Twitter Sentiment Analysis App

COMP 4601- Mobile Multimedia

Ibrahim Manssouri

Project report submitted to
Dr. Anthony White

Carleton University
School of Computer Science

April 9th, 2017

Abstract

Social media platforms provide everyone with access to the internet the ability to voice their opinion or concern regarding a multitude of topics. But with the enormous volume of content comes a growing difficulty to discern how people overall feel towards a particular product, service, public figure or event. Focusing on the Twittersphere [1], the proposed Twitter Sentiment Analysis App uses natural language processing to analyse the overall sentiment and visualise the results in real-time on a per topic basis, all of it packaged in a user-friendly mobile application.

Introduction

Public opinion and sentiment is considered a very important metric used in decision making in organisations and governments all over the world. Twitter with it's growing active user base estimated at more than 300 million [2] provides a a good set of data in regards to people's honest opinions and sentiments. Despite that, this data is rarely analysed outside of academic research, even though it could be useful to brands, products, services and governments. For organisations, Twitter does not currently offer a way to see if people think positively or negatively about their products or services. The same applies to political figures when it comes to elections or law making. This can of course be done manually, but they are labour intensive and inefficient. As such, the main motivation behind this project is to create a user friendly way to analyse which tweets are positive and which ones are negative, in order to better understand people's sentiment individually, and then use that data to generate a meaningful visualisation summarizing the analysis of hundreds or thousands of tweets into actionable data. This would allow to see where the majority is leaning towards in a particular trending topic on Twitter and any interested party would then be able to make decisions based on tangible, meaningful information.

This report will go over some background information on the methods and reasoning behind the motivation to solve this problem, discuss some related work done in the field of sentiment analysis, go into detail regarding the particular approach undertaking, discuss the outcome of the approach and finish up with what could be implemented in the future.

Background

Sentiment analysis or opinion mining is the use of natural language processing, and text analytics to identify and categorize opinions about a particular topic [3]. It is mainly applied to any subjective user generated content such as social media posts, online reviews, customer service and other forms of textual content. At it's simplest form, it relies on finding whether the text provided can be categorized under a positive,

negative or neutral polarity. More advanced forms can further analyse the tone or emotion behind each of these polarities, and as such, creating more categories e.g: Sad and mad are both negative sentiments yet each have different consequences. In our case, the application relies on the three basic categories: positive, negative and neutral. We also use keyword-oriented classification using a curated list of words called AFINN [4], which allows us to provide somehow accurate results in real-time. The AFINN list, although small is often validated by running an analysis using its scores in comparison with review content and scores found on Yelp, IMDB and Amazon.

