Product Appreciation Analysis on Twitter via Sentiment Analysis

Ali Uğur Aker
03645459
a.uguraker@gmail.com

Eralp Bayraktar
03669101
bayraktar.eralp@gmail.com

Ayhun Tekat
03669101
ayhuntekat@gmail.com

ABSTRACT
In this document, we describe our approach of using sentimental analysis on massive Twitter data to extract information about a product's success from the customer's viewpoint. Several approaches such as Naive Bayes and emoticon analysis are explained. The results obtained from these approaches and how they compare to the results obtained by using TextBlob are explained (TextBlob is a library that is considered successful in terms of conducting sentiment analysis on text to determine whether it is positive, negative or neutral). Our domain is the iPhone product and dimensions that we processed are battery, screen and camera.

Keywords
big-data; sentimental analysis; twitter; opinion mining; data mining;

1. INTRODUCTION
We aimed to build a product which will also have a real-world value. This project will enable marketers to analyze the perceived value of their products by processing publicly available Twitter data. The document is split into three sections, Implementation, Results and Conclusion.

Implementation section starts with explaining the tools that were utilized in the development of this project. It continues with the explanation of the TweetCrawler program and how it works and ends after an in depth explanation of our sentimental analysis approach. Results part tries to analyze and explain the results and the document ends with a conclusion section that sums it all up.

2. IMPLEMENTATION

2.1 Tools
We utilised two major libraries for this project, which are TextBlob and Apache Commons Compress.

2.1.1 Apache Commons Compress
After struggling to unzip and filter the data on the cloud as well as on our systems, we found out it wasn't feasible to progress in such manner. We searched for a better solution and found Apache Commons Compress. ACC is a Java library, which is currently at release 1.10 and supports bzip2, Pack200, XZ, gzip, lzma and Z formats. We utilised Apache Commons Compress to decompress in-memory and filter simultaneously, thus eliminating the need for disk space.

2.1.2 TextBlob
TextBlob is a Python library, which provides basic infrastructure to start Natural Language Processing (NLP) and in our case Sentiment Analysis. We used TextBlob's Sentiment Analysis as our reference point and utilized the available Naive Bayes classification algorithm to further improvements.

We also utilized this library to filter tweets which are not English (using language detection) and fix typos to decrease the noise in data.

2.2 TweetCrawler
We were faced with the challenge of processing hundreds of gigabytes of compressed Twitter data. Because the data being in text format, extracting it would take up to 2-3 times the disk space. For that reason we looked for ways of processing that data without extracting it beforehand. After some research we found Apache Commons Compress java library that allows reading from a compressed file without extracting it.

At this point we had to make some tradeoffs between space and performance. First option was to extract every compressed file with a script and then run a c or c++ program on the extracted text files. This option offers a fast running time in the expense of a lot of disk space. Second option was to use Apache Commons Compress library and read the files without extracting them. This option would not use any extra space however it would hinder the performance because it required using java rather than languages like c/c++ or python. We went with the second option because the task was parallelizable and any performance loss introduced by using Java language can be compensated by utilizing more processing cores.

Description of input-output behaviour of TweetCrawler:

Inputs

- Path to the folder that contains the compressed files.
- Product name //name of the product e.g. iPhone.
- Keywords //traits of the product to be analysed e.g. screen, battery.
- Forbidden words //words that will cause the tweet to be ignored.
- Number of CPUs to be used //Equal to number of cores if set to 0.

Outputs

- JSON files that contain the relevant tweets (and only relevant features of those tweets such as its content, timestamp etc.). Files are named after the keywords e.g. screen.json battery.json etc. Example line in a JSON file:

```
{"text":"I was going to buy the
iPhone 6 Plus, but I already have a
flat-screen
TV.","timestamp_ms":"1411278274662","
created_at":"Sun Sep 21 05:44:34
+0000
2014","id_str":"513564422963347458","
lang":"en"}
```

- err.out file. Has the description of any error that might occur during runtime. This output file usually has something in it if there are corrupted files in the input.
- finished.txt file contains number of files that are processed, start time, end time.

