Tracking State Information Operations on Twitter

Motivation

Social media has become the battlefield for the hearts and minds of various population groups around the world. World superpowers have recognized the benefits of using social media to project their soft power efforts and dedicated a considerable amount of resources towards conducting information operations in the digital space. The notorious Russian operation conducted by the IRA (Internet Research Agency) in 2014 led to 3,841 Twitter accounts being shut down. However, the media channels owned by the Russian Federation (RT, Sputnik, Ruptly) continue to thrive on Twitter and have amassed a rather large Followership.

The team set out to explore current activity of the Russian media channels on Twitter. The intent of the project was to see if the platforms use any form of manipulation to achieve their objectives and identify topics of interest. The manipulation attempts would manifest themselves in a use of technology to automate activity. This type of activity can aid a variety of activities, including manipulation of public opinion by making a topic appear popular or inflating popularity of a particular account. Automated activity is relatively easy to detect through data analysis and with the limited experience on the team, this method seemed most achievable.

Data Sources

The two data sources used for the project were tweet objects, extracted from Twitter API in a form of json files and text scraped. The API Twitter resources are located here: https://developer.twitter.com/en/docs/basics/authentication/oauth-1-0a. Overall, the team collected 533 jsons (224 from RT network, 224 from Ruptly network, and 65 from Sputnik) containing 1,705,600 tweets. Overall, the team has collectively spent over 2 days' time on collecting data. Some of the tweets dated back to June 2016. In order to avoid an overwhelming amount of data and long API rate limit times, the team selected users that most frequently interacted with the accounts. Specifically, tweets of users who have interacted with RT and Ruptly at least 20 times and at least 10 times with Sputnik content (due to lower interaction rates).

Twitter Data

A tweet object can be extracted in a variety of ways, however, most commonly used due to the formatting is json (jsonl). The tweet object contains 31 fields of information, with some containing even more fields. One of the most important ones for the team were created_at (used for frequency analysis), id (unique id of the tweet), and entities (containing urls that were used to extract article text).

Limitations: During the collection process the team encountered a few issues. Twitter REST API calls have rate limits with some being more 'expensive' than others, where the wait time is

longer and retrieved information is less valuable. After the team collected the media outlets' tweets, the goal was to get interactions from the tweet id. Due to the rate limit being 100 retweeters every 15 minutes, some of the collection processes took over 10.5 hours. Additionally, the **100 retweeters per tweet** limit could potentially exclude some of the users, we would've otherwise focused on. The **REST API limit of 3200** tweets was not anticipated to be an issue for our data analysis as according to PEW research, the average Twitter user tweets twice a month and most prominent communicators tweet 138 times a day. However, during our visual exploration phase, it became clear that some users in our data set tweeted over 2000 times a day. This level of activity allowed us to collect that user's tweets going back a week or less.

Lastly, some of the retweeters that we were able to collect from the media channel's tweets, have been banned by Twitter. No data was available for those accounts.

Article Text
The second dataset used was a set of article text derived from the URLs contained in each media channel's tweets. The data was accessed through web scraping methods as shown in the URL_Data_Cleaning Jupyter Notebook. The input used was a CSV containing such information as the URL actually written in the tweet, extended URL, and IDs to refer back to each tweet. The data was loaded into a *pandas* DataFrame object. The most important variable used was the "extended_url" column.

Data Manipulation Methods

Tweets
The dataset containing tweets was processed by using the pandas library. The json's collected during the first cycle were read in Jupyter Notebook and tweet IDs were extracted to a list. The second collection phase resulted in a pandas dataframe containing all the user id's who have interacted with the tweets. Using the value_counts method, we were able to see who the main disseminators of tweets were. We used the user id's to make another call to the REST API and extract usernames for each id. A third round of collection consisted of getting the tweets for users interacting with the media accounts 20 times or more (except for Sputnik, 10 times or more). We were able to collect 530 jsons during this phase. During the next phase we had to extract all the mentions from tweets, in order to build our 2nd degree users. The step required a few steps of breaking down the test fields of the tweet objects for each user's json and concatenating the dataframes, while keeping the user's name associated with the mentions. Three dataframes, one for each media channel, were saved to a csv and used to visualize the networks.

