Does misery love company?
Sentiment change over the Twitter social graph

COMP 4601 - Project

Student Name: Brian Ferch
Student Number: 100962115

Student Name: Jules Kuehn
Student Number: 100661464

Winter 2019

1 Abstract

We investigate tweet sentiment over time for 400 socially connected Twitter users by dynamically visualizing changes in tweet sentiment for roughly 400,000 tweets from 2016 to present.

The project scope consists of crawling the social graph, gathering tweets for each user in our graph, translating and analyzing sentiment for each tweet, and visualizing these sentiments over time. Each user is represented as a node, and the color of each node indicates that user's sentiment at a given time. The graph is displayed using a force-directed layout. From an initial seed node, followers of a user are added along with directed edges for each $(user, follower)$ directed relationship.

2 Introduction

A 2015 paper by Tsugawa and Ohsaki[1] found that "negative messages are likely to be reposted more rapidly and frequently than positive and neutral messages". We did not investigate retweets but rather sought to answer whether sentiment spreads more generally between users. This could confirm and expand on the results of that paper, or simply provide some useful insight into demonstrating the folk wisdom "misery loves company": that negative sentiment is contagious.

In lieu of any substantial statistical analysis, we visualize the results on an animated graph, allowing for us to explore the data interactively and spot trends.

We will begin with a general overview of the data sources and technologies used and a review of related work. Then we will discuss the specifics of our implementation, and results obtained.

3 Project overview

Twitter offers a free rate-limited API[2] which allows us to retrieve the required raw data from queries on a screen name or user id:

1. **userId** : a large integer, ex. 946341920036478976 for our seed user
2. **tweets** : most recent ≥ 3200 tweets for this user consisting of $[(time_1, text_1), (time_2, text_2), \dots]$
3. **followers** : $[userId_1, userId_2, \dots]$

Many of the gathered users had predominantly non-English tweets. In order to run sentiment analysis with VADER (Valence Aware Dictionary and sEntiment Reasoner)[https://github.com/cjhut we first had to translate all tweets to English. This was accomplished using Google Cloud Translate[4], a commercial service.

Based on visual analysis of the data, cleaning was performed programatically. This yielded several datasets usable by the client:

1. **tweets** : all tweets for all users, sorted by time $[(time, userId, sentiment, text), \dots]$
2. **users** : information for specific users such as average sentiment, number of tweets, and followers. $\{userId : userInfo, \dots\}$

This data was then displayed as a graph in JavaScript using the d3.js library[5]. Sentiment is then animated on each node by the timestamps of tweets. Some smoothing (rolling window averages) is selectively applied to the data to aid in spotting trends at different time granularity.

4 Related work

Projects with similar scope, or technologies related to the implementation of your solution. Briefly mention the projects and their relation to your report, and annotate appropriately.

5 Methodology

5.1 Data acquisition and processing

Although a historical archive of Twitter data may have been sufficient for our purposes, we opted to acquire some data ourselves via Twitter's API.

5.1.1 Tools employed

Cloud services were used heavily in the processing of this data. The data acquisition and processing was all performed on Google Colaboratory[x]'s free computing instances, storing temporary data on a mounted Google Drive for persistence. This allowed us to offload tasks which required substantial network activity, processing time, and storage. Processed results were periodically stored between intermediary steps in either Python's native 'pickle' format, or JSON and CSV formats.

We leaned heavily on Python libraries for specific tasks:

- **Tweepy** for calling the Twitter API
- **nxGraph** for creating a user graph and performing page rank
- **VADER** for analyzing tweet sentiment
- **matplotlib** for inspecting the data while cleaning
- **numpy** for general processing tasks

The above are all standard tools for these uses in Python, but let's explain why we chose VADER. Unlike mainstream libraries such as CoreNLP[x], VADER is specially tuned to analyse social media text. It is able to recognize emoticons such as :) and Unicode emoji[x], and also takes capitalization and punctionation such as ALLCAPS!!! into account for sentiment analysis.

Twitter's free API was sufficient for extracting 400 users and up to 3200 tweets each. (Rate-limiting made it impossible to achieve a larger crawl in the time-frame we had.) Google Translate's free API was however insufficient due to both rate limiting and trouble decoding Unicode emoji. Google's commercial Cloud Translate service was up to the task, although rate limits had to be manually increased to 10 million characters per 100 minutes.

5.1.2 Methodology

Seed user: First, we identified a seed node. After some experimentation with using popular celebrity accounts as seed nodes, quick inspection of the resultant graph showed that it was too broad and not deep enough for the desired visualization. We found that using a normal,

less popular user resulted in a graph better conforming to our intent (See Figure X). This user was selected by browsing followers of the City of Ottawa account. Our seed user[x] is local to Ottawa, and has only 2 tweets and 13 followers. (In retrospect, a user with more moderate activity (hundreds of tweets, dozens of followers) would have likely produced better results.)

Building a graph: Crawling the user graph had already been implemented simply in Python by Yuya Takashina's twicrawler[`https://github.com/ytakashina/twicrawler`]. We used this as base code, making modifications to crawl 'forward' through a potential chain of influence by adding a seed user's followers to a priority queue. The priority was determined by the PageRank score of that user's node in the partially acquired social graph. The aim was to process more popular users first, as we assumed these users would be of higher quality and influence.

