

Polarizing Activity of Russian Twitter Accounts Leading up to the United States 2016 Presidential Election

Sarah Pardo '22 and Panya Tang '22
CS230 Fall 2019 Final Project
Professor Takis Metakas

1. Background

The 2016 election for president of the United States between Donald Trump and Hilary Clinton was one of the most heated and polarizing in modern US history. Hillary Clinton, the Democratic nominee, who previously held the position of Secretary of State and was First Lady to Bill Clinton, was up against Donald Trump, an ambitious businessman who entered the race without any political experience. The stark difference in ideology between the two candidates created a divisive political environment, but tensions were further exacerbated by difficult international relations with Russia.

The United States and Russia have had political tensions for many decades. Vladimir Putin, the current president of Russia, holds anti-Western values that previous presidents have repudiated. Thus, when Putin spoke favorably of Trump, suspicions arose about potential conflicts of interest. The Russian government was politically motivated to support Trump as they believed it would destabilize the political institutions of the United States and give them an upper hand in diplomatic issues between the two countries.

After the election, the Special Counsel Investigation led by FBI director Robert Mueller found that the Russians had interfered with the elections. Firstly, by meddling with the results in voting and secondly, by spreading fake news to benefit Trump and disarm Clinton. The Russians bought targeted ads on Facebook and Instagram that attacked Clinton's campaign. The Internet Research Agency based in Saint Petersburg created thousands of fraudulent social media accounts, supporting Trump and opposing Clinton, which reached millions of people between 2013 and 2017. In addition, news articles circulated and spread false news. The propagation of fake news occurred mainly through the web, where unsuspecting readers would read an article written by a believable but fake newspaper company and take it as truth.

On Twitter, a social media platform where users can share updates and information in 140 characters or less, Russian news bots circulated fake stories or biased news amongst themselves and other real users who shared the same political stance. A simple "retweet", or share, leads to greater exposure of the tweet, generating activity surrounding the news. We investigated the activity of these fake Twitter accounts, examining the interconnectivity between the bots and the stories they (re)tweeted using data from the program TwitterTrails, "an interactive, Web-based investigative tool that allows its users to examine on Twitter the origin and propagation characteristics of a rumor and its refutation" (Metaxas and Finn, 2019).

2. Introduction

The problem we sought to understand is the interconnectivity between the false stories and the Russian Accounts on Twitter (RATs) that share them. By examining this network, we can find valuable information on the behavior and patterns of these accounts.

The "All_Russian-Accounts-in-TT-stories.csv.tsv" file contains information on the name of each Russian bot, its associated numerical user ID used by twitter, its tweet count, story count,

and finally the numerical IDs of the stories it has been associated with. Given this information, we wanted to create nodes for each Twitter account and story and analyse which of these nodes were most active or perpetuated most within the system. To do this, we had to create edges, representing a connection, between the accounts and the respective stories they participated in, resulting in a bipartite graph. Bipartite graphs separate vertices into two groups. In this case, there are edges between users and stories, but never between stories and stories or users and users. The program yED was used to generate different visualizations of our graph. Once created, our bipartite graph was used to answer the following research questions:

1. Who was the most active RAT
2. Which was the most popular story among RATs
3. What is the size of the largest connected component (LCC)
4. What was the most central node in the LCC?
5. Was there a trend in the profiles of the most popular users?
6. Was there a shared message amongst the most popular stories?

3. Methods

We created four Java Classes to investigate RAT activity. First, our `AdjListsGraph<T>` implements the `Graph<T>` interface, providing users with two vectors for vertices and edges, the latter representing edges through a linked list containing all of the vertices to which each vertex is connected. This class also provides methods to run breadth and depth first search from any node on the graph, returning a linked list representation of the traversal. Another method allows the user to save the graph to a TGF file, whose format can be converted to a visually accessible graph in yED.