Article Text
In preparation for our topic modeling, we extracted links from the entities fields of the media channels. This took a few steps to break down the multiple dictionaries contained in each item of the entities series. Additionally, due to the abundance of data, we chose to concentrate on

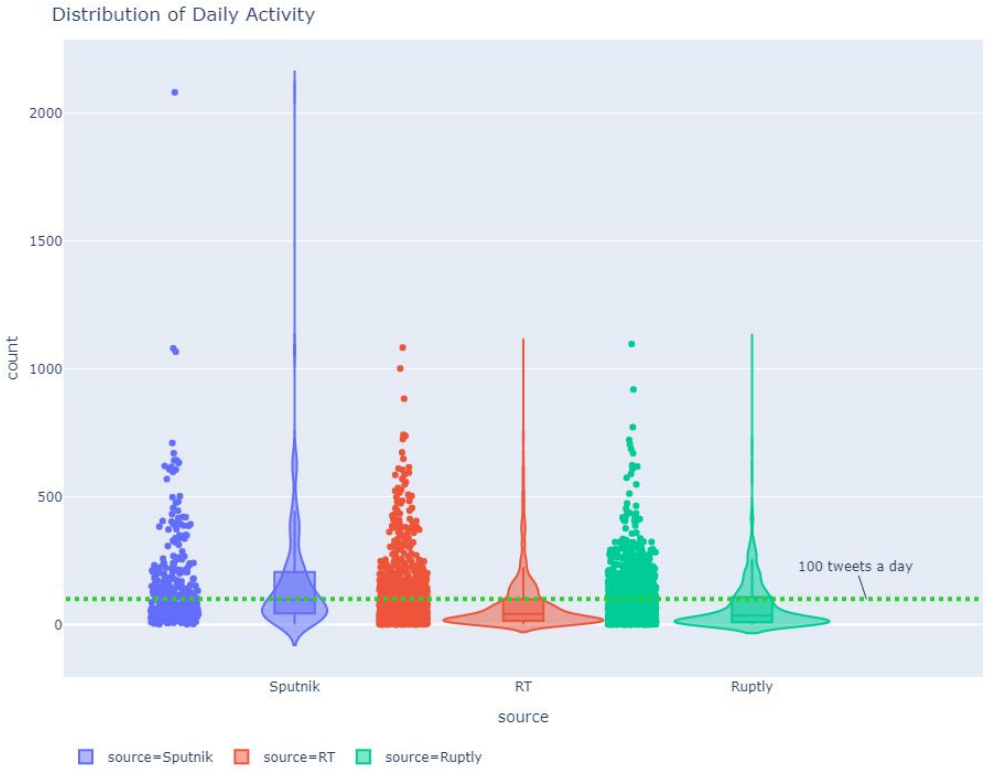
the articles that resonated the most, with 80 or more retweets. The process was performed by using the pandas library with sorting the data frame by setting a minimal value for the series. The output, csv files for the three accounts, served as the resource for topic modeling.

