of clustering here. This method decomposes a network into modules by optimally compressing a description of information flows on the network. Different modules are derived from the node with the highest PageRank within the module. The aggregated PageRank of all nodes within the module and the per step exit flow from the module is then computed. The map equation attends to patterns of flow on the network while the modularity maximization approach does not. Hence, the two methods can yield dramatically different results for some network structures. [1]

The resolution limit [25] of the map equation is set by the total number of links between modules instead of the total number of links in the full network as for modularity. This mechanism makes the resolution limit much less restrictive for the map equation than for modularity; in practice, it is orders of magnitudes smaller.

Unlike other community detection algorithms, the graph used for map equation is a *directed* one. The weight of the edges is proportional to the number of times a source twitter user mentions destination twitter user in their tweets.

The algorithm detected the communities as shown in the figure below. The user @nickreeves9876 (Nick Reeves) is a very active member in twitter and follows 19k users and is followed by 32k users. He also regularly tweets about #Brexit and is strongly against Brexit. Since map equation uses page rank, it is easy to see why Nick Reeves is the most highlighted user in the figure below.

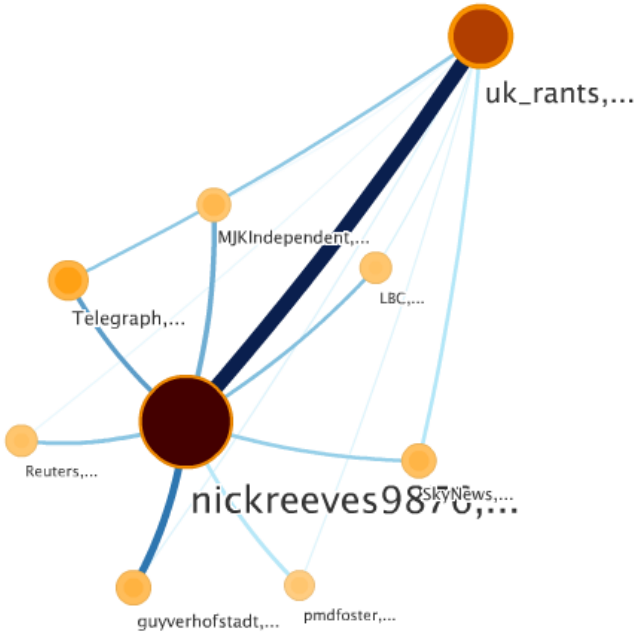

Figure 11 Community detection using map equation

Evaluation of community detection algorithms using sentiment analysis

We compared all the above community detection algorithms for our twitter network based on the sentiment polarity. First of all, we used the various algorithms to cluster the network into different communities depending on respective algorithms. Details could be found in the figures above. Then we computed the average of the sentiment polarity co-efficient of variation (CV) [26] within the communities, averaged over all the 4 weeks. We compared this value for all the 7 algorithms and concluded that the label propagation algorithm with the least variance works best for our network. The variance tells us that the communities formed by the algorithm have users whose sentiments are varying to that level. So, the algorithm having the least average variance, is the one which is able to detect the communities with like minded users, who have similarity in their opinions and hence least variance.

	mean_sentiment_CV
Kernighan-Lin	0.258871
Greedy Modularity	0.211898
k-Clique	0.255220
Label-Propagation	0.182775
Girvan-Newman	0.184610
Louvain Method	0.223064
Leiden Method	0.224992

Figure 12 Evaluation of communities using sentiments

If we check the average variance for each week, we find that the variance increases with time. We can infer from here that, with time, people kept changing their opinions about Brexit, based on more information gathered or shared within their community and neighbouring communities. Figure below shows the average result for each of the 4 weeks.

	mean_sentiment_CV_week1	mean_sentiment_CV_week2	mean_sentiment_CV_week3	mean_sentiment_CV_week4
Kernighan-Lin	0.225931	0.253910	0.253901	0.301744
Greedy Modularity	0.195532	0.209860	0.237113	0.205085
k-Clique	0.230284	0.248600	0.243407	0.298589
Label-Propagation	0.214525	0.200492	0.176703	0.139379
Girvan-Newman	0.201543	0.197264	0.198285	0.141350
Louvain Method	0.208941	0.222233	0.241756	0.219326
Leiden Method	0.202053	0.226074	0.242723	0.229120

Figure 13 Evaluation of communities using sentiments (weekly)

Interactive Network Visualization

In this section we discuss about the web application which we developed using D3.js library, and more precisely, the Force Layout algorithm, in order to present the users' network and their interactions between each other, in an interactive graph format. Each twitter user is represented by a circle/node and a line connects those users that mutually follow each other. Various functionalities have been included, such as presenting centrality measures for each user/node, clustering the users using the results of different community detection algorithms as calculated in previous steps and many others that will be presented later on, in this document. In general, this web application concerns an attempt to simulate other visualization tools, such as Gephi or Neo4j platform, and mainly to present our data in a more 'readable' format so that careful analysis on the nodes of interest can be performed.