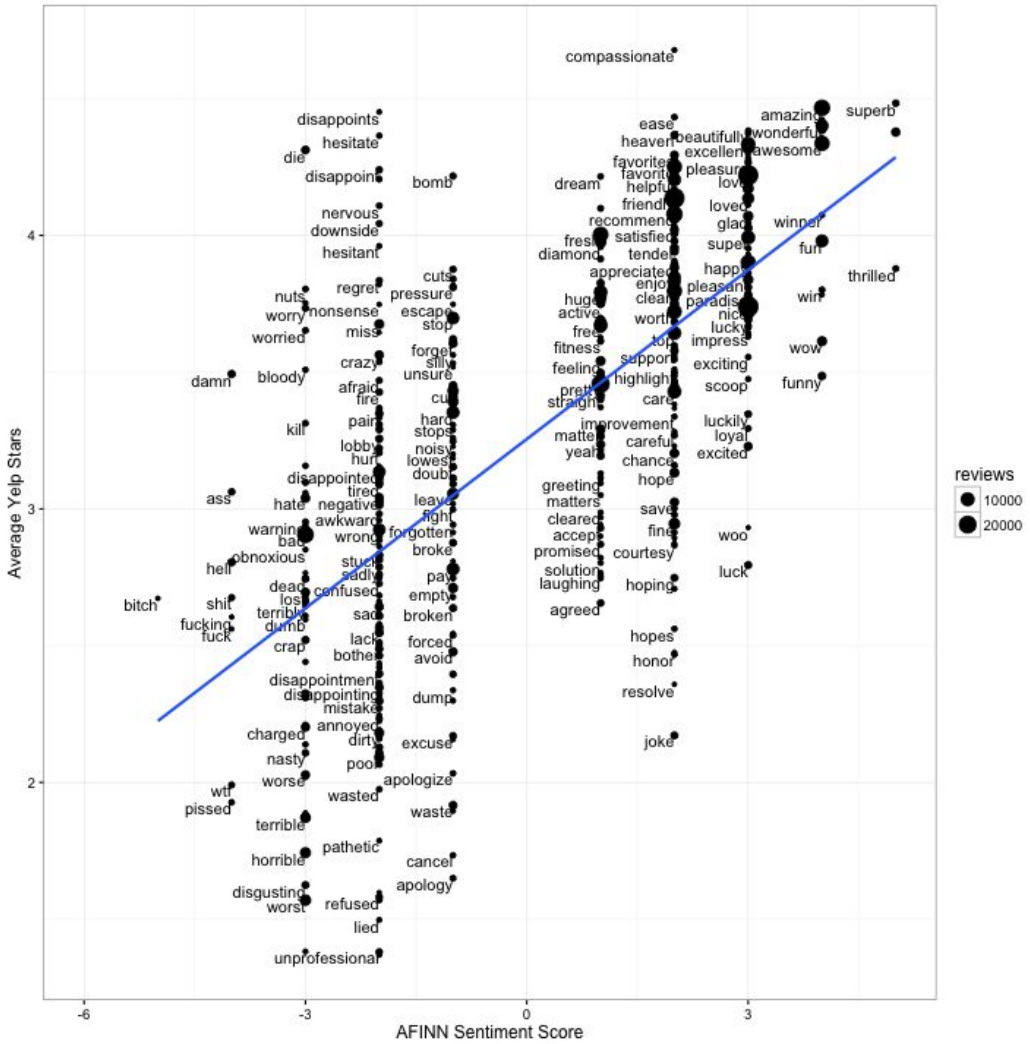

Fig. 1: AFINN per word score in correlation with average Yelp review scores [5]

As for Twitter, as stated above has hundreds of millions of active users, each tweet can represent a subjective sentiment towards any topic of interest. As such it makes for a great source of textual data that can be analysed for sentiment. Each tweet is only limited to 140 characters, a user writing a tweet will limit it to only one topic of interest. This makes it so that every tweet is usually about one topic in particular.

Related Work

In my research for this project, I have stumbled on a few academic papers discussing sentiment analysis in general [6], as well as Twitter sentiment analysis [7]. But one of the most prominent related works is usually found within social media marketing services that provide sentiment analysis as part of their toolset. Most important of these is the Radian6 now rebranded as Social Studio a product by Salesforce Marketing Cloud [8]. Sentiment analysis is indeed part of their suite and it is offered to customers as a way to unlock insights and indeed as stated in the introduction, actionable data. Their service is very robust, but expensive. Other smaller SaaS providers can be found as well.

Approach

The development of the proposed solution was split into two parts: a backend server and an iOS client.

Backend Server

Sentiment Analysis

The primary objective was to be able to do sentiment analysis on a large volume of tweets in real-time. The main library used was Sentiment [12]. This library uses the aforementioned AFINN scored keyword list. The list contains around 3382 english words and is expanded from time to time to include more words. The score given for each word is ranked between -5 and +5, depending on their positivity or negativity in terms of sentimental meaning.

Word	Score	Sentiment
annoy	-2	Negative
happy	+3	Positive

Table 1 : Example of scores in the AFINN word list

To analyse a sentiment, the sentence or paragraph provided is traversed and each word is compared to the words in the scored list. If a match is found, the word and it's score are saved. A positive word will have a positive number, and a negative word will have a negative number. At the end, the final score of the provided sentence or paragraph is calculated, it being the sum of scores of the found words and the result is reached. Depending on if the final score is positive, negative or zero, the final sentiment is positive, negative or neutral respectively. A comparative value is calculated which is the ratio of scored words found to the total number of words in the

analysed text. There is an important language paradigm employed and that is negation. Words like “can’t”, “don’t”, and “isn’t”, etc… are taking into consideration when giving a word a score. An expression preceded by one of these words is usually given an inverted score, as in a positive sentiment becomes negative and vice versa. For example: *“I am happy”* has a final score of +3, showing a positive sentiment, but *“I am not happy”* has a final score of -3, a negative sentiment.