From this graph of user nodes, we removed all but the 400 highest page-ranked nodes.

Getting tweets: Then, for each of the 400 user nodes, we retrieved the last 3200 tweets from each user using Tweepy based on an existing script[`https://github.com/dpzmick/one-liners/blob/master/tweet_dumper.py`]. This retrieved the tweets in batches of 200. We frequently ran into rate-limiting and had to run the script at intervals on appropriately sized mini-batches of users.

We then roughly followed the tutorial "(Almost) Real-Time Twitter Sentiment Analysis with Tweep & Vader"[`https://towardsdatascience.com/almost-real-time-twitter-sent`
Tweets were cleaned of links, retweet handles, user handles, and special characters (leaving emoji). The tweets were sent in batches to Cloud Translate, and the translated text was analysed for sentiment with VADER, yielding positive, neutral, negative, and compound sentiment scores for each tweet. We wanted a single real number for our sentiment visualization, so we stored only the compound sentiment score along with each tweet.

Inspecting and processing data: We now have all *tweets* for all users, sorted by time and scored with compount sentiment $[(time, userId, sentiment, text), \dots]$. We gather together some useful information on the users from the graph and tweets into *users*. This includes the followers of that user (nodes connected through out-edges on the graph), and the average sentiment scores. A second average is made of non-zero sentiments.
The raw sentiment values of all tweets over time is shown in Figure 1. Manual inspection of zero-sentiment tweets showed that many were of low quality (tweets which were simply URLs, stripped characters, or poorly translated). Therefore, the zero-sentiment tweets were removed. The non-zero average for each user was then subtracted from each of that user's sentiments s:

$$s_{new} = s_{raw} - avg(s \in tweets_{user}, s \neq 0)$$

As our visualization is time based, we want to make sure we have sufficient tweet density

for a useful visualization at a given time. Inspecting the timestamps of the tweets showed that less than 10% of gathered tweets occured between 2009 and 2016 (see Figure X). The processed **tweets** list now consists only of non-neutral tweets from January 1, 2016 to the crawl date April 13, 2019, with sentiment scores normalized by user average. We are left with 61.6% of the total tweets gathered, but de-noised and with sentiment value somewhat exagerrated, making it easier to visualize change in sentiment.

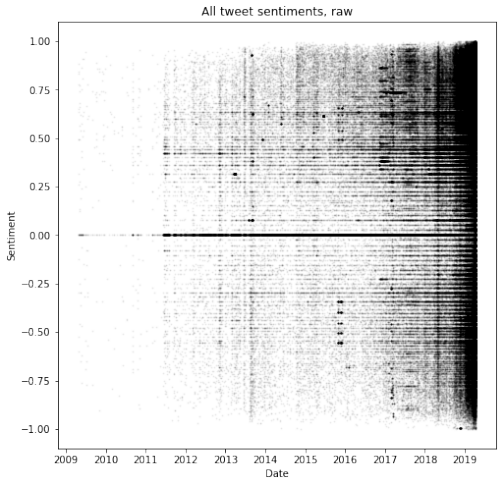

Figure 1: Raw sentiments

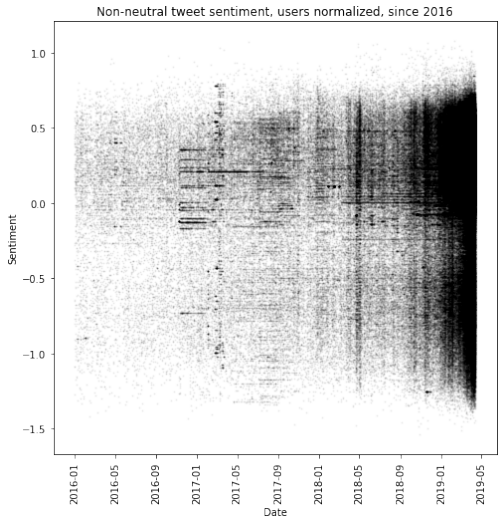

Figure 2: Processed sentiments

The time granularity of this data was very fine (by second) at this point, making it unsuitable for animation over several years. To compensate for this, and to smooth sentiment values for more useful visualization, we employed several rolling window averages:

1. 3 day window, 1 day step (length: 1199)
2. 1 week window, 3 day step (length: 400)
3. 2 week window, 3 day step (length: 400)
4. 4 week window, 3 day step (length: 400)

Users who did not make any tweets in that time window are not given an average sentiment for that window.

To better understand the data, we graphed the averages for each of these windows. We can see that increasing the window size shows less granular trends over months while the smallest window size shows more granular trends over weeks and even days (see Figures X-X).

