Ranking social media content to aid predictive modeling
COEN281: Term Paper

Giovanni Briggs (gbriggs@scu.edu)
Jeff Wick (jwick@scu.edu)
Vincent Tai (vtai@scu.edu)
Maxen Chung (mhchung@scu.edu)

June 13, 2017

Abstract

Polling users is an extremely important process. It is used in a number of applications, from business who want to evaluate consumer confidence, to presidential campaign managers who want to predict the outcome of a presidential election. Traditional polling techniques though are suffering from increasingly low response rates making it more difficult to gather data in a controlled setting. Meanwhile, social media usage is on the rise, and much research has been done to construct The issue though is that very few people have examined how to enrich their datasets with the underlying social graph. Due to the extreme ease of which people are able to publish content on social media, there is a lot of noise that can impact one's ability to accurately utilize social media data. This paper proposes using Google's PageRank algorithm for ranking users in the social graph and then using those ranks in the sentiment analysis process to create more accurate and stronger conclusions when using social media data.

Contents

List of Figures 2

1 Introduction **3**
1.1 Related Work 5
1.2 Hypothesis 8

2 Methodology **10**
2.1 Creating the Dataset 10
2.2 Ranking Twitter Users 14
2.3 Sentiment Analysis on Tweets 14
2.4 Ranking and Sentiment Analysis combined 15

3 Implementation **16**

4 Data Analysis **19**

5 Conclusion **22**

List of Figures

2.1 Overview of Twitter's Streaming API 13

3.1 Flow of data objects through our scripts. 17
3.2 Entity relationship diagram for the tables created by our Python scripts 18

4.1 Results from running our weighted sentiment analysis algorithm. 19
4.2 Histogram of the initial polarity results. 20

Chapter 1

Introduction

Our objective is to analyze new ways to generate accurate polls and predictions from non-traditional polling techniques. More specifically, we are attempting to find a way to accurately poll social media content. Traditional polling techniques such as random-digit dialing have been the primary form of polling. Random-digit dialing is particularly effective because of its randomness which creates very representative sample sets. Maintaining a representative sample set is extremely important in being able to accurately predict outcomes. The issue with random digit dialing in particular is that it is suffering from extremely low response rates. Over the past 15 years, response rates have dropped from 36% in 1997 to 9% in 2012 [1]. This necessitates a new way to be able to gain a representative sample set in order to poll public opinion. These sample sets can be used on a variety of topics such as predicting United States presidential elections, evaluating consumer confidence in a product, or evaluating where people stand on certain social issues.

Several other publications have proposed using sentiment analysis on social media posts in order to collect representative sample sets on a variety of topics as a way to replace traditional polling methods [9] [8] [10]. All of these papers focus on using sentiment analysis on the individual social media posts, but none of them try and use the underlying social graph to enrich their datasets and predictions. In this paper specifically, we are going to use Twitter as our primary social media network. Twitter has 328 million

monthly active users as of January 2017 [13]. This is relatively low compared to Facebook's 1.9 billion monthly active users [3], but Twitter comes with a few advantages. First, Twitter has a 140-character limit whereas Facebook posts do not have a limit. This means that that data collected from Twitter is easier to manage. While some users do try and create longer posts by posting multiple tweets that are meant to be read in a specific sequence, most users abide by the 140-character limit. Second, every Twitter follower can "follow" any other Twitter user. For each Twitter profile, the user has a "followers" and "following" value. The followers value shows how many people follow that user. Following shows how many other users this particular user chooses to follow. In Facebook, there is only the concept of "friends." Being a friend with someone is a mutual connection in which both users are connected. In Twitter, User A can follow User B, but User B does not have to follow User A. This creates a directional graph that can be used to further enrich the textual data that Twitter offers. Using this graph, we can identify users that are more important than other users, similar to how Google's PageRank algorithm ranks certain web-pages as more important than others.

This is related to the topics we covered in COEN 281: Pattern Recognition and Data Mining. In this course we have covered PageRank extensively, as well as the importance of maintaining a representative dataset. This problem that we are trying to tackle is also a data mining problem, not a machine learning problem. With machine learning, you already know what problem you are trying to solve. For any given problem, there are a set of known algorithms that you can apply to find the answer you want. In data mining, you don't necessarily know the answer to the problem. You are looking for patterns in the dataset that you have available. The primary focus of this paper is mining the data on Twitter to create a representative dataset that can then be used in machine learning algorithms. We will use some machine learning techniques to evaluate the effectiveness of our dataset, but the main focus of this paper is explaining a method for building a sample dataset from Twitter that can then be used in a number of applications.