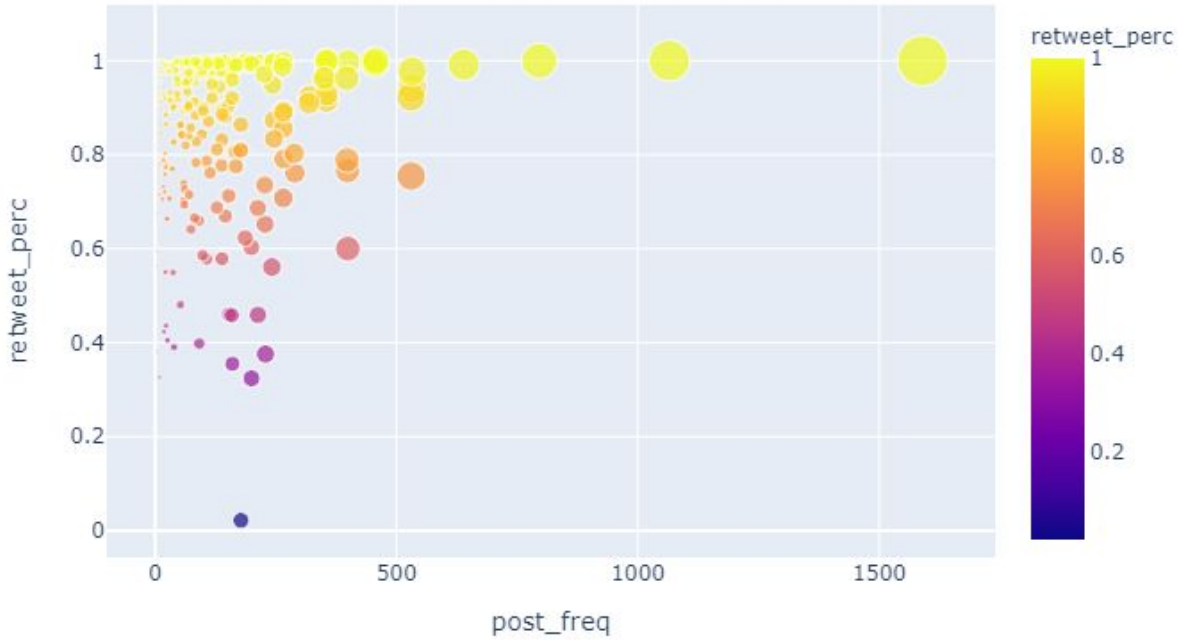
The web scraping procedure done to obtain the data needed before the topic modeling is found in the Jupyter Notebook entitled URL_Data_Cleaning. The actual method used was different from the planned method in that there were fewer steps performed in order to improve code efficiency. This paradoxically increased the time spent on coding it as plans had to be scrapped as new information came in, which was a challenge to reconcile. Another challenge was that the Ruptly url dataset was found to be unsuitable for the topic modeling analysis planned for it because they were primarily video links with the descriptions in the tweets, so simply looking at the tweet data would be effective enough in discerning the topics. This also added on to the time it took to develop the code.

The topic modeling process was done with heavy referencing to Alice Zhao's Natural Language Processing (NLP) in Python tutorial. The process involved the use of the *nltk*, *gensim*, and *scipy* libraries. For the first pass of Latent Dirichlet Allocation (LDA) model building, some topics were successfully discerned. However, we were unable to link topics with tweets. If we had more time, we would make a network visualization with topics for tweets labelled. An important takeaway is that proper NLP topic modeling takes time and multiple attempts to get meaningful output.

Analysis and Visualization

One of the key questions that the team set out to answer was whether the Russian-owned media channels were using automated activity. During the initial steps of data exploration, we visualized the retweeter's network collected using tweet ID. After visualising the daily activity of the top 15 users, sorted by interaction with the media content. We displayed the 100 tweet mark on the graph below to visualize the amount of activity associated with the Russian owned media channels. The mark serves as an indicator of high human level activity where anything above is most likely automated. A few insights came from the graph. Sputnik, displaying the most outliers, is most likely using more automation then its counterparts. Sputnik is also the only outlet with a median of daily user activity at 97.

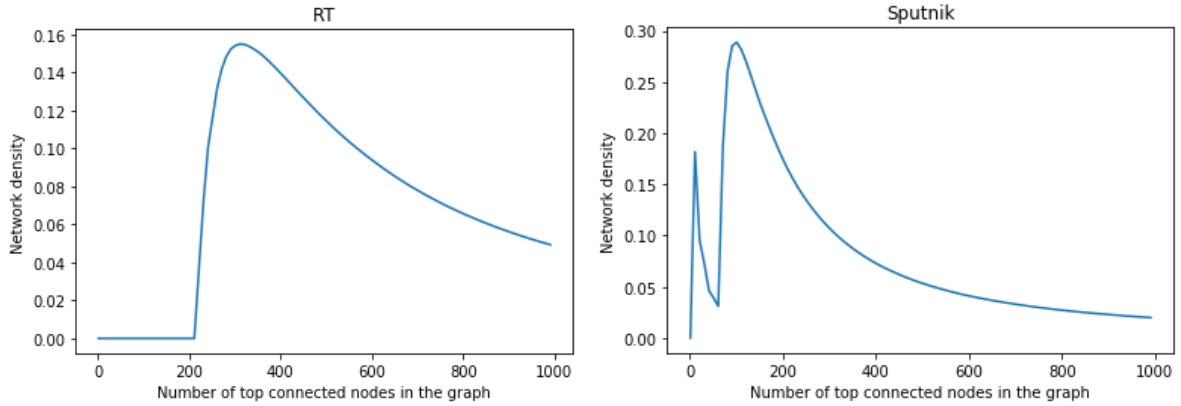

Distribution of Daily Activity
2000
1500
1000
500
0
count
100 tweets a day
Sputnik
RT
Ruptly
source
source=Sputnik
source=RT
source=Ruptly


retweet_perc
1
0.8
0.6
0.4
0.2
0
0
500
1000
1500
post_freq
retweet_perc

The percentage of the users' tweets being retweets was also a good indicator or automated activity. As seen in the graph above, a high percentage of RT's retweeters' activity is retweets. For example, the far left outlier has an average daily activity of over 1500 tweets with 100% of them being retweets.

