Fake News on Twitter

Lukas Lindqvist, Viktor Wällstedt
Email: {lukli841,vikwa174}@student.liu.se
Supervisor: Niklas Carlsson, {niklas.carlsson@liu.se}
Project Report for Information Security Course
Linköpings universitet, Sweden

***Abstract*—Social media today is filled with misinformation also known as "fake news". One of the more popular social media platforms which today is experiencing this problem is Twitter. The aim of this project and paper were to come up with a method of telling whether a tweet on Twitter was to be considered as fake news or not based on information gathered from the Twitter API. This proved to be a difficult task and in the end the project instead resulted in a tool that can be used to inspect a tweet and its retweeters by visualizing their characteristics. This tool was created in the form of a web application with front-end written in React and back-end in Node. The flow of the tool could be described as the front-end first receiving a targeted tweet from the user. This targeted tweet is then sent to the back-end which by being connected to the Twitter API fetches the data related to the targeted tweet. Finally the front-end is able to visualize this data. Thereafter the user is potentially able to manually identify characteristics of the data and based on this try to guess whether the tweet is to be considered fake news or not.**

I. Introduction

News in society today are no longer retrieved only through the traditional channels such as TV and papers. Most people today read the news on the Internet and often on social media sites. The trouble with social media sites compared to traditional established publishers is that they do not have an audit process of what people post on their sites. Therefore there is a serious problem with people posting fake facts, facts which people often believe to be true [1]. In a report done by researchers at MIT it was stated that fake news are more likely to be shared by people than regular news and therefore spreads further [2]. One of the goals with this paper is to inspect the retweeters of a specific tweet to get a better understanding of who spreads tweets and where they are located. This could be interesting in many ways, for example if someone in America have a majority of retweeters located outside of America, one might wonder why? Furthermore we will try to figure out how general data retrieved from Twitters API can be used to determine whether a tweet is to be considered as fake news or not.

To give an overview of this report a short presentation of the report sections will follow. To identify the background of the problem and how it affects the society the report begins with a background chapter. In the next chapter we discuss the methods used in this project. After the method a chapter about results and analysis will follow where the solution and its results are analyzed. The next section is a discussion of the project and what could be done differently. Thereafter, inside the related works chapter, the report compares its own results and analysis to other similar works. Finally, the report is finished with a conclusion of the methods used, results and a part about potential future work.

II. Background

As mentioned earlier in the introduction, there is a shift going on in society today of media platforms using local physical medias to digital worldwide medias. People today read their news on smart devices, and even thought the traditional media houses are adapting to this shift, people nowadays gets pumped with information through their social media usage. Traditional media houses have strict laws and policies regarding what is allowed to be written and published, whereas social media sites are a brand new arena where there are not really any limitations of what users can portrait as the truth. A part of the problem is that the social media sites currently do not take any responsibility for what is being published on their sites. As of late the world has begun to take notice and now the social media site big shots have been put into the spotlight and questioned about their way of dealing with what is published onto their sites.

One of the major reasons that the world is beginning to realize the scale of this problem is the fuzz about foreign countries using social media to affect democratic elections [3]. Recently the Facebook founder Mark Zuckerberg was summoned to the U.S. (United States) congress to testify on the subject [4]. This problem is not in any way isolated to the U.S., it is a global problem. Another example is The Swedish Civil Contingencies Agency which made a statement last year clearly warning about the possibility of fake news that would try to affect voters prior to the democratic election taking place in Sweden in 2018 [5].

Facebook is not the only social media site with this problem, and in this report we are going to focus on Twitter. If we take a look at Twitter and the aspect of fake news, there is a well known president that is not afraid to make use of Twitter to propagate his thoughts on different matters. The website PolitiFact is specialized in taking news published on the Internet and making a fact-check review out of them [6]. Politifact has made an interesting scorecard of Donald Trumps' Twitter credibility shown in figure 1 below. This picture clearly indicates a disturbing amount of fake news being published by a person with high influence in the global society. He is spreading this information by using the social media site Twitter.