Using sentiment analysis on social media posts is not a new idea. What is new is our approach to enrich that sentiment analysis with data from the Twitter social graph. In the next section, we will discuss previous work re-

lated to sentiment analysis on social media, as well as other attempts to replace traditional polling methods. We will then detail our hypothesis and goals more completely. In Chapter 2, we will discuss our methodology for collecting and analyzing our data. Chapter 3 will explore the technical details of how we gathered our data. We will follow this up in Chapter 4 with a deep analysis of the data we gathered and how it lines up with our expectations. Finally, in Chapter 5 we will summarize our findings and make recommendations for teams who wish to further explore this topic.

1.1 Related Work

As mentioned earlier, using sentiment analysis on social media posts is not a new concept. More generally, searching for ways to replace traditional polling methods is something that many people have been trying to do. Wei Wang, David Rothschild, Sharad Goel, and Andrew Gelman in their paper "Forecasting elections with non-representative polls" describe a method for predicting presidential elections while using a non-representative sample set [14]. The authors established a poll on the Xbox platform in 2012 during the 45 days immediately preceding the election day between Mitt Romney and Barack Obama. Everyday, they asked users if the election were to be held today, who would they vote for, and also collected attributes about the user such as age, gender, political party affiliation and location. Wang, et. al very quickly realized that their sample set was not representative of the larger US population. Their sample set was 93 percent male, compared to the 47 percent national average, and their initial results suffered because of it. Rather than re-sample their dataset to make it more representative, they used a technique called *post-stratification*.

Post-stratification is a technique that allowed the authors to mimic a representative sample set using their very skewed actual sample set. The idea of post-stratification is that you break your sample set into *cells*. Each cell is a unique combination of all of the available attributes. For example, if a sample set contained gender and age, then one cell would be *18 year old males* and another would be *60 year old females*. The next step to post-stratification is to find a representative sample set that has all of the same

attributes as you have in your sample set. The authors chose to use the exit polls from the 2008 election. Next you break the representative dataset into the same cells as your sample set, and compare the probability of a cell occurring in the representative set, Pr, versus the probability of that same cell occurring in the sample set, Ps. Each cell is then assigned a weight based on the relationship Pr/Ps. This relationship should make intuitive sense. If the probability of a given cell occurring in the sample set is too high, then we need to reduce its effect on the final output and vice-versa. The final step of post-stratification is to apply each weight to the output generated by a given cell.

Wang, et al. used a hierarchical Bayesian model to predict the probability that each cell would vote for Obama, and then post-stratified the result for each cell. First, they predicted how much of the popular vote Obama would win, and accurately predicted him winning the popular vote. Not only that, but they predicted the outcome with 0.6 percentage points. Wang, et al. also designed a model for calculating the number of electoral college votes Obama would win but applying their hierarchical Bayesian model and post-stratification on a state by state level. Since most states in the United States operate on a "winner-take-all" standard, if a state was predicted to have greater than 50 percent support for Obama, all of that's states electoral college votes went to Obama. The end result was not as accurate as their popular support estimate, but still predicted Obama winning with 312 electoral college votes versus his actual 332 electoral college votes. Wang, et al. prove a very effective method for using non-representative data to make accurate predictions and show a way to replace traditional polling techniques for predicting presidential, and local, elections.

Post-stratification comes with some issues though. Mainly, it requires that you have access to a representative dataset that describes the population you are examining and it needs to have the same attributes as your sample set. Luckily, within the United States, there are plenty of representative sample sets such as the US Census data and exit poll surveys. For our purposes, the use of post-stratification could be useful. There is no guarantee that the tweets that we take from Twitter will be representative of a given population. Post-stratification can allow us to adjust our sample set if we

discover that it is skewed in one direction or another. Unfortunately, the attributes we can gather from Twitter about its users are very limited, and so we may not be able to break our dataset into enough meaningfully distinct cells. This could have an impact on our ability to effectively post-stratify the data if necessary.

The effectiveness of post-stratification alleviates the pressure of needing to build a system that generates a representative dataset every time. So now, we can turn our attention to ranking Twitter users. One system that could work is TrustRank, which was proposed by Zoltan Gyongyi, Hector Garcia-Molina, and Jan Pedersen [5]. TrustRank is meant to be an improvement upon PageRank. With PageRank, a page is considered "good" if it has a high number of incoming links. The idea is that if a page is referenced by a lot of other pages (has a high volume of in-links), it must be worthwhile. TrustRank takes a different approach and operates under the assumption that "trust" is conferred from one page to another with the end goal of eliminating spam. If a highly trusted page links to another page, then that secondary page must also be a trusted source. If an untrusted page links to another page, then that secondary page must also be untrustworthy. While PageRank uses the in-links to establish a page's rank, TrustRank uses the out-links.