- Console output. The program constantly outputs its progress to the console. Example output is shown in Figure 1.

```
Counting number of files to be processed...
Number of bz2 files in folder:33776
Thread 6    did tweets/10/26/01/42.json.bz2 -- 0.0% complete
Thread 0    did tweets/10/26/01/15.json.bz2 -- 0.0029606821411653247% complete
Thread 7    did tweets/10/26/01/14.json.bz2 -- 0.005921364282330649 5% complete
Thread 4    did tweets/10/26/01/08.json.bz2 -- 0.008882046423495974% complete
Thread 3    did tweets/10/26/01/48.json.bz2 -- 0.011842728564661299% complete
Thread 1    did tweets/10/26/01/34.json.bz2 -- 0.014803410705826624% complete
Thread 2    did tweets/10/26/01/10.json.bz2 -- 0.017764092846991947% complete
Thread 6    did tweets/10/26/01/21.json.bz2 -- 0.02368545712932259 8% complete
```

Figure 1.Console Output

2.3 Sentiment Analysis

To have most precise results, we combined two sentiment analysis approached together in this project. First approach is widely used and known Naive Bayes Analysis and second one is Emoticon Analysis, which is specifically developed by our team for our analysis domain.

2.3.1 Naive Bayes Analysis

Naive Bayes approach ranks words as positive and negative using a training set, which is formatted as a JSON Array in the project. According to rank of individual words, overall rank of the sentence is determined and sentence is ranked as positive and negative in the end. Ranks of individual words has weights, which enables algorithm to bring ranks of words together to understand the meaning of sentence.

```
{"text":"the iphone 6s battery is so good", "label":"pos"},
{"text":"Still waiting for the day when iPhone battery life is decent ", "label":"neg"},
{"text":"iPhone battery life is garbage.", "label":"neg"},
{"text":"iPhone.Its easy to use and has a good battery life.", "label":"pos"},
{"text":"Battery life is crappy", "label":"neg"},
{"text":"Battery life is bad", "label":"neg"},
```

Figure 2.Training Set

We can understand training mechanism using figure 2. Using fifth and sixth entry, algorithm gives words “Battery”, “life”, “is”, “crappy” and “bad” negative ranks. But words that we want to actually rank, are only “crappy” and “bad”. But from fourth entry, words “battery” and “life” gets positive ranks. Thus Naive Bayes understands that these words are not strongly negative or positive, but is still assigns them a rank with a lower weight. We tried to balance our training set such that neutral words do not affect the algorithm significantly but removing them may improve the performance slightly.

First approach of our team was using a training set, which is available online but, Naive Bayes training sets has domains and using a set for another domain will not provide correct results. We made some tests using a set, which is prepared for movies. The size of set was thousand times larger than the set we used in our project but, Naive Bayes failed even in easiest tweets such as “I love my iPhone's battery.”

The only solution was preparing a set, which is specifically focused on our domain. Since selecting appropriate tweets one by one is a long process, we prepared a script, which speeds up collecting tweets for training set. In traverses the JSON file which contains tweets and displays them one by one. If user finds tweet not suitable for training set, he may press “s” and script will move on to the next tweet. If user presses “p”, application adds tweet to training set with positive rank, else if user presses “n”, tweet will be added as a negative tweet.

2.3.2 Emoticon Analysis

Besides using Naive Bayes approach with a custom training set we introduced a whole new sentiment analysis approach: Emoticon analysis. After spending some time with the thousands of tweets that we downloaded, we figured out that almost a quarter of tweets contains emoticons. Also since our data source is a social media channel, we can assume that this situation is valid not only for our data set.

Analysis works as following. We ranked emoticons in figure 3. as negative and figure 4. as negative. Tweets are ranked by appearance of negative and positive tweets.

Figure 3. Positive Emoticons

Figure 4. Negative Emoticons

The only gap in Emoticon Analysis is two sentences, which are used consecutively. An example may be “I hate my iPhones battery. Getting a Samsung Galaxy soon :)”. This tweet will be ranked by positive by our approach but since our domain is iPhone, this tweet is negative. After some observations with data set, we saw that this situation is not very common and can be neglected.

Emoticons are perfect way for expressing feelings and opinions and we will discuss more about its benefits in results section.