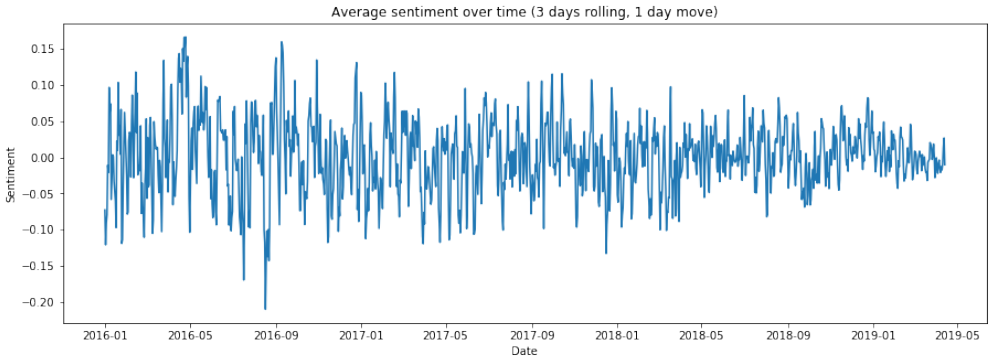

Figure 3: Average tweet sentiment for all users, 3 day rolling window

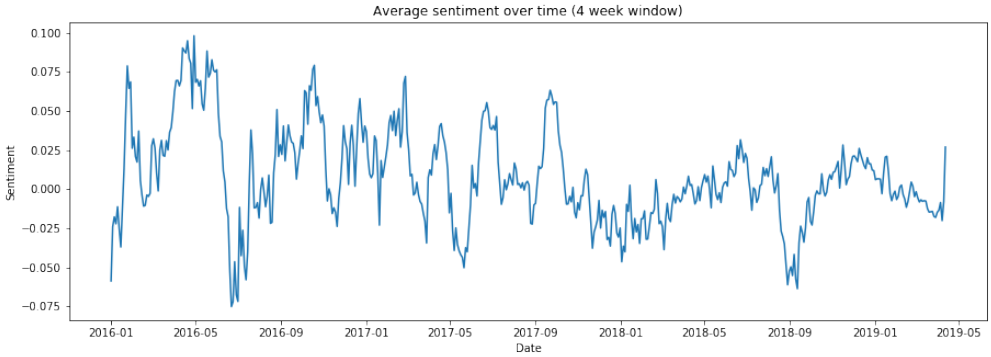

Figure 4: Average tweet sentiment for all users, 4 week rolling window

5.2 Visualization

6 Discussion

This is the opportunity to showcase analysis or resulting benefits from your project implementation. Without getting into Future Work too much, briefly discuss what setbacks and desires your project implementation experience has left you with. Other than that not much advice on discussion; varies by project scope.

7 Future work

Discuss possibilities of future work, for example in terms of 1) what would you consider implementing given more time on this project and 2) how can this project be used by or furthered by the open source community to improve future works?

8 Conclusion

Summarize the content of the report with a focus on the goal, analysis, and resulting value. Recap relevant points from the discussion, but do not restate it. Finally, you can choose to touch on future works as you close out the paper.

A
Figures and tables

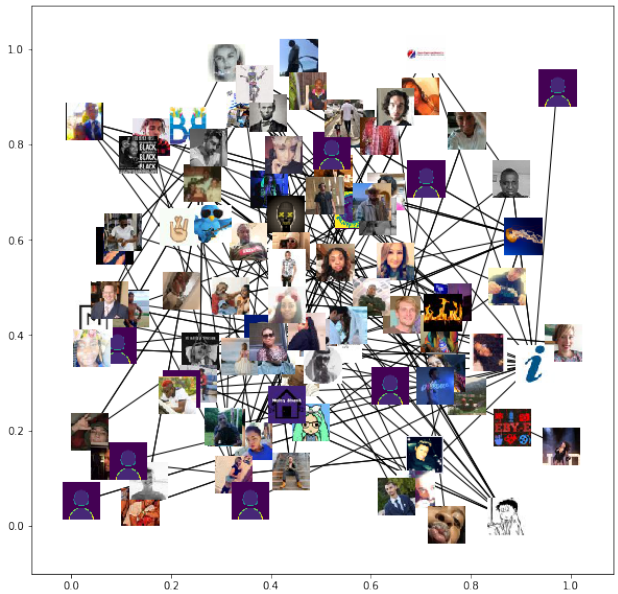

Figure 5: Highest PageRanked users from unpopular seed user

B
References

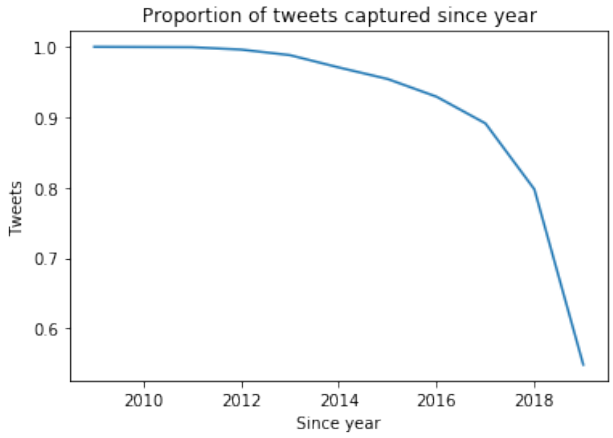

Figure 6: Percentage of total tweets captured by starting year