Unfortunately, the definition of "trustworthy" and "untrustworthy" are largely subjective and require manual intervention. It is not feasible then to label every page on the web. To get around this, TrustRank operates with a preset group of pages called the "seed." Each member of the seed is manually identified as "good" or "bad." If the page is considered to be a "good" page, it's TrustRank is 1, and its trust is then conferred upon all the pages it links to. However, the amount of trust is inversely proportional to the number of out-links that page has. If Page A has a TrustRank of 1, and it links to 4 pages, each of those pages will get a TrustRank of *1/4*. The idea is that the more out-links a page has, the less care was placed in choosing trust worthy pages.

The usage of TrustRank in Twitter has been explored by Shen Hua and Liu Xinyue [11]. Rather than conferring trust from a seed of trustworthy sources, these authors proposed doing the opposite - conferring distrust from

a seed of untrustworthy sources and named their algorithm *Anti-TrustRank*. Hua and Xinyue found that untrustworthy sources tend to operate as a pack and form a small, socially connected world. These untrustworthy accounts (which are referred to as "spammers") could potentially be connected to normal users, so care must be taken in the propagation of distrust. The act of following someone on Twitter is easily done, and so just because a spammer follows a normal user does not mean the normal user is also a spammer. To combat this, the Anti-TrustRank relies on the inverse-link relationship. The idea is to penalize users who actively follow spammers. In Hua and Xinyue's tests, the algorithm performed very well and placed more than 90 percent of spam followers to the bottom 10 percent of the ranking system.

1.2 Hypothesis

We believe that we can implement PageRank, with Twitter followers as being analogous to "links" in the original PageRank. By combining our ranking with regular Twitter sentiment analysis, we believe we will notice an increase in polling accuracy when compared to using traditional sentiment analysis alone. Our first hypothesis is that the gain in accuracy will be due to higher ranking users having more sway on the polling process. The intuition behind this is that users who have a high ranking will be users who are followed by other high ranking users. In the real world, these users have a high social cachet, and are therefore more likely to hold sway over people's opinions. Additionally, users who have a large follower base will be more retweeted and favorited. This has the effect of spreading their opinion to users who are not even followers, allowing their sentiment to be spread beyond just their immediate follower base. In this way, our ranking algorithm is seeking users whose Tweets have a maximal effect on the public opinion. By giving more weight to high ranking users, we believe that we will be able to obtain a more accurate depiction of the sentiment regarding the election, or whatever is being polled.

Our second hypothesis is that the improvement in polling accuracy will be caused by the diminished effect of low ranking users. Low ranking users should not be ignored, as we are trying to take a representative sample, but

their effect on the polling numbers should not be as great as those with a high rank. Low ranking users are likely one of two things: a regular person or a Twitter-Bot. If it is a regular person, then we want to count that person's sentiment, but they are unlikely to have sway over a non-negligible portion of the population. If it is a Twitter-Bot, then we do not want to count its sentiment at all, but because this is not the focus of our paper, we will settle for the low ranking it is likely to receive due to its low followers.

Chapter 2

Methodology

We will implement a PageRank-style algorithm, using Twitter followers as "links". We will need to handle the typical PageRank problems, such as "dead ends" (Twitter users who do not follow anyone). This problem is split into two parts - first is collecting a strong dataset, and the second is actually running our ranking algorithm.

2.1 Creating the Dataset

Twitter offers an Application Programming Interface (API) that allows developers to pull information from Twitter. The relevant API endpoints are:

1. https://dev.twitter.com/rest/reference/get/friends/ids: Given a user's ID, obtain a list of all of their friends.

2. https://dev.twitter.com/rest/reference/get/followers/ids: Given a user's ID obtain a list of all of their followers.

The issue with these endpoints is that the Twitter has rate limits on how many times you are allowed to query the endpoints. Both of these endpoints have a 15 requests for every 15 minutes [12]. In other words, we can only examine the followers and followees of a a single user every minute. This process is too slow to build a useful social network graph. Luckily, there are several datasets that contain a subset of the overall Twitter

social network graph. M. De Domenico et al. in their paper created a dataset of 456,631 nodes and 14,855,875 directed edges, but this process took them 25 days to complete [2]. Arizona State University offers another Twitter dataset representing 11,316,811 nodes and 85,331,846 directed edges [15]. Both datasets are rather old, with M. De Domenico et al.'s being from 2013, and ASU's being from 2009; however, they still provide us with a representative social graph that we can work with. We chose the ASU dataset simply because it has a greater volume of nodes and edges. M. De Domenico et al's dataset also only looked at user's who wrote a status to Twitter about the Higg's Boson particle, and so their social graph was a much smaller, and closely knit group of individuals; whereas the ASU set was much broader.