Finally, emojis are a way that users express themselves often on social media, and often carry with them a sentimental meaning. As such, the AFINN word list was augmented with a list of scored popular emojis, the list was built while using a pre-compiled list of scored emojis [13]. The list contains emojis written in text format using punctuation, as well as graphical emoticons.

Emoticon	Score	Sentiment
😠	-4	Negative
:$	-2	Negative
:)	+3	Positive
😄	+3	Positive

Table 2 : Example of scores in the emoticons list

Performance was an important aspect as the application needed to analyse at least 100 tweets per request. Performing a benchmark test on a text composed of 1434 words, resulted in being able to perform around 1145 operations per second.

```
→  sentiment git:(master) ✗ make benchmark
node ./test/benchmark/performance.js
sentiment (Latest) - Short  x 406,139 ops/sec ±1.96% (79 runs sampled)
sentiment (Latest) - Long   x 1,145 ops/sec ±1.63% (78 runs sampled)
```

Fig. 2: Sentiment Analysis Benchmark

API

With the main feature in place, an API was built that would be able to serve the iOS client with tweets themselves and the analysis scores. It was subdivided into 3 main functions: fetching trends and tweets from Twitter, analysing the sentiment, and compiling the sentiment data, all for use by the client. All the endpoints return a JSON response. The API was also versioned to allow for easier updates in the future without breaking existing functionality with clients. To fetch trends and tweets, the data was fetched directly from the Twitter API, using the Twit library [14] that acts as a simple function wrapper.

Endpoint: GET /trends

The server is able to fetch the top 50 trending hashtags/keywords on twitter at the time of the request, based on the location of the client as the topics would be relatively more relevant to the user. The location is provided as latitude and longitude parameters by the client.

Endpoint: GET /tweets

This is where the bulk of the sentiment analysis happens. Upon request of the client, who simply passes a query parameter indicating a hashtag or topic, the server is able to search based on the query parameter for up to 100 tweets at a time. The Twitter API allows the fetching of the latest, most popular or a mixed set of latest and popular tweets per request. The decision was made to use the mixed mode, as it gave a good sample or popular sentiment, while also providing newer tweets. Once 100 tweets are fetched, each tweet text has it's sentiment analysed. Each tweet is then saved to the database along with the sentiment analysis. A query object is created, where the tweets are collected under a specific keyword. This allows the API to also retrieve older tweets relevant to the query, and will allow for a larger sample to be built over time in order to assess the sentiment on a larger scope. Improvements could be made here, that will be discussed in the following section. The resulting collection of tweets and their scores is then sent back to the client. A complimentary endpoint (/update) was used to try and fetch newer tweets using the same mechanism described above.

Performance wise, and since the intention was real-time analysis, using JavaScript built-in timing function, each request to fetch 100 tweets, analyse the sentiment of each and return them to the client takes on average 180ms depending on network conditions. The result was satisfying, but further improvements could be implemented.

Endpoint: GET /stats

The final endpoint, in order to provide a data set in order to visualise on the client side, all the sentiments of any query are categorized as positive, negative or neutral along with their scores and counts, and returned to the client.

iOS Client

The intention with the client was to provide a user friendly and straightforward approach to find trends, search, view tweets and their sentiments and finally visualise the overall sentiment. A minimalist design approach was undertaken, as seen in the first screen (Fig. 3). Using the CoreLocation API available, the location of the user was captured and an immediate request to the API server was made passing the user's coordinates is made, retrieving an updated list of trending topics. In addition to that, it was imperative to provide the user with a way to make custom searches for any query they were interested in.