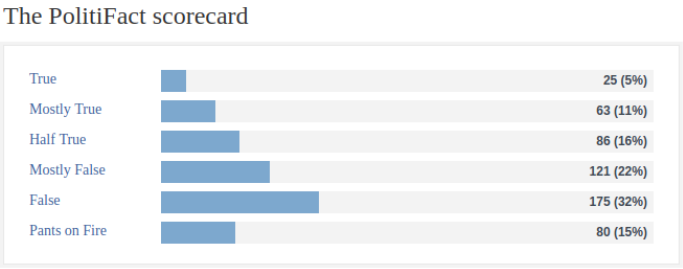

Fig. 1. Donald Trumps' Twitter credibility.
Source: http://www.politifact.com/personalities/donald-trump/

III. Method

A. Theoretical Methods

Before we started coding we sat down and thought about what kind of data from the Twitter API that could potentially be interesting to us when trying to detect whether a tweet is fake news or not. Of course information about the tweet itself is required, such as the text, origin, author, author followers/friends and statuses. After some consideration we also thought about Russia's involvement in the U.S. election, maybe it could be interesting to see the origin of retweets? For example, if a tweet is about politics in the U.S. and lots of retweets are located in Russia, maybe that could be a sign that this tweet is a Russian attempt to manipulate the voters in the U.S.

Another characteristic that crossed our minds as interesting is the amount of followers/friends and statuses retweeters have. To increase the retweet/favorite count people could create fake accounts just to retweet/favorite fake news tweets, without making friends or creating statuses themselves. Another scenario could be that an account is used for only creating manipulative statuses and then make comments on high profile tweets to lure people to their own manipulative statuses. In this case the profile could have zero followers, zero friends and a lot of statuses.

B. Limitations

There are certain limitations to the Twitter API, especially when used with the free-tier. These problems are mainly rate limitation, and no access to historical data. For Twitter, historical data is anything but the latest 100 of something, such as retweets. Because of this the back-end had to be written in a way so that it could collect data over a certain period of time without user interaction to give us as much data as possible to analyze. To solve this we implemented a functionality called subscribe which the front-end can use to tell the back-end to subscribe to a certain Twitter user. After a subscription call has been made the back-end will collect data over a 3 hour period on that users next tweet. Read more about how this is done below.

C. Practical Methods

Note: Error handling inside code snippets will be omitted for brevity and readability.

1) Back-end: To be able to make API calls to Twitter we decided to use an available open-source package (called twitter) written in JavaScript [7]. API-keys to Twitter are required to be allowed to make API-calls, these keys can be created on Twitters application page [8]. With these two together we created a local client that can easily make API calls to Twitter, see listing 1. As seen in the code snippet, it is good practice to use environment variables for API-keys.

```
1 const client = new Twitter({
2   consumer_key: process.env.TWITTER_CONSUMER_KEY
        ,
3   consumer_secret: process.env.
        TWITTER_CONSUMER_SECRET,
4   access_token_key: process.env.
        TWITTER_ACCESS_TOKEN,
5   access_token_secret: process.env.
        TWITTER_ACCESS_SECRET
6 });
```

Listing 1. Setup Twitter API connection.

To use the client (shown in the code snippet above) for collecting data about a specific tweet, we call the Twitter API as pictured in listing 2. This request will return information about who wrote the tweet and information about the tweet itself.

```
1 export const getTweet = async id => {
2   const tweet = await client.get("statuses/show"
        , { id });
3   return tweet;
4 };
```

Listing 2. Get tweet information from Twitter API.

We can apply the same method as shown in listing 2 to make use of all available endpoints on the Twitter API. A similar method is used to collect the most recent hundred retweets of a tweet. Because of limitations to the free plan of Twitter API it is only possible to retrieve the hundred most recent ones. To combat this we created our own subscription algorithm. It is started by passing a Twitter screen-name to our back-end which will check the users last tweet. Then it will check back every 5 seconds to see when the user tweets again. When this happens it starts to collect the hundred latest retweets every 10 seconds. This goes on for 10 hours. By doing this we can retrieve much more data than just the hundred most recent ones.

```
1 const collectRetweets = async id => {
2   getRetweets(active_subscription.latest_tweet);
3   let subscription = setInterval(latestRetweets,
        1000 * 3);
4   active_subscription['subscription'] =
        subscription;
5 };
```

Listing 3. Subscription of a tweet.

2) Front-end: We decided to write the front-end of the application in React using the state managing library Redux to keep track of the data inside the application. The underlying reason behind choosing React is that it has a big community which grants access to many third-party packages and there are much help available from the community. To visualize the Twitter data we used one of these third-party packages called react-chartjs. The way React works is to divide the rendering and visualization generated by the code into components or

containers which then are responsible for showing content. The difference between a component and a container is that a component is hard coded to only return some JSX (HTML) whereas a container is connected to the Redux-store and able to read in data from the store and also call actions to manipulate the state (data). We decided to make one container for the search bar and one container for listing the Twitter data. To handle the data we have available inside of our front-end we use Redux which makes use of so called actions (requests to API:s) and reducers (updates our data storage depending on the result of an action). We created one action for fetching information about a tweet given the tweet id, and another action for fetching retweet information related to that same tweet. To be able to make use of the data returned by the actions we also created two reducers which makes sure the data returned by the back-end is saved inside of our front-end. At a later stage we also added another action used to initiate a subscription at the back-end. This action does not change the state of our front-end in any way and is purely used to gather more data that can be used later.