While exploring the network data, an initial lead was found when plotting out the subnetworks' densities as the number of most connected nodes (those with the highest level of degree or connections) increased.

Each subnetwork was limited to the n most connected nodes, meaning each increase in n added nodes with fewer edges. This led to a predictable decline in density. However, they were also preceded by extremely low density. In the case of RT, the 200 most connected nodes did not have a single edge between them. Sputnik had a wave, and Ruptly was a mixture of the two.



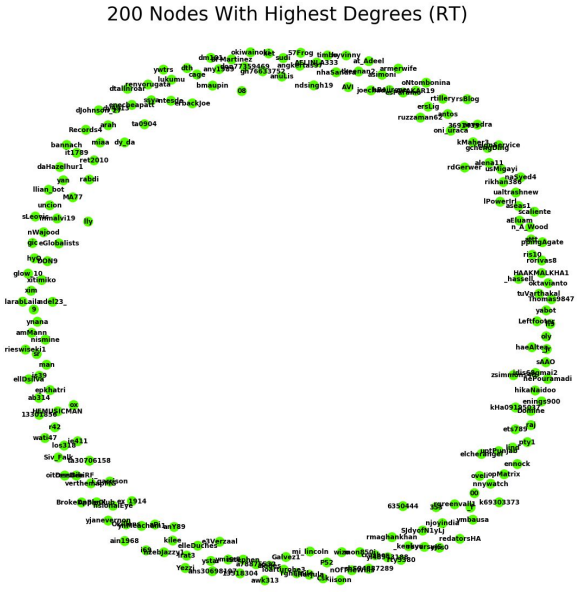
The RT and Sputnik line plots

While a social media network is not entirely identical to a regular social network, it was still curious that the 200 most popular nodes had no contact with each other. Sputnik had a similar phenomena, though with only 16 such nodes.