The ASU dataset represents the social graph in a series of rows containing Twitter user IDs. In each row there are two numbers, and represents an edge of the graph where the first ID is *following the second ID. In other words, the first ID is a* follower of the second ID. While this is a compact form to represent the social graph it doesn't lend itself well for our ranking algorithm. It is also too much data for us to manage in memory.

We chose to break the dataset down into a smaller subset and represent the data in a JSON format. The keys to the JSON file are user IDs, and each key points to another object. These sub-objects contain three keys themselves:

1. *followers*: a list of the Twitter IDs that are following the user
2. *following*: a list of the Twitter IDs that this user is following
3. *weight*: 1 divided by the length of the *following* list. This is the weight of all the outgoing edges that would be used with PageRank

Our resulting file does not contain all of the edges. Instead, we chose to only read one out of every 25 edges giving us a resulting graph of 3,413,274 directed edges. Every time we read an edge, we made two entires. Since the first value follows the second value, the first value was placed in the second values "follower" key. The second value was then placed in the first value's "following" key. At the end of this process we had 1,628,757 nodes. It is a smaller and more compact representation of the social graph from the ASU

dataset. This does however reduce the average number of edges per node from 8 to 3. However, even in the ASU dataset, there are some nodes that have hundreds of thousands of followers, and some that have none. In our condensed representation, the maximum number of followers that a single node has is 22,646. The max number of users that a single node is following is 8,576. Our condensed form maintains this property that some users have many, many followers while some have none, but we may have increased the number of dead ends.

Now that we have our social graph we needed to obtain a set of tweets. Unfortunately, Twitter's Terms of Service and Privacy Policy prohibit the storage of user tweets. The Stanford Network Analysis Platform had a dataset that contained user IDs and tweets, but it has since been taken down by request from Twitter.

We still have a major problem - we don't know how many of the user IDs in our social graph are still active. In order to determine that, we would have to do a *GET /user/lookup* call on the Twitter API. We can request information for a 100 users at once, and at a rate of 300 requests per 15 minutes. At this rate, it would have taken us about 13 days to determine who out of the 1.6 million users are still active. At this point, we still don't have any tweets to actual run sentiment analysis on. So we would need to take this smaller subset of users and then make more API requests to gather tweets from them. The 13 day process of simply determining which users are still active is already too prohibitive given the time constraints of this project.

Instead, we chose to use Twitter's Streaming API to gather tweets about a particular topic. Figure 2.1 shows how the process works. First, you provide a set of phrases that you want to receive tweets about, and Twitter will send a small subset of its Tweets to you as they occur in real time. We asked the streaming API to send us tweets that contain the phrases "trump", "donald trump", or "realDonaldTrump."