As we just mentioned, all the data that are used in this visualization step, were acquired in previous steps, and were stored in json format, suitable for immediate use. We developed five .html files, having almost all of them the same functionalities, with some small differences between the FullNetwork.html and those .html files concerning the network visualization per week. With double-clicking on each .html file, someone can easily see the network visualization presented on the web browser, taking however

into consideration that internet connection is required also for that, because the necessary JavaScript libraries have not been stored topically, have not been downloaded in other words.

There are four main network plots, each plot representing the properties of the nodes for each of the four weeks of April such as number of tweets, retweets, sentiment etc. The visualization are done for undirected network (mutual following). Some of the key analysis that could be done from this visualization are:

- Play around with the network by moving each nodes to the required locations. Basically re-arrange the network in the way you want. This was not possible in the visualizations done in Python/R.
- Hover over each node to know the network statistics such as number of tweets, number of retweets, and centrality measures of the twitter user for the given week.
- Select a node and highlight only the connections of this node.
- See the sentiment of the nodes for each week of April.
- View the communities depending on the user-selected community detection algorithms.
- View the communities within the neighbourhood of the given node. See figure below.

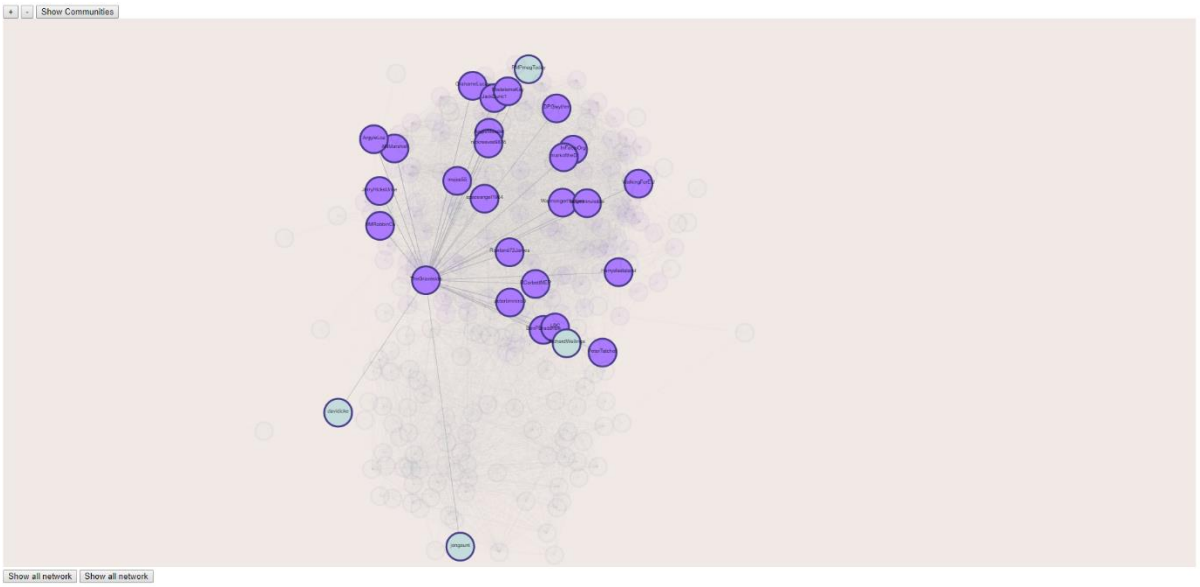

Figure 14 Finding all communities within the network of a selected user using D3.js

For detailed analysis using the D3.js application, the reader is encouraged to check the YouTube video by clicking this link.