In order to represent nodes that were distinguishable as users or stories in our bipartite graph, we created a class `RATvertex` with two constructors, one general and one specialized for the additional information specific to user nodes. The former has parameters for a `String` representation of the user/story ID and a boolean to indicate whether or not the node was a user. Another `toString` method was created specifically for user nodes to provide the additional information. We also implemented the `Comparable` interface to compare the respective edge counts (the number of connections to other vertices) of two `RATvertex` instances. Though the story count for users should be equal to the edge count, we still used the edge count variable in case there were discrepancies in the file and for unity between user and story vertices.

Once we had created the `RATvertex` class, we were able to construct a `RATgraph` class whose constructor has an instance `AdjListGraph<RATvertex>` variable called `bipartite` as well as counter variables for the total user and story vertices.

The RATgraph class contains a method called `readFile`, that takes in a text file with the required information to produce the bipartite graph. Within the text file, the first line is the header for the information on the RAT graphs: *screen_name*, *user_id*, *tweet_count*, *story_count*, and *stories_comma_separated*. The information associated with each row is separated by tabs, and the stories for each RAT are separated by commas, which determined what methods we used to process the data. Our objective was to read through the lines of the file and create user and story RAT vertex instances. To ensure a file existed for reading, we used try/catch statements to handle IO and `InputMismatchException` exceptions.

The `readFile` method contains two Scanner objects. The first reads the entire file. Then, using a while loop, the second Scanner reads one line at a time. Within each line, the second scanner processes the information under each column head and declares local variables to store it. These variables are passed as arguments into the `RATvertex` constructor. The `totalUser` count is then updated and the user vertex is placed into a Hashtable holder (explained in the following paragraph). The remaining information on each line is the comma separated story ID values. `readFile` stores all of these in an array and creates new `RATvertex` objects for any “unseen” vertices. The method creates edges between the stories and the aforementioned user vertex.

One obstacle we faced was to ensure that there were no repeat nodes of stories or users. We used a Hashtable to check and store nodes once they have been added to the bipartite graph. A Hashtable is more efficient to search through for repeat nodes than checking the entire vertices vector for every story on every line. Our Hashtable had String keys and Integer values. Whenever a new story or user ID was processed by the Scanner, we checked, using the `containsKey()` method, if the ID was already inside the Hashtable. If it was, we simply added an edge between the current user and the existing story. If it was nonexistent, we created a new story `RATvertex`, created an edge between the current user and the story, and then added the story ID to the Hashtable.

In the case that the ID already exists in the holder, we had to find the correct vertex to which to draw an edge. To accomplish this, we created a helper method called `findVertexFromId()`, whose parameter was a string story ID. The method loops through the vertices in the bipartite graph to find a vertex with a matching ID. By this method, the vertices vector will only be looped through in the case that the vertex certainly exists, rather than every time.

To help us find which users should be investigated qualitatively, we created a method with the `RATgraph` class called `mostActiveUsers`, which returns a list of the top ten most active users. We defined “active” as participating in the greatest number of stories, since those users were perpetuating the greatest amount of information on Twitter. This method utilizes `HeapSort` and its `sortInDescending` method. A for loop is used to store only the user vertices within a vector. This vector is then passed through the `sortInDescending` method and a sorted vector is produced. Then a sublist of the first ten users is returned. A second method, `mostPopularStories`, works exactly the same way except that story vertices are stored instead of user vertices.

Another of our research objectives was to find the largest connected component in the graph. One way to answer this question is to find the two nodes the furthest apart; the number of steps between them would indicate the number of vertices encountered along the way and thus the size of that component. We created a method called `furthestApart()` within the `RATgraph` class which runs a breadth first search call from every node in the vertices vector within the bipartite graph. Since the breadth first search returns a `LinkedList`, we can count the number of steps between the first and last nodes using the size of the list. The method stores the highest number of steps in a local variable and updates it continually, returning a summary of the greatest size found and the first and last nodes in that `LinkedList`.

Once we collected our data using the `mostActiveUsers`, `mostPopularStories`, `furthestApart` methods, we began a qualitative investigation using the `TwitterTrails` and `Twitter` websites. We read the `TwitterTrail` investigations associated with the most popular stories and found the available user profiles of the most active users.