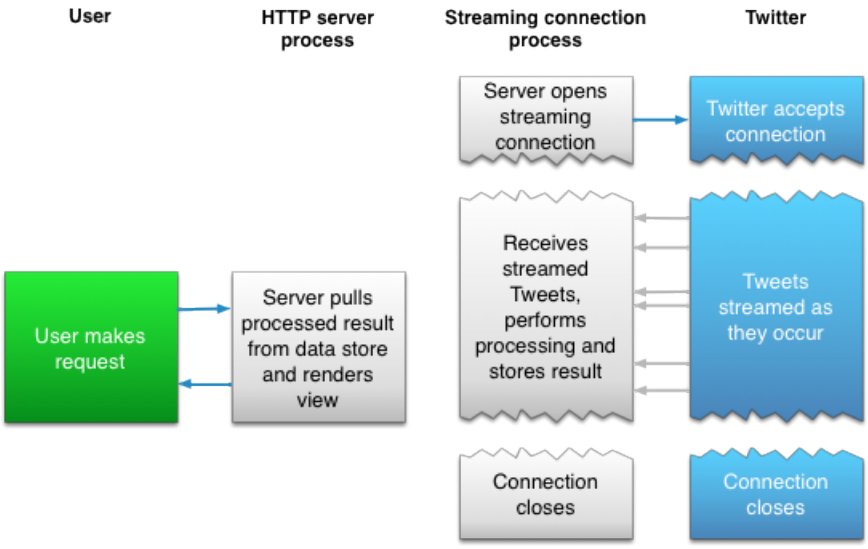

Figure 2.1: Overview of Twitter's Streaming API

The issue with this approach is that it is highly unlikely that an incoming Tweet is from someone in our social graph. To overcome this, we decided to randomly assign incoming Tweets to users in our social graph. First, we hash the 1.6 million user IDs to consecutive integers from zero to 1.6 million. Then, as we receive Tweets from our stream, we hash the user ID of the author of the Tweet to one of these consecutive integers. After streaming data for about 10 hours on June 1st, 2017, we had curated 3,345,566 million tweets and dispersed them across approximately 860,000 of our users. This leaves 200,000 users in our dataset that were not given a tweet. We decided that this disparity is fine since some users use Twitter more often than others, and not every user in a social network graph is going to share a tweet about a given topic. They can still be used in our ranking process, but their rank just won't be directly applied to the tweets we gathered.

It is also important to note that June 1st, 2017 was the day that President Trump withdraw the United States from the Paris Accord [6]. The tweets we collected may have more to do with Trump's decision to leave the Paris Accord than just about how people feel about Trump in general. In fact,

doing a manual random sample from the tweets shows that many of them are related to this specific event.

2.2 Ranking Twitter Users

Now that we have a dataset of Twitter users and their respective tweets, we can begin to look at how to rank Twitter users in order to aid in predictive polling. We chose to implement PageRank on the social network graph that we created in order to rank each user. Each user is given a weight between 0 and 1 that represents their relative importance in the graph. The idea again is that we will be able to use this ranks in combination with the sentiment analysis to create a more accurate poll.

2.3 Sentiment Analysis on Tweets

We used an open source Python library called TextBlob [7] to perform the sentiment analysis. Running sentiment analysis on the tweets gives us two values:

1. *polarity*: floating values ranging from -1 to 1, where -1 means negative sentiment, 0 is neutral and 1 is positive.

2. *subjectivity*: floating values from 0 to 1 where 0 means the statement is objective and 1 means the statement is extremely subjective.

We used TextBlob to generate these results for every Tweet in our database, and created a new table called *sentiment* that contained a given tweet's ID value and the corresponding polarity and subjectivity values. Our primary concern in this paper is the use of the *polarity* value; however, the subjectivity value still has uses. For example, if one wanted to eliminate heavily subjective text from their polling analysis, they could do that by setting a threshold on the *subjectivity* field or vice versa. Tweets that are purely objective are likely to be more neutral in tone, and so they do not tell us as much about how people feel towards a given topic.

One could also combine the two values for a very different result. The goal of using sentiment analysis as a replacement for polling is to identify how people feel. If we use the *subjectivity* value as a weight on the *polarity* value, we will increase the importance of strongly subjective tweets and decrease the importance of objective tweets. This process could be especially helpful if many tweets are only slightly positive or slightly negative, but are very subjective. The high subjectivity would help move the tweets from neutral territory into strong positive or negative sentiment.

2.4 Ranking and Sentiment Analysis combined

To achieve our desired results, we will have to tune how our ranking algorithm interacts with the polling data. Ideally, we want to give high weighting to those with a high rank, but not so much that it makes those with low rank negligible. In fact, we want the low ranking users still have an influence on how the algorithm decides. This is because the low ranking users are the general population, and their sentiment towards topics is important. While high ranking users may be better indicators of the general sentiment, the sentiment of the majority is also very important.

Chapter 3

Implementation

Chapter 5 contains a high level understanding of how our project works. In this chapter we will discuss the actual implementation of our project in greater detail.

We used Python 3.5 to write a set of scripts to perform each step mentioned in Chapter 5. Our first script is *format_data.py* which takes in the ASU dataset, selects a sample of the data, and outputs a JSON formatted file which is then used in *small_pagerank.py* Our PageRank implementation than creates a CSV file where each row contains a user's ID, and their corresponding rank. It will also write this data to a table in our SQLite database named *ranks*. The CSV was created because we originally wanted that data to be more portable than the other pieces, but we eventually needed to include it in our database anyway.

To collect data from Twitter's Streaming API we created a script called *get_tweets.py* which takes in the JSON formatted social graph, and a query to send to Twitter's Streaming API. Since we are collecting a large volume of tweets over time, we didn't want to hold all of the tweets in memory and then dump to a CSV or JSON file when we were ready. Instead, *get_tweets.py* creates a SQLite database and writes the tweets to a table named *tweets*. This script will also create a table to hold the user information, named *users*, from the JSON formatted social graph. Our sentiment analysis script, *sentiment.py*, adds another table to the database named *sentiment*.