Here follows a small summary of how the interaction with users were supposed to work: a user is presented with an input field which the user can use to target a tweet id. The tweet id can be found at the end of the URL inside a users browser after selecting a tweet on Twitters website. After supplying our application with a tweet id, the tweet id is sent to our back-end where we fetch data from the Twitter API which is then returned to the front-end. This data is then processed to give us the parts we are interested in and finally visualized with the help of react-chartjs. Later on we also added the subscription alternative which allows the user to supply a Twitter user screen name and check the "subscription" alternative next to the input field to initiate a subscription. The reasoning behind this subscription alternative is to gather as much data as possible while using the free plan of Twitters API.

In the following part we present the results of using our previously stated methods. We show our resulting tool, how it is used and also how the data is visualized.

Fig. 2. Input field for tweet id or screen name for subscription.

In figure 2 we can see the input field used by the users to supply a tweet id to fetch data for it, or a screen name to start a subscription on a users next tweet.

Here in figure 3 we see the targeted tweet information displayed inside our application. There are more data available but these are the properties we considered to be interesting.

Figure 4 shows our retweeter locations graph. We decided to leave out the bar of users which did not have a location set for their profile since that bar always was much larger than the rest, making it hard to see them. Out of hundred retweeters the majority of given locations usually were empty. Furthermore we decided to only show locations used by two or more retweeters. The reason for this is that there were a lot of junk locations and non interesting data being shown otherwise.

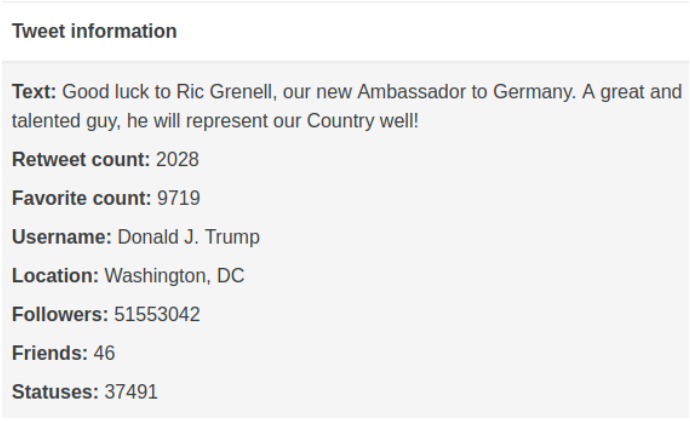

Fig. 3. Targeted tweet information.

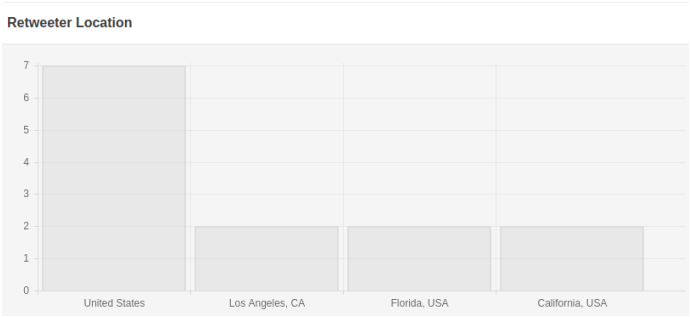

Fig. 4. Graph of retweeter locations.

In the figures 5, 6 and 7 we show the retweeters amount of followers, friends and statuses. This is the data we utilize out of the hundred retweeter profiles we are given by Twitter.

IV. Results and analysis

In this section our methodology and the result of our work is discussed.

The biggest hurdle we encountered was limitations set by the Twitter API. The only way with the free plan to collect retweets is to target a specific tweet and then get the last hundred retweets of it. You can not cursor through all retweets from the start to get a complete dataset, you are only able to get the last hundred. You can also only retrieve these retweets 300 times per 15 minutes. Which means if we wanted to get a good dataset of any tweet we would have to target a user before the tweet is made and then try to get the most recent retweets as often as possible for a certain period of time. This makes it hard to collect good data because we cannot know if a tweet is either false or true before it is made. We can also only target one user at the time, because of the rate limitations set by Twitter. This makes it very time consuming to collect data.