4. Code

This section will present our code on the proceeding pages for the classes we programmed in the following order:

1. `AdjListsGraph`
2. `RATvertex`
3. `RATgraph`
4. `Investigate`

We used `BlueJ`, an integrated development environment for Java, to write and test our code.

5. Conclusions

After running our codes and creating variations of our bipartite graph, we were able to answer the research questions. There were a total of 896 nodes, 287 of which were users, and 609 of which were stories. There were 7160 edges created between the users and stories. We found that the largest connected component of the graph was also 896 nodes, signifying that every user was connected through a story and vice versa. The two nodes farthest apart were `User Jenn_Abrams` (id: 2882331822) and `User RUS_IN_USA` (id: 762948178178150400), and they

were 896 steps apart, indicating the entire graph was one connected component; every user was connected to another through a story and vice versa.

User Jenn_Abrams was also the most active RAT, participating in 144 stories. By studying the graph visualization, it is apparent this user is also the most central node, as it appears in the very center of the several layouts including Balloon Tree and Radial (Figure 1). User RUS_IN_USA, on the other hand, appears on the periphery of the visualization.

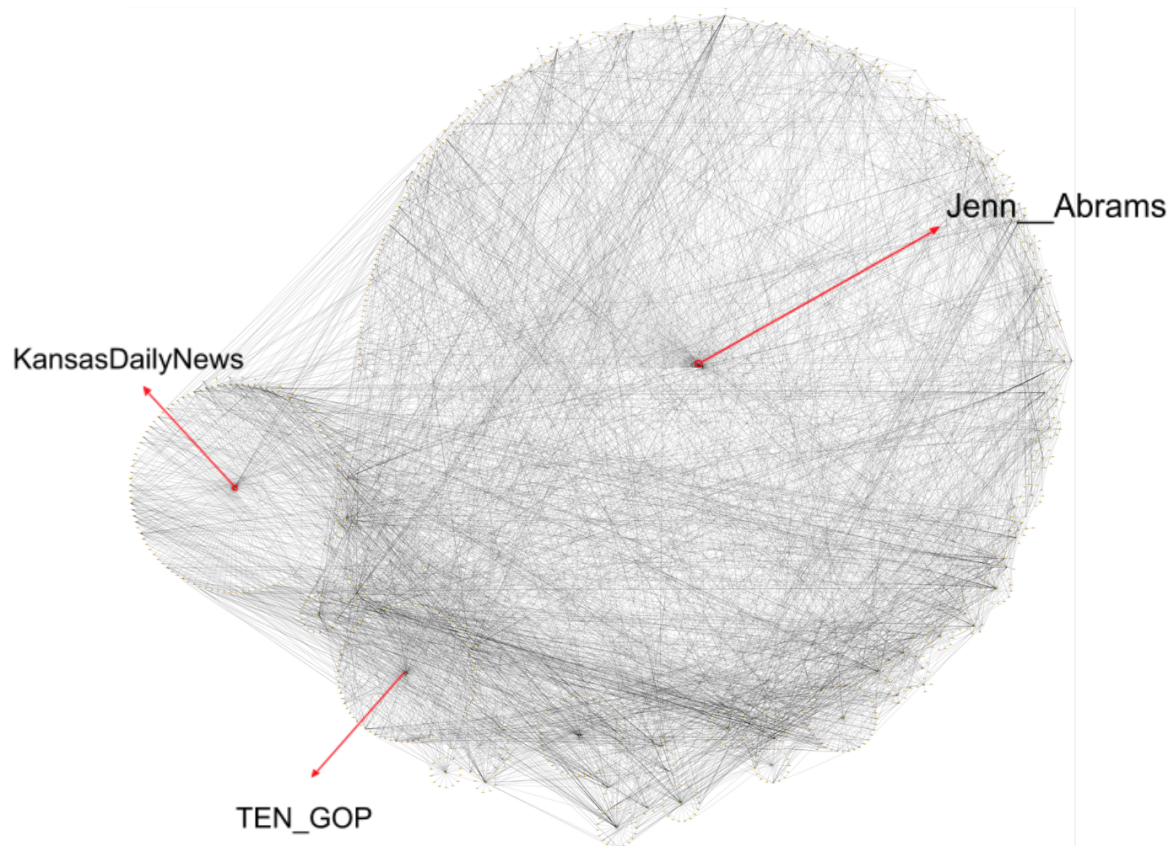


Figure 1. yED Tree (Balloon) Graph Visualization Depicts the nodes of the graph in “tree” formation with adjacent nodes circled around central nodes; the top three most active accounts were at the center of the three biggest circles, indicating those accounts are central nodes.


Table 1. Top 10 Most Active RAT users

	Screen Name	ID	Tweet Count	Story Count	Twitter Profile
1	Jenn_Abrams	2882331822	248	144	suspended
2	KansasDailyNews	2587843805	165	114	suspended
3	TEN_GOP	4224729994	519	101	suspended

4	TodayNYCity	2752677905	151	93	suspended
5	NewOrleansON	2530830345	169	92	suspended
6	DailySanFran	2495567768	99	77	DNE
7	WashingtOnline	2743327187	95	70	suspended
8	PigeonToday	2912754262	91	68	suspended
9	AmelieBaldwin	1679279490	158	64	DNE
10	ChicagoDailyNew	2547141851	75	63	suspended

