Project Report

Data Mining: Tools and Techniques

Social Media Sentiment Analysis and Opinion Mining

**July 27, 2016**

## Nupoor Paneliya, 1501756

## Chinmay Modi, 1509628

**Table of Contents:**

1. Purpose of the project
2. Expected result of the project
3. Approach taken in the project
4. Classifier Performance
5. Results
6. Similar classifier and their accuracy
7. Future work
8. Code

**Purpose of the project:**

The project aims to collect data from social media, specifically Twitter, and analyze the text to determine the sentiment present in it, as well as obtain a clear picture of a political opinion it may present. We aim to data mine and present certain data about the upcoming 2016 USA presidential elections

**Expected Result of the Project:**

At the end of the project, we aim to have a working code suite that performs data collection, data cleaning, data organization, along with an analyzer tool and a statistics generation tool that is able to collect tweets about politics and the elections, and analyze them to determine a multitude of results. Specifically, we hope to achieve the following outcomes:

1. How much support a candidate has on Twitter, which translates into how much support they have in a significant demographic.
2. Ascertain and quantify a candidate’s positions on various policies and opinions on issues based on their own tweets.
3. How much do people agree with various policies of candidates, like economic policies, education policies, foreign policy, military spending, etc.
4. Analyze any given user’s tweets to obtain a clear picture of where they stand on individual issues, then as a whole, match them to a candidate and thus predict which candidate a user may support based on their non-deterministic tweets.

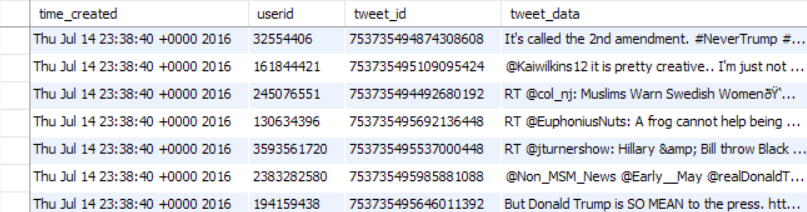
**Approach taken in the project:**

*Data Collection:* We have collected tweets using Twitter streaming API based on certain keywords and hashtags pertaining to politics in general, and the elections and candidates in particular. Thus, we do not need to eliminate a large amount of useless data.

This Twitter API gives data in JSON format. Please refer code in “StreamCollector” python file which collect the data.

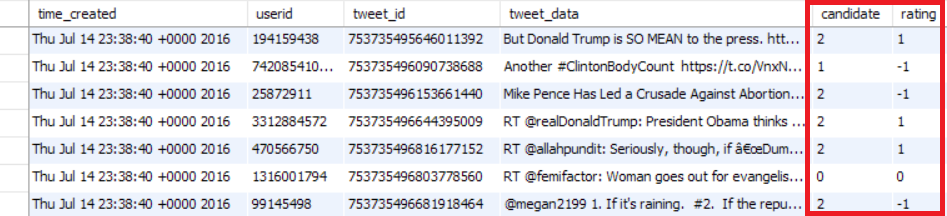
Tweets collected: 143020

*Data Pre-processing:* Data collected is in JSON file and this data contains many unnecessary attributes which are not require for this mining. So we clean up the collected data and integrate it into MySQL database.



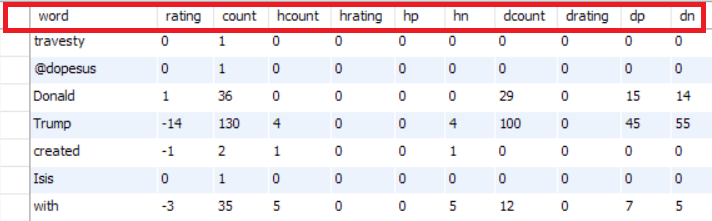
*Prepare training dataset:* After gathering relevant and needed tweets, we create training set by manually rating some tweets. We have rated 1 for Hillary and 2 for Donald make it +ve for positive tweets and –ve for negative tweets.

Please refer code in “training\_set” file.



*Tweets Tokenization:* We need content of the tweets which is embedded in text. To start our analysis we need to break the text down into words. Purpose of tokenization is to split a stream of text into smaller units called tokens, usually words or phrases. Each word is given positive or negative rating and counter is maintained to count its frequency.

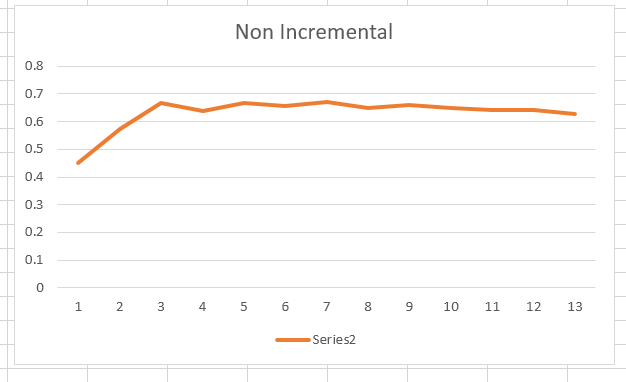
Please refer python file “classifier\_train” that tokenize the tweets.