2.3.3 Naive Bayes and Emoticons Together

Naive Bayes and Emoticons are combined in our project to get the most precise results. First we run Emoticon Analysis. If there are more positive emoticons than negative emoticons, the tweet is ranked as positive immediately.

If there are more negative emoticons than positive emoticons, the tweet is ranked as negative immediately. If number of positive and negative emoticons are equal, then we have two options. Either no emoticons are used, or user have mixed feeling about the product. In both situations, we execute Naive Bayes Analysis and result of Naive Bayes analysis will be final.

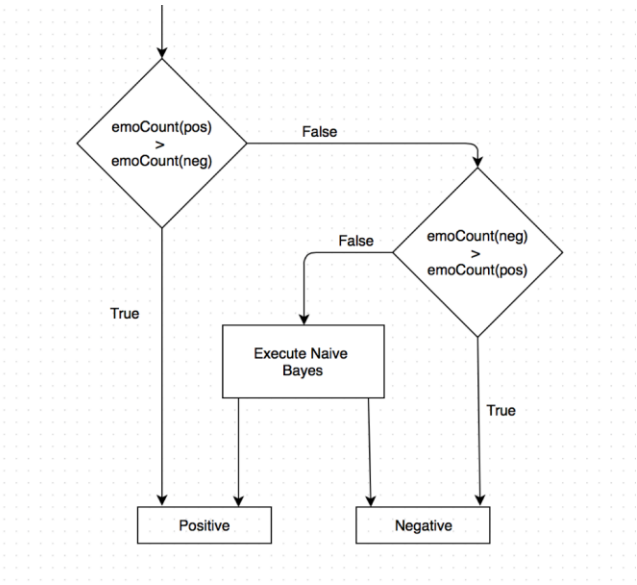

Figure 5. Overall Execution Diagram

3. RESULTS

To recap, in this project our goal was to convert social media entries to quantifiable and measurable customer feedback using existing technologies and then further improving the available algorithm. This means we have two kinds of results for our project.

1) Benchmark of the improved algorithm with the available one
2) Quantifiable and measurable customer feedback

Benchmark of the improved algorithm with the available one

Best way to benchmark such a classification algorithm is to chart the false-negative (type-II) and false-positive (type-I) rates of the new algorithm. To do this kind of testing one needs to know the real positives and real negatives, unfortunately we didn't have access to resources to create real data, instead we created pseudo-real data with TextBlob's Sentiment Analysis and benchmarked against it. We ran the main algorithm to create Positive and Negative buckets, then we ran our improved algorithms separately for these buckets and plotted the following chart.

	TextBlob Positive	TextBlob Negative
Bayesian Positive	72.98%	15.70% (false positive)
Bayesian Negative	27.02% (false negative)	84.30%

Figure 6.

Then we did the same thing for the Emoticon Analysis Add-on.

	TextBlob Positive	TextBlob Negative
Emoticon Positive	78.00%	10.72% (false positive)
Emoticon Negative	22.00% (false negative)	89.28%

Figure 7.

Here we can see that adding Emoticon Add-on had a huge improvement of decreasing the false negative as well as false positive rates! Generally these two values are inversely related.

Quantifiable and measurable customer feedback

To begin with, the metric we use, which is percentage of positive tweets, is not equal to percentage of satisfied customers. Our result is not percentage of satisfied/happy customers, but there is surely a correlation between these two values, thus the metric we extract is useful to analyze trends. A customer may also commend as well as criticize the same product regarding different dimensions, i.e. good screen but bad battery life. This is why we divided tweets into dimension buckets, it guides companies to what to improve.

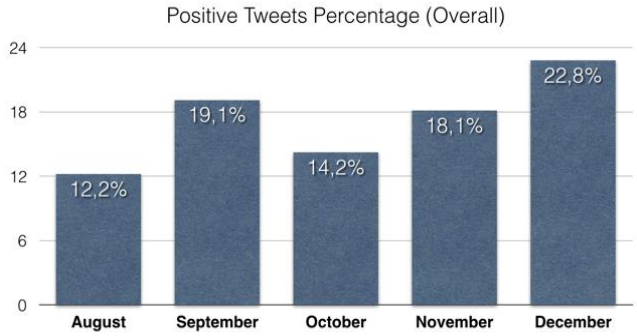

Figure 8.