We found that the top ten most active accounts are currently suspended or no longer exist (Table 1). Even without more account information, however, we are able to see a trend in the screen names. The majority, with the exceptions of Jenn_Abrams and AmelieBaldwin, have names that closely resemble a newspaper or at least a company, rather than an individual; five of the top ten accounts have the word “today” or “daily”. We hypothesize that handles mimicking newspaper names may face less scrutiny. Given the appeal of Twitter to quickly deliver information, it may be necessary for the accounts to resemble newspapers in order not to arouse suspicion.

Though none of the accounts are currently accessible, we have access to previously saved account pages for two of the accounts: Jenn__Abrams and AmelieBaldwin. These accounts resemble people rather than newspapers, and have convincingly realistic profiles. The most active account, Jenn_Abrams, could pass as a woman who is passionate about right-wing politics. The profile biography indicates the user is receptive to “any offers/ideas/questions” and includes an email, which increases the supposed veritability.



Jenna Abrams
@Jenn_Abrams

Calm down, I'm not pro-Trump. I am pro-common sense. Any offers/ideas/questions? DM or email me jennnabrams@gmail.com (Yes, there are 3 Ns)

Location: USA
Language: English

Created at 12:57 PM - 29 Oct 2014 - Age 1864 days

24,744
TWEETS

22,874
FOLLOWING

70,083
FOLLOWERS



Amelie Baldwin
@AmelieBaldwin

Wife, Mother, Patriot, Friend

Location: USA
Language: English

Created at 6:53 PM - 17 Aug 2013 - Age 2303 days

39,157 TWEETS	2,285 FOLLOWING	2,849 FOLLOWERS
-------------------------	---------------------------	---------------------------

Another active account, Amelie Baldwin, has a believable picture of a woman and biography. Notably, Jenn__Abrams has about 25 times more followers than this account, which is the 9th most popular. We are not able to see the content of every story tweeted by each user, but presumably the tweets shared will have an impact on the account's popularity in addition to the account profiles themselves.

Some very important, but overlooked, contributing factors to making these accounts believable were the age of the account and the location they were created in. Political statements coming from perceived foreigners may reduce the credibility of the account as some users may feel like a non-American does not have a say in what goes on in the United States. In addition, having an account that is relatively old helps create the facade that there is a real person behind the account and actively using it, instead of a bot created for a specific purpose.