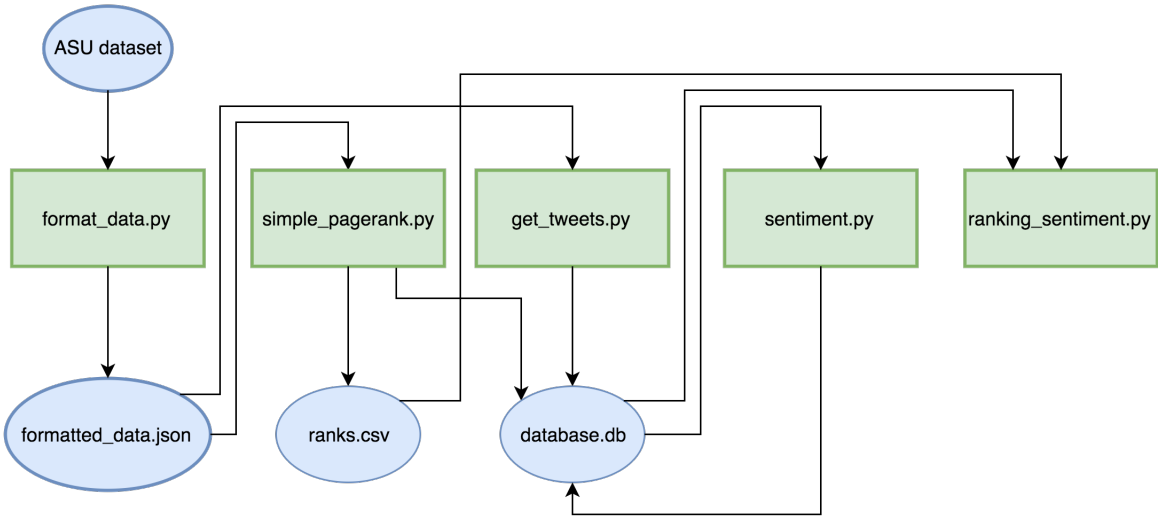

Figure 3.1: Flow of data objects through our scripts.

Figure 3.1 shows how the different data objects are created and manipulated through our scripts. We first start with the ASU CSV dataset which *format_data.py* will reformat into a JSON file. That JSON file (which here is named *formatted_data.json*) is then passed as an input argument to *simple_pagerank.py*. This script will then output a CSV file with the corresponding ranks for each user. Our JSON formatted social graph is also used in *get_tweets.py* which creates our SQLite database. This database is then used in the *sentiment.py* script which writes its results to a new table in the same database. Finally, the database and CSV file with the user rankings are used as the input for our final script which combines the PageRank results along with the sentiment analysis to give us the results detailing how people feel towards our chosen topic.

We recognize that these are a lot of different files and could be condensed. The ranking output should also be a table in our SQLite database. Luckily, the results are small enough that they fit into main memory and can easily be joined with the sentiment analysis results.

Figure 3.2 shows the entity relationship diagram for the three tables in our database. Every tweet must belong to one user, but not every user has

a tweet. Also, any given user may have multiple tweets. Every tweet in our database does have a set of values in the *sentiment* table.

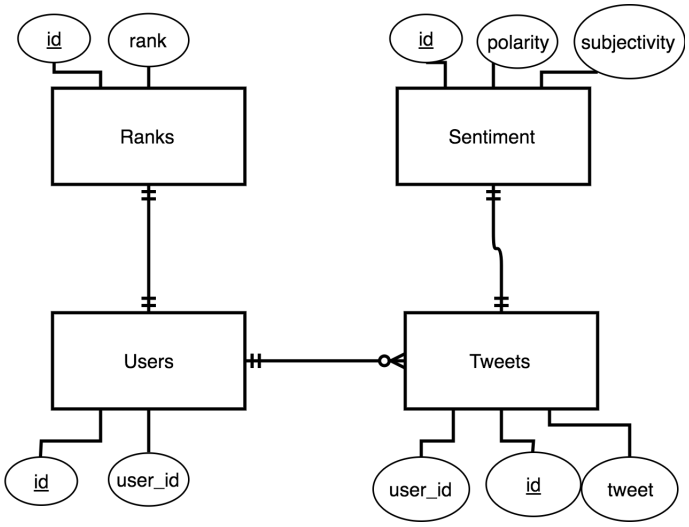

Figure 3.2: Entity relationship diagram for the tables created by our Python scripts