List of figures

Tools & Software used

Tool/Library	Language	Purpose
tweepy	Python	- To collect the original tweets (retweets not included) of users who have tweeted about #Brexit in English language for the whole month of April, 2019. - To find the relationship (follower/following) among twitter users who have tweeted #Brexit.
textblob	Python	To perform sentiment classification on user's tweets.
pandas	Python	Data transformation.
numpy & sklearn	Python	To support additional data transformation & manipulation.
matplotlib	Python	For plotting & visualization
networkx	Python	Main library used for graphs. - For creating & plotting graphs - For calculating majority of the algorithms in graphs including centrality measures & community detections (except Louvain, Leiden, & map equation)
python-louvain (community)	Python	For community detection using Louvain method
leidenalg & igraph	Python	For community detection using Leiden method. Leidenalg library only supports the algorithms for graphs constructed using igraph library.
Map Equation's flash software	-	To run the map equation algorithm.
D3.js	JavaScript	The main Library used to visualize the whole network and present centrality measures in an interactive way
jQuery	JavaScript	For presenting some user's information
qTip2	JavaScript	For presenting some user's information
HTML5	HTML5 (SVG)	For developing our web application for visualizing the network
CSS	-	To configure the style of the visualization
JSON	JavaScript Object Notation	To store and transport data

References

1. D. Edler and M. Rosvall, The MapEquation software package, available online at http://www.mapequation.org
2. Günce Keziban Orman, Vincent Labatut, Hocine Cherifi. Comparative Evaluation of Community Detection Algorithms: A Topological Approach. Journal of Statistical Mechanics: Theory and Experiment, IOP Publishing, 2012, 2012 (08), pp.P08001. ff10.1088/1742-5468/2012/08/P08001ff. ffhal00710659f
3. Twitter network analysis: identifying influencers and innovators by Andrew Lamb.
4. Maximal Clique and K-Clique Analysis of Twitter Data Network by Sathiyakumari. K, Vijaya. MS, P S G R Krishnammal College for Women.
5. A Stable and Distributed Community Detection Algorithm Based on Maximal Cliques by Feng Gui ; Yunlong Ma ; Feng Zhang ; Min Liu ; Rong Yin ; Weiming Shen
6. K -Clique Community Detection in Social Networks Based on Formal Concept Analysis by Fei Hao ; Geyong Min ; Zheng Pei ; Doo-Soon Park ; Laurence T. Yang
7. https://networkx.github.io/documentation/stable/_modules/index.html
8. https://en.wikipedia.org/wiki/Clique_(graph_theory)#Definitions
9. https://neo4j.com/docs/graph-algorithms/current/algorithms/
10. Bedi, Punam & Sharma, Chhavi. (2016). Community detection in social networks. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 6. n/a-n/a. 10.1002/widm.1178.
11. Using the Leiden algorithm to find well-connected clusters in networks Vincent Traag, Ludo Waltman, Nees Jan van Eck
12. Tweepy (24 May, 2019). Retrieved from https://www.tweepy.org/
13. TextBlob: Simplified Text Processing (24 May, 2019). Retrieved from https://textblob.readthedocs.io/en/dev/
14. Natural Language Toolkit (24 May, 2019). Retrieved from https://www.nltk.org/
15. Pattern CLiPS (24 May, 2019). Retrieved from https://www.clips.uantwerpen.be/pattern
16. Natural Language Processing (24 May, 2019). Retrieved from https://en.wikipedia.org/wiki/Natural_language_processing
17. Brexit (24 May, 2019). Retrieved from https://en.wikipedia.org/wiki/Brexit
18. Scott Murray, «Interactive Data Visualization for the Web», O' REILLY, 2013, pages. 206-211.
19. Mike Dewar, «*Getting Started with D3*» , O' REILLY
20. D3.js, [Online] https://d3js.org/
21. Force Layout, [Online] https://d3-wiki.readthedocs.io/zh_CN/master/Force-Layout/
22. Force Layout, [Online] https://github.com/d3/d3-force/tree/v1.2.1
23. JQuery, [Online] http://learn.jquery.com/about/
24. QTip^2, [Online] http://qtip2.com/
25. Resolution limit in community detection (24 May, 2019). Retrieved from https://www.researchgate.net/publication/6609402_Resolution_limit_in_community_detection

26. Coefficient of variation (24 May, 2019). Retrieved from https://en.wikipedia.org/wiki/Coefficient_of_variation
27. Sentiment analysis (24 May, 2019). Retrieved from https://en.wikipedia.org/wiki/Sentiment_analysis