Table 2. Top 10 Most Popular Stories

	Story ID	Edges	Story Description (from TwitterTrails)	Skepticism
1	190816579	53	Mistrial in the first trial of one of the officers accused of killing Freddie Gray	Undisputed
2	781754897	48	Invalid ID	N/A
3	9871042182	46	Trump - 5/20/2016	Undisputed
4	8281346356	38	Susan Rice	Hesitant

5	8231029693	34	You do not have permission to view this story.	N/A
6	4881270967	34	Jeff Sessions Confirmation Hearing	Undisputed
7	6781268418	29	White Genocide	Hesitant
8	9911201671	28	#ifthemediariggedtheelection	Undisputed
9	4251092648	27	You do not have permission to view this story.	N/A
10	3911263100	27	Pizzagate-2016.12.08	Dubious

The descriptions for the top ten Twitter stories shared amongst the RAT accounts aim to provoke Trump supporters. For example, the stories 9871042182, and 4881270967 pertained to American politicians from the Republican party while stories 190816579 and 6781268418 were written on topics that would particularly rile up white voters. Of the top ten stories, only seven were accessible through Twitter. Five of the seven were either categorized as undisputed or hesitant for their level of skepticism. We believe fake accounts may have shared stories that were more factual because a true story would propagate for a longer time than a fake story, which would more likely be debunked. If the goal of the fake accounts were to reach as many conservative and moderate voters as possible and influence their political opinions, using specific stories that have bases in truth would make the long term goal of swinging the votes towards the Republican party more achievable.

The most popular story was 190816579, which had edges to 53 different users. The webpage <http://twittertrails.wellesley.edu/~trails/stories/title.php?id=190816579> has one line of text that reads “Mistrial in the first trial of one of the officers accused of killing Freddie Gray”. This story refers to the event where Freddie Gray, a 25-year-old black man from Baltimore, Maryland, sustained a spinal injury at the hands of the police, which subsequently led him to his death on April 19, 2015. Six police officers were brought to court, the first being Officer William G. Porter. Charges against him were eventually dropped because of mistrial. A story that focuses on the mistrials would potentially be divisive amongst Americans with differing views on police authority, police brutality, the pervasiveness of anti-black racism in the United States, and gun control.

6. Collaboration

In the beginning stages, we talked and planned out our strategy for solving our project problems. We began by using Sarah’s AdjListsGraph class from lab that contained many of the necessary methods already and implemented the rest of them, such as depth first search and breadth first

search, as outlined in the Graph<T> interface. Since we were struggling to get our DFS method to work, we went to Annabel's SI hours and Stella's office hours to receive help.

In addition to the AdjListsGraph class, we worked on writing the code for the RATvertex class, which helped distinguish whether a vertex was a story or user. We decided to write this class so that we could adapt our graph to a customizable vertex object, in this case a Twitter story or Twitter account. We discussed what instance variables should be present together and wrote the getter methods and constructor together. Sarah wrote a user vertex oriented toString method and a compareTo method.

For the RATgraph class, we wrote the beginning portion of the code together. Over Thanksgiving break, Sarah debugged the code we had already done and continued coding the program, while also communicating with the instructors and Panya regarding the problems we were facing. She also decided to implement a hashtable to help keep track of the vertices within the readFile method. Panya began writing the background section and reading up on the necessary sources of information to help contextualize our research paper.

After Thanksgiving, both of us went to office hours to obtain more help with debugging our readFile method in the RATgraph class. Sarah began writing up the outline for our methods section and Panya created a separate introductory section to help clarify the purpose of our paper. Together we used the outline to help us complete this portion.

In order to help find the top 10 most active users and popular stories to help our conclusions section, Sarah used HeapSort and emailed Stella to figure out how to implement multiple interfaces. Sarah tabulated the information in graphs and we used this to analyze the results by looking at patterns amongst the stories and users. Panya did research on the stories and drew conclusions between the level of propagation and the amount of skepticism associated with each of the stories. We also discussed together the best visualizations to represent the information, but were unable to render an Orthogonal layout due to a lack of computer memory.