A) The data belong to 2014 so this graph is about the release of iPhone 5. This graph shows the percentage of positive tweets without considering any dimensions. We can see on August people have been using their iPhone 4S for about a year and they were bored about it, so the situation is mostly negative with degrading battery lives etc.

B) On September the new iPhone 5 was announced which was the first iPhone with a bigger screen, which made a big buzz (which we will see in the next graph). People were also commending the design and everything, because it was new :)

C) On October the iPhone 5 is released for sale and the first people to get it were mostly tech people and geeks, who have high expectations and who are better than average person to find things to criticize. People also waited in line or had to travel across multiple cities/shops to get their iPhone 5, which obviously increases expectance. This high expectancy dropped the percentage of positive tweets as we can see in the graph.

D) On November & December people with less expectancies were getting their iPhone 5's and also during Christmas period people were getting iPhone 5's as presents, which psychologically increases happiness and people are more optimistic about presents. This results in high percentage of positive tweets.

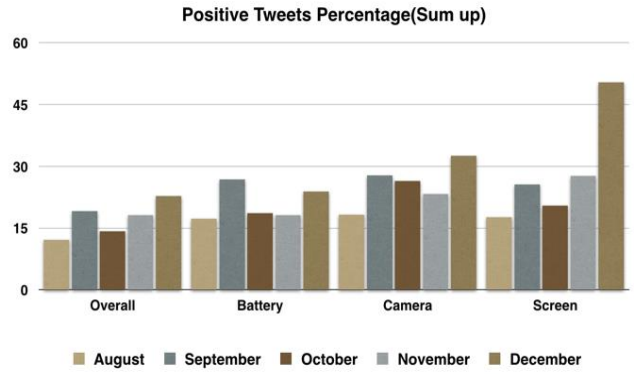

Figure 9.

We can see the trend explained in the previous graph for each dimension (consecutive group of bars). What's interesting here is the positive appreciation the screen dimension has gained! We can see people are satisfied with the camera, and that the battery is dragging the other dimensions down, but screen is a huge improvement because iPhone 5 is the first iPhone with a bigger screen.

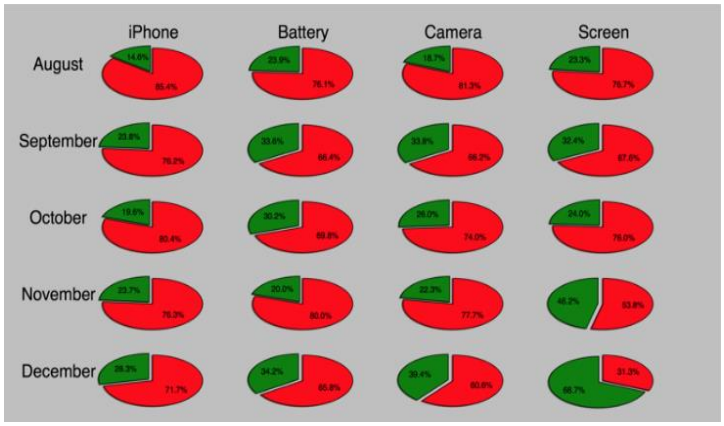

Figure 10. Pie-chart representation of Figure 9.

Based on this graph we could say Apple should produce iPhones with bigger screens, and no surprise Apple released iPhone 6 with bigger and iPhone 6+ with even bigger screens. We can see that's what people wanted looking at this graph.

4. CONCLUSION

There are many sentiment analysis libraries online including our reference analysis algorithm, textblob. But these approaches try to understand feelings mostly, which may fail in some domains. Product reviewing has a slightly different language than expressing regular feelings.

In this project we tried to implement a sentiment analysis approach for a specific domain and tried to create a value for marketing departments. Also, we tried to do this as efficient as possible. Using thousands of sentences in a training set might provide more reliable results but computation delays should be also considered.

Languages evolve by time and with emergence of social media and messaging applications, emoticons become a part of our language and by introducing Emoticon Analysis, we tried to respond to this change.