The data we are able to collect is also not the best. For every retweeter we collect the users location, followers count, friends count, and statuses count. The problem with the location is that a big majority does not enter a location for their profile, and even if they do it is not always a country. Most users instead enters their city, which makes it hard to retrieve their

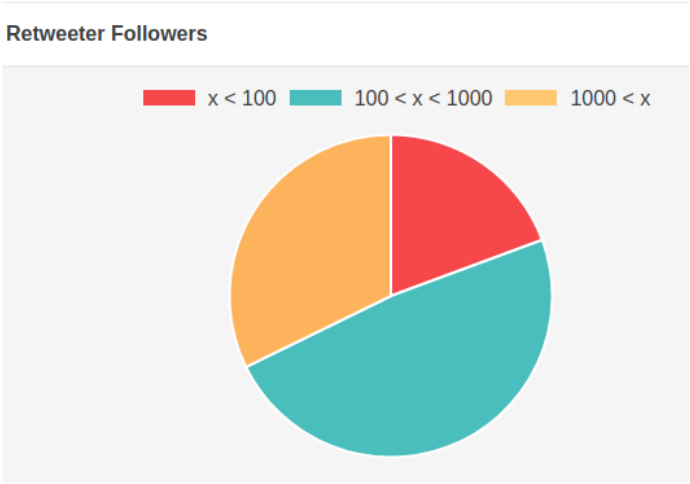

Fig. 5. Graph of retweeter followers.

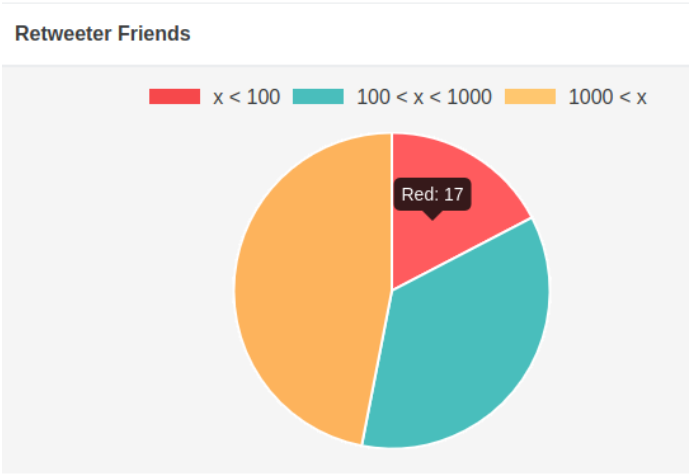

Fig. 6. Graph of retweeter friends.

country. There are also many users who input something totally unrelated to geography which is unusable for us. In the end we think we did the best we could with the limited possibilities offered to us by the free plan of Twitters API.

V. Discussion

The biggest improvement to be made with our method would be to use a higher tier of access to the Twitter API. Preferably the one that grants access to the full archive, but the one that goes back 30 days would also work. With them it would be possible to examine all retweets of any tweet, instantly, instead of having to subscribe to a tweet before it happens. It would also make it possible to go deeper into a retweet chain, extracting data about retweets of retweets. The application would be able to show the entire history of the tweet. One could theoretically build a map that would show where the tweet was retweeted from and when, down the entire chain. It would take some time to collect all the data from Twitter still, but it would be possible. So any future studies should consider using the premium tier. One alternative is to

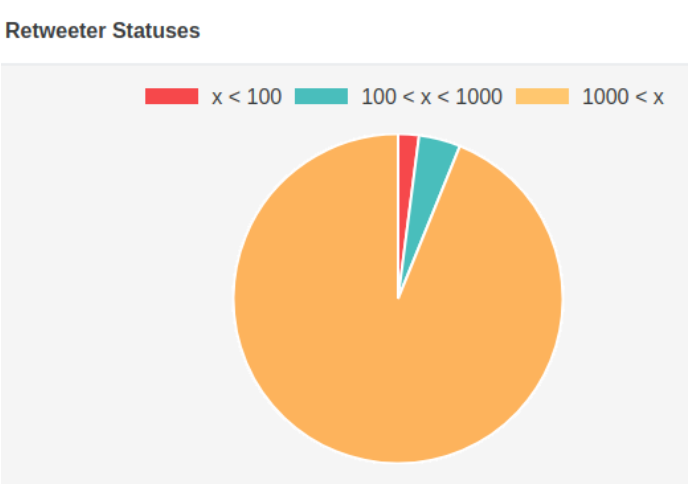

Fig. 7. Graph of retweeter statuses.

apply for data from Twitter for academic purposes, they did this in the MIT study so it has been done before [2].