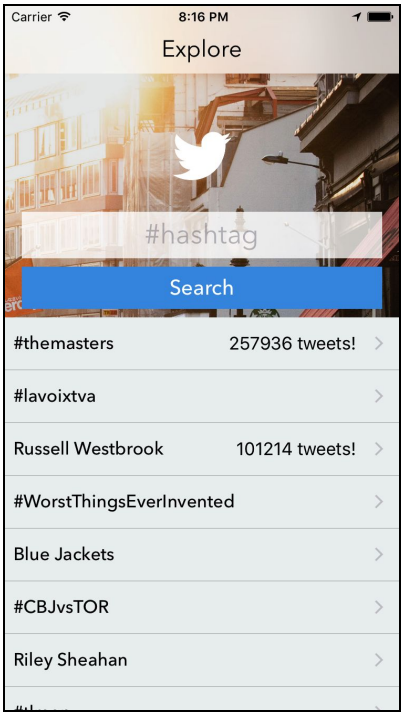

Fig. 3: First screen of the iOS client application

Upon selecting a trending topic or inputting a certain query and pressing the search button, a request is made to the API server requesting the tweets related to the query. A Twitter feed will appear (Fig. 4) not unlike the default Twitter feed, but on display will be a sentiment score associated with every tweet. Upon tapping on that tweet, the full sentiment analysis will be displayed, showing how the sentiment score was calculated.

The feed was built using a UICollectionView, but an important aspect of the feed is the fact that it does not request a huge amount of tweets for display purposes as pagination was implemented. The initial request is limited to only display 20 tweets even if the backend database has already hundreds of tweets, and as the user scrolls down, an additional 20 tweets are requested if available. This is made to keep the response size of the data small, boosting UI performance.

Finally, clicking on the Stats button in the top right corner is where the most important function of the application resides. This view requests the full data set of sentiment for a particular query, categorized into positive, negative and neutral data. Using the Charts library [15] a pie chart is then fed the data to produce an elegant visualisation of the data (Fig. 5). This view calculates the ratio of positive, negative and neutral words in all of the tweets and displays a section of the pie sized appropriately. This allows the user to get an immediate “big picture” understanding of the general sentiment towards a particular topic. Below the chart is an overview of the number of tweets that fit into each category.

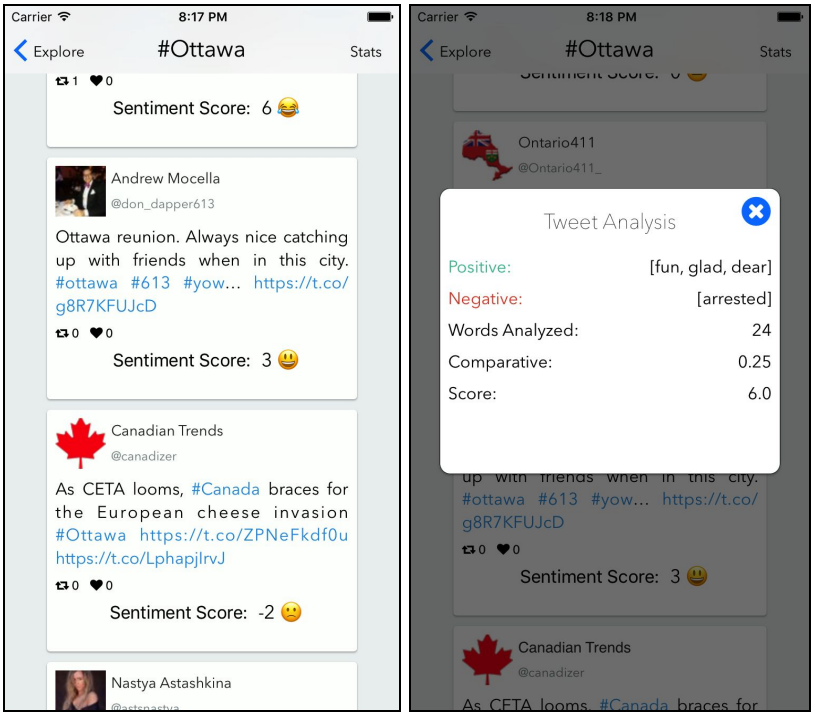

Fig. 4: Twitter feed with sentiment score and detail view

Discussion

Overall the proposed application achieved the goals and objectives set out initially. It is able to retrieve tweets related to a topic, analyse their sentiment, and visualise the result in a user friendly way, all in real-time. This would allow any interested party, such as marketing manager to understand the sentiment of Twitter users towards his brand, product or service, or even politicians towards their electorate regarding a particular topic for example. Decisions can be made based on actionable visible data.