Below is a code snippet that shows how to join all three tables together.

```
SELECT * FROM users, tweets, sentiment ON users.user_id =
    tweets.user_id AND tweets.id = sentiment.id
```

If the ranking output was also a table named *ranks*, we could join all four of these results with the following query:

```
SELECT * FROM users, tweets, sentiment, ranks ON users.user_id =
    tweets.user_id AND tweets.id = sentiment.id AND users.id =
    ranks.id
```

Chapter 4

Data Analysis

Once we gathered all of the necessary data-points into one table, we were able to compute scores for the positive and negative sentiment achieved both with our ranking algorithm, and without our ranking algorithm. Additionally, we looked at what values to use as our threshold for a positive and negative tweet. Figure 4.1 shows our results.

	Positive	Negative
Unranked w/o Adjustment	57.6%	42.4%
Unranked w/ Adjustment	44.1%	55.9%
Ranked w/ Adjustment	39.5%	60.5%

Figure 4.1: Results from running our weighted sentiment analysis algorithm.

The polls for this time have Trump's approval rating hovering around 39 percent [4], so this is mark that we are aiming for. Starting with the unranked, unadjusted sentiment analysis, we can see that our results are not even on the correct side, with Trump actually having more positive sentiment than negative. To adjust TextBlob's tendency toward positivity, we

used 0.13 as the threshold for a positive tweet, meaning that any tweet with polarity greater than or equal to 0.13 is positive, and less than 0.13 is negative. The reason for the 0.13 is from human observation on a sample of the tweets. For example, the tweet "@realDonaldTrump: Join me live at the @WhiteHouse" is a completely neutral tweet, meaning is it not expressing support or opposition toward Trump.

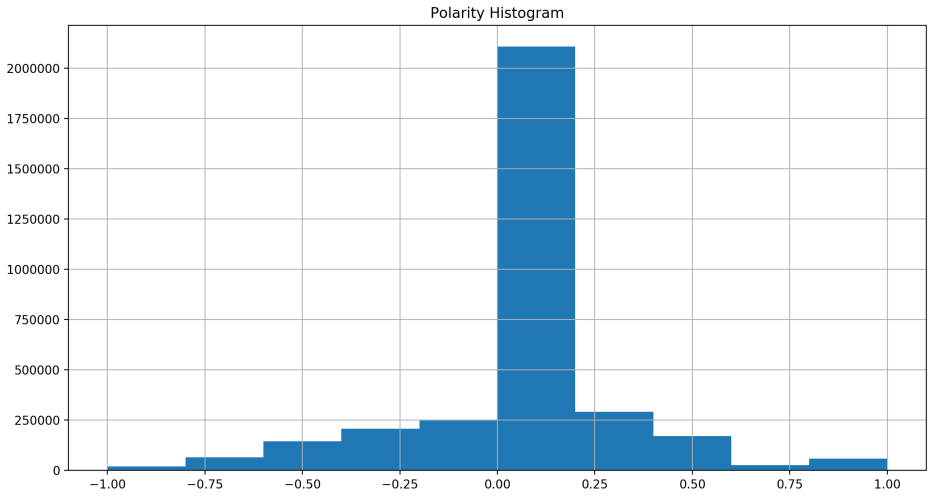

Figure 4.2: Histogram of the initial polarity results.

However, this tweet received a polarity ranking of 0.13. 0.13 also happens to be about half a standard deviation of our results. Figure 4.2 shows the histogram of the unweighted polarity results. Over 2 million of the tweets are within the [0,2.4) bucket, showing that a vast majority of the tweets are neutral, skewing towards positive. We also experimented with different values of subjectivity, and found that not including anything with a value of 0.2 or less improved our results. Using these two rules, we used the following formula to come up with the results in Figure 4.1:

$$\text{if polarity} > 0.13 \text{ and subjectivity} > 0.2$$

add 1 to positive sum
else if polarity < 0.13 and subjectivity > 0.2
add 1 to negative sum
Positive = positive sum /(positive sum + negative sum)
Negative = negative sum/(positive sum + negative sum)

Once we have accounted for the adjustment, we can see that our polls now accurately reflect that Trump is viewed more negatively than positively. However, the value is not quite the value that we are looking for, which is where our ranking comes in. The reasoning behind our algorithm is that users with very low rankings should probably not used at all, while users with high rankings should have a large amount of influence. To this end, the first approach we tried was to add the rank to the sum instead of adding 1. However, this yielded roughly the same results (< 1 percent) as the unranked results, so we clearly were not taking into account the influence of the rankings enough. Once we used ranking squared instead, we found that the results yielded the roughly 39 percent approval rate we were looking for. This makes sense, as we want the low ranking users to have very little effect and the highly rated users to have a large effect. Looking at the rankings of the users, the top ranking users have rank values in the 10^{-3} range, while low ranking users have ranks in the 10^{-7} range. So, by squaring the values, we are obscuring the low ranking users.