VI. Related work

In a study published in Science [2] done by researchers at MIT, they went deep into the historical data made available for them by Twitter. With this data they could look deeper at any specific tweet. They were able to look at retweets of retweets, a so called cascade. They were therefore able to collect a complete dataset, that is all retweets, of every tweet, compared to our tool which only gets the samples Twitter returns on every call to their API. Our data is therefore not as complete, but we are able to check any new tweet in real time, whereas the researches at MIT were only able to look back at historical tweets. We also do not check a tweet automatically with fact checking sites as they did. This is kind of impossible for us because we are not able to know which tweets will be fact-checked since they are fact checked some period of time after they are tweeted. And since we do not have access to historical data we would at that time not be able to get a good data set. The researchers were also able to collect retweets of retweets, a chain, this is also too much for our tool because of API limitations. If a payed version of the Twitter API was used instead of the free plan, we could have done the same thing. So our studies are similar but works in different ways, mostly because of limitations set by the free plan of the Twitter API.

VII. Conclusions

The original goal of this project was to construct a method or tool that could help with classifying tweets as either true or false. We were not able to do that in a way that gives clear results. There are several reasons behind this, mainly that it is hard to build an algorithm that can take any text and then find sources that either backs it up or refutes it. This is why fact-checking organizations such as Politifact does it by hand on every individual case. Each case needs a new solution and different sources. The tool we built is instead related to the

original goal but has been scaled down significantly to better fit the project size.

Our tool does have its uses even thought it does not give a suggestion of whether a tweet is true or false. It works very well for tracking a users next tweet in real time. Since you have to target a user beforehand it works best on users that we know will get a lot of traction, an example would be Donald Trump or any other high profile person. The data we collect does give a pretty good picture of a users followers. Especially where they are from, and their activity on Twitter. We did some tests on Trump which showed that the absolute majority of his retweeters are very active members on Twitter with a small group of users who had tweeted less than 100 tweets.

A. *Future work*

A good future development of this tool would be to include some external third party package for classifying a tweets truthfulness by using the Twitter data that we collect. Hopefully this kind of packages will start showing up soon, which seems likely since the area is getting a lot of attention in the world as of late. Our tool would then be used for fetching data to the classifier and also for visualizing it, that could end up being what is needed to find connections between true/fake tweets and their tweet/retweet data. Also, if a higher tier of the Twitter API costs to much or cannot be used for any other reason, one could create multiple applications at Twitter and alternate which account is being used between requests. This would circumvent the rate limit, thought it may be frowned upon by Twitter if found.

REFERENCES

[1] Trump's Pants on Fire tweet that blacks killed 81% of white homicide victims [Online] Available: http://www.politifact.com/truth-o-meter/statements/2015/nov/23/donald-trump/trump-tweet-blacks-white-homicide-victims/ [Accessed: 7- Apr- 2018].

[2] S. Vosoughi, D. Roy, and S. Aral, *The spread of true and false news online*, vol. 359 no. 6380 1146-1151 Science, American Association for the Advancement of Science, January 19, 2018.

[3] Russia-linked posts 'reached 126m Facebook users in US' [Online] Available: http://www.bbc.com/news/world-us-canada-41812369 [Accessed: 5- May- 2018].

[4] Mark Zuckerberg Testimony: Senators Question Facebook's Commitment to Privacy [Online] Available: https://www.nytimes.com/2018/04/10/us/politics/mark-zuckerberg-testimony.html [Accessed: 5- May- 2018].

[5] MSB varnar för falska nyheter inför valet i Sverige [Online] Available: https://sverigesradio.se/sida/artikel.aspx?programid=83&artikel=6656044 [Accessed: 5- May- 2018].

[6] The Principles of the Truth-O-Meter: PolitiFact's methodology for independent fact-checking [Online] Available: http://www.politifact.com/truth-o-meter/article/2018/feb/12/principles-truth-o-meter-politifacts-methodology-i/ [Accessed: 5- May- 2018].

[7] Twitter for Node.js [Online] Available: https://www.npmjs.com/package/twitter [Accessed: 5- May- 2018].

[8] Twitter Apps [Online] Available: https://apps.twitter.com [Accessed: 5- May- 2018].