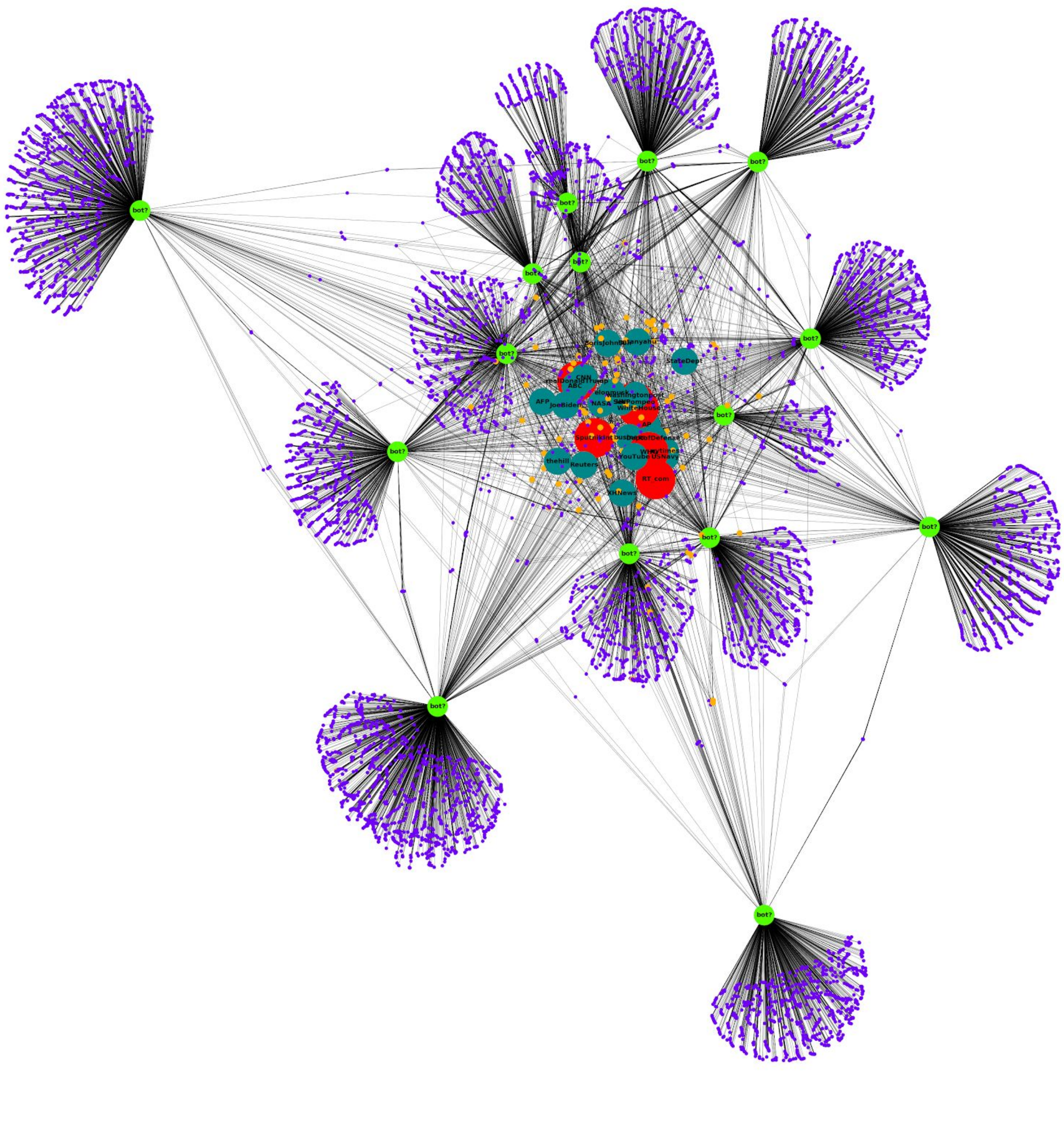
From there, the next successful turn was in plotting out a subgraph of the most connected nodes, along with all of their neighbors or direct nodes. The resulting plots were very insightful in 2 aspects. Firstly, each suspected bot node had a sort of "purple bloom"; a hoard of nodes to which it was connected, but none of which were connected to any other node, much less each other. At first, we thought that this was merely a reflection of the data collection methodology, with suspected bot nodes simply being the first layer connected, and their blooms being the second and final layer. However, upon closer inspection, none of the suspected bots had been scraped in the first layer. The fact that these blooms had densities of 0 helped support the theory that the suspected bots were in fact bots (one would expect that at least some of an account's connections would know each other). It also shed light on how the bots operated - by

hitting up huge amounts of disconnected users/accounts in order to spread or support ideas, views, or opinions.