There are of course a multitude of improvements and feature additions that can be added. The first improvement would be the integration of the API server with the Twitter streaming API to retrieve and analyse tweets as they are published by users. This feature would open up further analysis and visualisation capabilities such as charting sentiment over time as they are coming in. Using the geolocation tag of tweets would also allow the mapping of sentiment on a map. Improvements to both the API and the client would open up such possibilities.

The other facet of improvements would be on the sentiment analysis. Obviously with the way people communicate in 140 characters, the application sometimes fails to give an accurate sentiment for a tweet. So an additional larger set of data would ultimately provide more accurate results albeit at the cost of performance. A balance would have to be established. Additionally, the app does not know what the positive or negative sentiment is. Negative for example can range from mad to angry to sad to disappointed. The app currently does not categorize further than that.

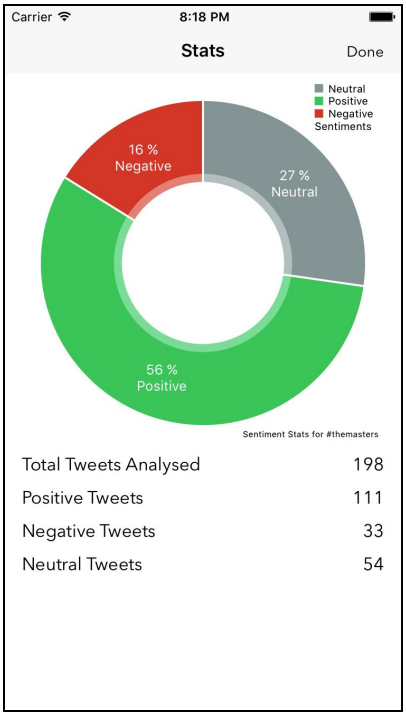

Fig. 5: Statistics view for a particular query

Conclusion

The Twitter Sentiment Analysis App is a great step forward into streamlining emotion and sentiment mining of a large user base that generated enormous amounts of content such as Twitter. Already it can be used to visualise sentiment of things around the world.

This project was a great learning experience to create a platform that can generate meaningful information that can be further used in developing recommendation and analytical systems.

In the future, as datasets grow and computing power increases, having the ability to use deep learning to perform tone analysis fully understand how a user is feeling would provide a much deeper insight. Already, IBM Watson [16] has the ability to perform such a task. Giving existing AI assistants such as Siri and Alexa the ability to understand our emotions opens up the path towards more sophisticated recommendation systems. Furthermore, associating user profiles with their actions and sentiments can further branch out into medical fields such as psychiatry.

References

[1] https://en.oxforddictionaries.com/definition/twittersphere
[2] https://www.statista.com/statistics/303681/twitter-users-worldwide/
[3] https://en.wikipedia.org/wiki/Sentiment_analysis
[4] http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010
[5] http://varianceexplained.org/r/yelp-sentiment/
[6] https://arxiv.org/abs/cs/0212032
[7] https://arxiv.org/abs/1103.2903
[8] https://www.marketingcloud.com/products/social-media-marketing/radian6/
[9] https://en.wikipedia.org/wiki/Node.js
[10] https://en.wikipedia.org/wiki/Express.js
[11] https://en.wikipedia.org/wiki/MongoDB
[12] https://github.com/thisandagain/sentiment
[13] https://github.com/dematerializer/emoji-sentiment
[14] https://github.com/ttezel/twit
[15] https://github.com/danielgindi/Charts
[16] https://www.ibm.com/watson/developercloud/tone-analyzer.html