Finding Our Roots: A Hybrid Approach to Word Segmentation

Nick Schultz
Swarthmore College '11
nschult1@swarthmore.edu

Kevin Pytlar
Swarthmore College '12
kpytlar1@swarthmore.edu

Abstract

Product performance in the market is a topic heavily researched by analysts and anyone else interested in the performance of companies and/or the economy. Product sentiment is a main influencing factor in the performance of the product in the market place. The following paper will describe techniques used for analyzing the sentiment of Twitter users toward different products.

1 Introduction

Sentiment analysis can be heavily quantitative, which we will cover in more detail in the methods section of the paper. Quantitative trading techniques are gaining popularity amongst banks and traders. However, these quantitative techniques can be based purely on the movement of the market, which raises question as to how much the buying and selling reflects actual market demand. Twitter can be used to collect data which directly reflects the market because actual people are tweeting their opinions about market products. Thus, successfully analyzing the sentiment of tweets can lead to actual quantitative demand and supply information that truly reflects the sentiment of the market. (market?)Economic analysis using tweets as a data set can result in more realistic trading practices because it bases the practices on actual demand statistics. This will help the market reflect a rational market (rather than what?) as it is supposed to.

In a broader sense, sentiment analysis methods can be applied to any data set and can gather different sets of quantitative information which can be used to assess and analyze very different topics. A more efficient sentiment analyzer can help push the effectiveness of natural language processing techniques, and further help in the effort to understand language through the use of computer algorithms. When interacting with other people, it is obviously important to understand the opinion or sentiment of what is being discussed . If we can teach computers to understand the sentiment or opinion of people, we can start to understand the overall sentiment of large sets of communication data, because of the capabilities of computers to process large amounts of data so quickly. Understanding the sentiment of a crowd can help to improve the quality of service in a vast amount of fields.

We used Twitter to collect data on smart phones, specifically the iPhone 4, Evo 4G and Windows Phone. Twitters API is free, so we were easily able to connect to their servers and search for the products we selected. We ran searches for 1 month, compiling millions of tweets from many different users about the same topics. Each tweet could contain any one or all of the search topics. Using these tweets, we can use different sentiment analysis techniques to quantify the sentiment towards the product.

Our sentiment analysis was semi-supervised. Completely supervised methods consist of hand-making or using an already made list of sentiment words and sentiment scores. Finding the sentiment of a tweet can be as simple as finding words in tweets that are in the sentiment list and adding to-

gether the sentiment scores of the words that are found in the tweet. If the final score is positive, positive sentiment is expected, if the score is negative, negative sentiment is expected. The implementation section describes how our program is semi-supervised and how it analyzes the data which it collects. Additionally to describing our program and the results, we will connect our project with other sentiment analysis papers. For example, how we could incorporate other methods described in other papers to continue to our sentiment analyzer. We will also discuss how our methods build on the methods in other papers. In our evaluation section, the paper will discuss what sort of experiments we ran, what data we got from those experiments and how that data shows the the program's skill at classifying sentiment in tweets.

2 Methods

The main goal of our project was to create an algorithm that learned the vocabulary of the positive and negative words in the corpus it was trained on. We did this in two steps, first aggregating a score for each word in the corpus, then deciding which of those words to add to our improved sentiment list.

2.1 Original Sentiment List

First we built a small sentiment list of words with sentiment scores. The scores are positive or negative, and scored 1-3 based on how positive or negative we thought the word is. The list is made up of language that is clearly positive and negative, there are no words that are neutral or could possibly go either way. We also tried to use words for this initial sentiment list that would be used in a medium like Twitter, and that had clear meaning in that context. The list we created was 27 words long and had similar total sentiment scores for positive and negative feeling.