From these results we can conclude several things. The first is that the initial unranked without adjustment value shows that the sentiment analysis is flawed, or we gave the algorithm tweets not related enough to the subject. A cursory reading over the tweets reveals that some of the tweets are not very on topic, or mention Trump in passing, instead of as the main topic. However, the fact that the tweet quoted above received a polarity of 0.13 shows that the sentiment analysis itself is slightly flawed. By using the adjustment, we are able to get rid of the bias of Textblob, while using our ranking algorithm allowed us to correct for users whose input should receive less consideration.

Chapter 5

Conclusion

In this paper, we have attempted to show one way in which one can use social media to replace traditional polling techniques. By using a combination of sentiment analysis on user's social media posts and ranking those users with an algorithm such as PageRank, one can create a metric that shows how people feel towards a particular topic.

One of the biggest challenges we had was collecting data to actually run the ranking and sentiment analysis on. We gathered tweets over a ten hour period using a rather small social graph. If we had more time, we could have gathered more data and made a more representative social graph. Other papers took around 25 days to collect all of their data, and we would have benefited from having that amount of time. Especially since Twitter's social graph is growing at a much slower rate, it would not be infeasible for someone to create a full graph and run a ranking algorithm on it. It would be best if Twitter could provide it themselves and update the ranking whenever a new node is added, but until then, researchers can create their own social media graph.

Another problem we have is validating our results. The only real metric we have to show that our results are valid is that they closely align with current polls; however, one of the objectives of this paper was to try and show that sentiment analysis might be *better* at polling than traditional polls are. While we feel confident that our results are accurate and show the benefit of using a combination of ranking and sentiment analysis algorithms, it is

hard to prove that our method is any more accurate than existing polling methods. Also, since we were unable to gather more information about each user in the Twitter social graph, such as age, gender, and location, we were not able to run poststratification as discussed in our Related Works section. Poststratification may have also helped improve the accuracy of our results.

In the future, we hope that others will continue examining how to use PageRank, and other ranking algorithms like it, as a way to improve sentiment analysis of social media content. The applications of this method are not limited to just presidential approval rating. It can be used to examine consumer confidence in a product, attitudes and sentiments towards social issues, and even in prediction of political elections. As the response rates to traditional polling techniques continue to decrease, it will become increasingly important that we find ways to accurately gauge people's attitudes and sentiments through other mediums.

Bibliography

[1] Andrew Kohut, Scott Keeter, Carroll Doherty, Michael Dimock, Leah Christian. Assessing the representativeness of public opinion surveys. 2012.

[2] M. De Domenico, A. Lima, P. Mougel, and M. Musolesi. The anatomy of a scientific rumor. *Scientific Reports*, 3:2980 EP –, 10 2013.

[3] Facebook. Facebook q1 2017 results, 2017.

[4] Inc. Gallup. Gallup daily: Trump job approval.

[5] Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with trustrank. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 576–587. VLDB Endowment, 2004.

[6] Kevin Liptak and Jim Acosta. Trump on paris accord: 'we're getting out', Jun 2017.

[7] Steven Loria. Textblob.

[8] F. Neri, C. Aliprandi, F. Capeci, M. Cuadros, and T. By. Sentiment analysis on social media. In *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 919–926, Aug 2012.

[9] O'Connor, Brendan, Balasubramanyan, Ramnath, Routledge, Bryan R., and Smith, Noah A. From tweets to polls: Linking text sentiment to public opinion time series.

[10] Saad M. Darwish, Magda M. Madbouly, Mohamed A. Hassan. From public polls to tweets: Developing an algorithm for classifying sentiment from twitter based on computing with words.

[11] Shen Hua, Liu Xinyue. Propagating anti-trustrank with relationship strength for fighting link farming on twitter.

[12] Twitter. Rate limits: Chart.

[13] Twitter. Selected company metrics and financials, 2017.

[14] Wei Wang, David Rothschild, Sharad Goel, and Andrew Gelman. Forecasting elections with non-representative polls. *International Journal of Forecasting*, 31(3):980 – 991, 2015.

[15] R. Zafarani and H. Liu. Social computing data repository at ASU, 2009.