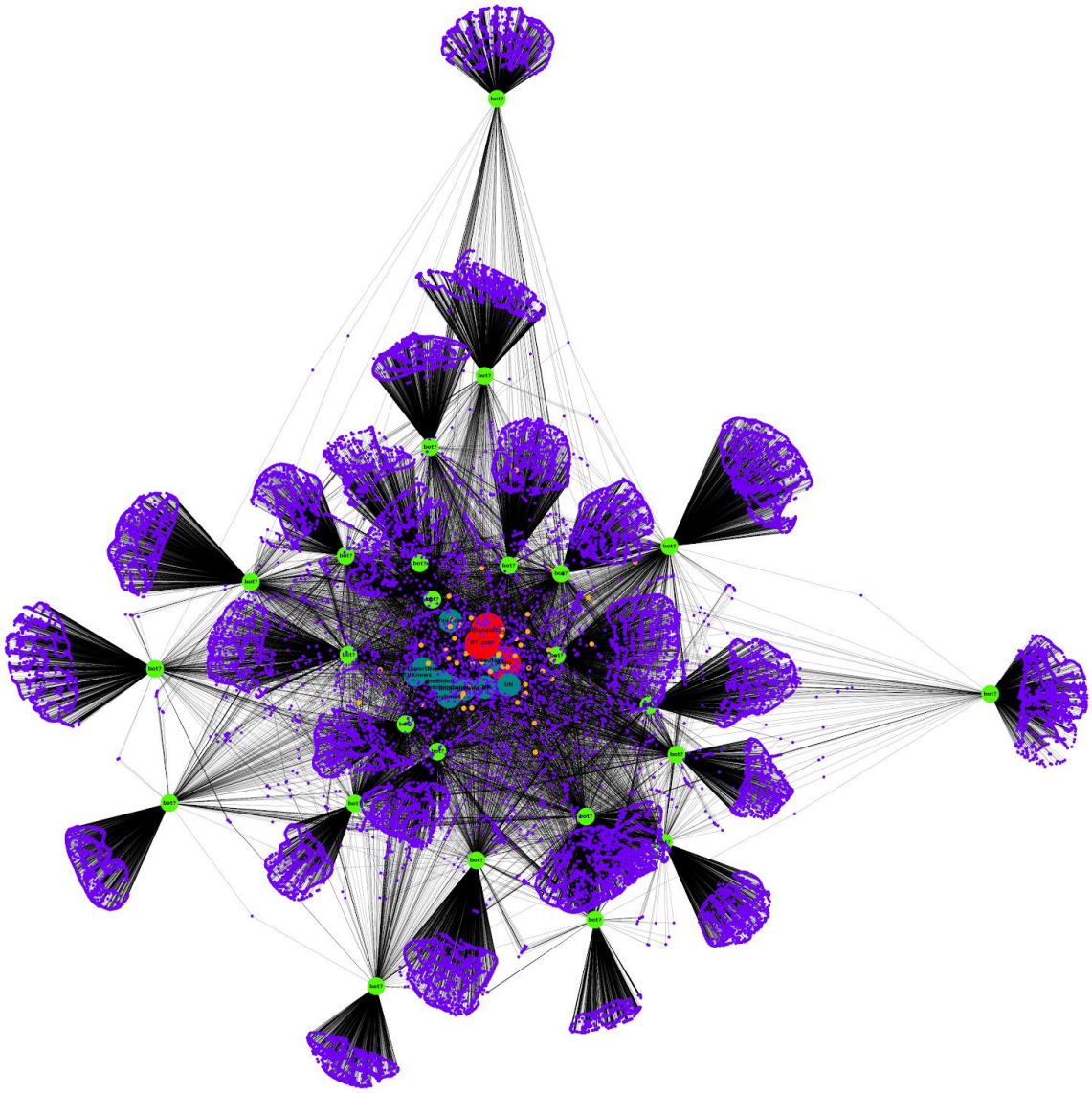
Secondly, at the plots' center major players or channels were found. For RT, those included: realDonaldTrump, JoeBiden, the New York Times, CNN, the WHO, and the UN. For Sputnik, the players included much of the same, as well as others such as: NASA, the US Navy, StateDept, and Elon Musk. Because the plot was a springplot, these nodes were found in the center due to their high level of connectivity from the suspected bot nodes. It can be theorized that these central nodes act as threads or forums, where the bots can interact and spread ideas to multiple target audiences, potentially without directly communicating with them, rather than the central nodes being actual targets themselves. While these forum nodes do not indicate specific ideas being shared, they do help to illustrate areas of importance.



Sputnik’s subgraph is shown below, made up of the 16 most connected nodes (that weren’t connected to each other) and all of their neighboring nodes. The lime green circles represent suspected bots; their color and size exist outside of the size scale. The remaining dots scale in size as they increase in degree of connectivity. At the center can be seen the “forum” nodes while on the extremities individual and disconnected accounts.



The same plot for RT with a sample of 25 suspected bot nodes (due to computational limitations) is shown below. An identical structure can be seen.



Statement of Work

Viktor Avdulov retrieved and processed data from the Twitter API and prepared the data for visualization and topic modeling. Additionally, Viktor combined the data sources and visualized the overlap of the data sets.

Tyvand McKee performed the exploratory network analysis on the data extracted by Viktor. He created the network visualizations.

Monica Yen used web scraping to retrieve article text from urls RT and Sputnik tweeted. She cleaned and organized the data and performed the topic modeling procedures.

Reference

Zhao, Alice. “adashofdata/nlp-in-python-tutorial: comparing stand up comedians using natural language processing.” Github, n.d., **https://github.com/adashofdata/nlp-in-python-tutorial/.**