2.2 Data to Analyze

We collected millions of tweets, in English, which are about smart phones. We can do this by using the python Twitter API, specifically the search() method. In the search method we were able to specify the language to English and the subject of the tweets we want. As mentioned in the introduction, we were searching for tweets relating to the iPhone 4, Evo 4G and Windows Phone. Because Twitter is such an open forum, there are a lot of spam messages, especially messages that pertain to consumer electronics. As such, we worked to eliminate duplicate messages, because they were most likely to be spam. We also cut out usertags (i.e. @twitteruser) and urls from the tweets, as those would have no sentiment attached to them. After all of our parsing we cut the 10 million tweets we mined from Twitter down to around 250,000 unique tweets.

2.3 Score Aggregation

To get new scores for words in our tweet corpus, we look for currently scored words that occur in the same tweet. We sum up the total sentiment of words in a tweet, using sentiment provided by the sentiment list. If a given word is not in the sentiment list, it contributes 0 to the sum. Once that sum is computed, each word adds it to the words running total of scores of tweets it is in. We also keep track of how many times the word shows up and how many tweets it appears in.

2.4 Sentiment Weight Algorithm/Adding to sentiment list

Once we have the set of words and the scores associated with them, we run the part of our algorithm that improves our sentiment word list. We have two thresholds which we use to determine whether a given word should be added to our improved sentiment list. If a word is selected to be added to the sentiment list, the value it uses is the average score per instance of the word. The first threshold we look at is the percentage of tweets the word is used in. We want to have a lower bound on this percentage because if words which were just in one tweet were allowed into the sentiment analyzer, the score it would be added with would not be averaged over a large number of appearances, so the score would skew the sentiment list.

The second threshold eliminates words that are not strongly correlated with one sentiment or the other. We look at the list of words and scores that got through the percentage of tweets threshold and take a given percentage of them off the top. The idea behind this is to only add the most polarizing words, and eliminate the words that can be used in either sense. Words that were in the previous sentiment list

are not given any sort of advantage over new words to be added, besides the fact that the words already in the sentiment list are guaranteed to have at least their sentiment score in the next round.

2.5 Scoring Tweets

Once we have a comprehensive sentiment list, created using several iterations of the bootstrapping algorithm described in the two previous sections, we are ready to score some tweets. To score tweets we count up sentiment based on the final sentiment list we have. The scorer looks a lot like the score aggregation algorithm, with a few added features. We can score tweets by looking at each word in a tweet; if the word is also in our sentiment list we can add the sentiment score of that word to a total sentiment score for the entire tweet. We added a negation scoring system. If we find a word in a tweet that is also in our sentiment list, before adding the sentiment score of the word to the score of the tweet, we look to see if the word 'not' is one or two words before the current word in the sentence. If 'not' is there, we switch the polarity of the sentiment score of the current word then add it to the score of the tweet.

We have also implemented a word enhancer scoring method. If we use this in scoring a tweet, we look for enhancing words such as 'so' or 'incredibly' one place before the current word in the tweet which was found in our sentiment list. If we find an enhancing word, we multiply the sentiment score of the current word by a weight and then add the weighted sentiment score of the current word to the sentiment of the entire tweet. Like our original sentiment list, this enhancer list was created by hand. If we find a negation then an enhancer, for example, 'not so great', we do not use the enhancer. This is because instead of enhancing the emphasis on great, not so decreases the sentiment score of great. To avoid this, we do not look for enhancing words where negation is also found.

3 Related Work

4 Evaluation

To evaluate our system, we created a gold standard. To create the standard we hand-scored all of the words in 50 tweets with either a positive, negative or 0. Because of the normalization of the sentiment values in our sentiment list improvement, most of our sentiment values were around 1, so the difference in magnitude was not a factor. After we hand scored our gold standard, we trained a sentiment list on all of the other tweets for each variation of our algorithm based on the thresholds and number of times run through the sentiment list improvement algorithm. The values we tested on can be seen in table 4.

Factor	Low	High	Step
Top Percent	10	30	10
Percentage of Tweets	.01	.1	.02
Number of Iterations	1	5	1

5 Analysis

6 Future Work

7 Conclusions