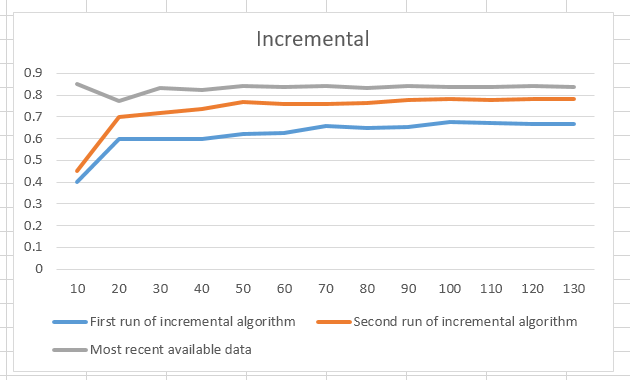
*Building Classifier:* The classification method we use is based on text emotion analysis i.e. predicting what a textual data is about, based on ratings of individual words or tokens. Each word has an impact of a neutral rating, as well as applying to both candidates. We can then determine which candidate the tweet is about, as well as whether it is positive or negative, based on the word composition in the tweet. The algorithm is self-healing in nature, since a word that may have been biased in the training set may still be normalized over time if it appears in conjunction with other neutral words in neutral tweets. Tweets are rated if they have at least 60% of their words already classified. This allows us to rate words that are missing from the DB, on the fly. Thus, the classifier has an increasing accuracy as it analyzes more and more tweet data.

Please refer python file “classifier\_final”.

**Classifier Performance:**

****

The above figure shows the classifier accuracy without implementing the self-healing part. Over time, the accuracy of the classifier tends to stabilize in the 60-65% range. While this seems acceptable, the performance is kicked up a notch when we add the self-healing component.



This is where the program really shines. In the very first run on the same data as the previous run, the accuracy is already slightly higher than the non-incremental version, stabilizing in the 65-70% range.

Running the code on the same data shows remarkable improvements. The accuracy rate goes up to almost 78%, and what is more important, it is now able to classify tweets which were initially ignored due to not fulfilling the criteria.

Lastly, we let the code run on about 10k tweets worth of data, then tested it against a set which was almost 1 month newer then the training set. We wanted to check how well the classifier holds up against new data which has a very different feel to it compared to previous data. Specifically, there was a lot of political turmoil and unrest between the two data sets, including addition of a number of frequently used terms, and we wanted to see if we could work with it.

Surprisingly enough, the code managed to hit an accuracy rate of almost 83%. What is more shocking though, is that there were certain keywords and tweets which were absolutely missing from the training set, and yet the program correctly classified them.

**Results:**

The tweets show two main results:

First, the support for Hillary Clinton is usually subdued, and tends to hover around smaller absolute values. In contrast, the opinion on Donald Trump is very polarized, with most tweets about him having high absolute scores.

Second, Hillary’s support has gone down significantly in the time period of our data collection. At the beginning of our project, when Bernie Sanders was still in the race, Hillary had significantly more positive ratings, and held a lead over Donald. But after Bernie bowed out of the race, and after DNC leaks were revealed by WikiLeaks, Hillary’s support has gone down a lot, to the point of falling below Donald.

Our results are consistent with the media portrayal of the candidates, including the trends of support for both.

If this data is an indicator of the elections, we can predict that while Donald is a lot more controversial than Hillary, he may still win the election solely based on the absolute amount of negativity surrounding Hillary.

**Similar Classifier and Accuracy:**

1) http://www.laurentluce.com/posts/twitter-sentiment-analysis-using-python-and-nltk/

Accuracy is 0.8 with handpicked tweets on generic data

2) http://cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf

0.65 with Natural language processing with keywords based analysis

0.8 with Bayes

3) http://crowdsourcing-class.org/assignments/downloads/pak-paroubek.pdf

0.6 - 0.8 based on method of analysis

4) http://www.cs.columbia.edu/~julia/papers/Agarwaletal11.pdf

60.83 is highest accuracy of multiple methods

5) http://blog.datumbox.com/how-to-build-your-own-twitter-sentiment-analysis-tool/

83.26 highest accuracy

6) https://arxiv.org/ftp/arxiv/papers/1509/1509.04219.pdf

Approx. 80% accuracy

7) http://www.aclweb.org/anthology/W13-1106

58.96% highest accuracy with a combination of tools

**Future Work:**

Improve accuracy of classifier, by using better word analysis techniques, more focus on word position in sentence, bigrams and trigrams as well as phrases.

Improve parameters of classifier for better accuracy.

Lastly, try to streamline the process for live performance. We may be able to implement a web based service that provides live updates on both candidates’ support, along with geographical distribution of the support as well.

**Code:**

All the source code is available in files along with this report.

Explanation of each file:

StreamCollector.py

This file connects to the twitter API and allows us to download live tweets. Then, we select those containing relevant keywords to us, and store them in a MySQL database as well as a text file. Twitter data is in json format, and we extract only those features we feel are relevant, namely time, userid, tweet id as well as the actual tweet data itself. This allows us to save a lot of space.

Training\_set.py

Next we run this file. Its purpose is to create a training data set from our collected tweets. It presents tweets to us, takes positivity and candidate alignment ratings from us, and proceeds to store them in the DB.

Classifier.py

This file is slightly misnamed. All it does is properly tokenize the tweet data, strip useless information from it, and ignore words of size below 3. It is included as a header or import file in the next two programs.

Classifier\_train.py

This file builds the word rating database from tweets belonging to the training set. Pretty straightforward work, with a slightly complex system I built myself.

Classifier\_final.py

The culmination of our project. It reads tweets not yet rated, then tokenizes them, finds individual word ratings and assigns them a rating,

The reason this works so well is because it only chooses to rate tweets who have at least 60% of their tokens already present in the word rating table. And more importantly, the words in such tweets that were not previously rated are initialized and assigned a rating as well. Thus, it continues to learn ratings for new tokens even after training is complete.

Currently we have not implemented adding these ratings back to the database due to lack of time, but it does give the rating in stdio, which allows us to rate its accuracy. Based on this rating, it also prints out the accuracy of the program. Note that our rating is only used to find accuracy, and if I am rating a tweet as incorrectly classified, there is no impact on the database from it.