Market Prediction Using Twitter Sentiment: A Text Mining Approach

Nicholas L. Brown

Syracuse University

Abstract

This paper considers the problem of effectively gauging market reaction to public sentiment. Using Twitter and text mining sentiment classification methods I sought to find a relationship between tweets about a company and its market performance. Multinomial Naive Bayes proved to be effective at sentiment classification however the full capability of the model was limited by the quality of the training data. For the final analysis simple linear regression was conducted on the daily sentiment score and two market indicators, daily return and closing stock price. No trends were readily apparent in the data I sampled however I believe more research could yield interesting and strong results.

Market Prediction Using Twitter Sentiment: A Text Mining Approach

With the rise of social media and the internet, unfiltered information is constantly disseminated instantaneously from platforms like Twitter. On multiple occasions tweets have directly affected a company's share price[1] though these cases have been limited to influential tweeters it proves that a company's share price can be directly affected by sentiment expressed over social media.

Data Preparation and Collection

Good data was one of the most pivotal pieces of this research project. The process of collecting, cleaning, and annotating the data is described below.

Data Collection

There were three main sets of data which were collected to conduct the analysis: bulk tweets annotated for sentiment, tweets about a particular public company, and financial data about that same public company.

Twitter Sentiment Data. Used to train the sentiment prediction model, annotated twitter data was highly difficult to acquire due to the terms of service associated with the twitter API. A dataset of 1.4 million tweets was provided by sentiment140, a project which specializes in analyzing sentiment for brands and products. Because of the implications further on in the analysis it was important to understand the process the creators of the dataset used to annotate the tweets. Rather than human annotation the set was created by detecting the

[1] On February 21, 2018 media influencer Kylie Jenner tweeted that she no longer used the application Snapchat, the primary product of Snap Inc. Her tweet sent the stock into a 6% dip wiping out $1.3 billion in market cap.

presence of happy or sad emoticons, this caused the dataset to be absent of neutral values with only a 4 (representing positive) or a 0 (representing negative).

Twitter Financial Data. Used to conduct analysis this data was requested through the twitter premium API. Because of the limitations imposed on academic access to the API I was limited to collecting the past 30 days of tweets. The methodology for collecting the tweets is as follows: 100 tweets were collected for each day in a 30-day window. Tweets were requested by searching the desired stock ticker, the 'or' keyword, and the full company name spelled out. Limitations in this collection method were apparent from the start with twitter search returning several unrelated results depending on the company.

Financial Data. Daily financial data was also gathered and reduced to the same 30-day window of the tweets. Using the yahoo finance API financial data included closing price and daily return of the stock was gathered for conducting the final analysis.

Data Cleaning

Because of the limited search features of the twitter API and the nature of tweets themselves, data had to be extensively processed and transformed. After acquiring the tweets from the API, they were processed in two steps.

Regex Cleanse. A simple regex pattern was used to parse the individual tweets removing any non-English letter leaving only the tweets words with no punctuation or emoticons. The exact regex pattern is below:

```
'[^a-zA-Z ]'
```

Topic Cleanse. After disappointing initial results, I analyzed the tweets given to me by the API, I noticed that twitter search was unreliable and made mistakes which I could not readily explain (returning entire tweets in foreign encoded languages). Because of this I decided to implement a simple lexicon approach whereby tweets were later searched again to ensure they matched at least one of the key words related to the company. Though this approach is still not smart enough to differentiate between homonyms, e.g. Tesla as a vehicle or inventor, it represents a massive improvement in data quality for the final analysis.

Model Evaluation

At the center of this project was the Naïve Bayes sentiment classification model, tuned to achieve maximum accuracy and speed. Though I originally chose to work with both linear support vector machines and Naïve Bayes the former had to be abandoned primarily because of low performance in preliminary hold-out tests. Additionally, support vector machine required higher processing and time requirements involved in creating the model. Vectorization and tokenization settings were modified to find the most accurate model. The process of vectorizing and evaluating the full model is detailed below:

Vectorization Settings

Several vectorization settings were experimented with to produce the most successful model. The final form of the model used the following vectorization settings:

- TFIDF weighting (to emphasize rarer words in the mass of hashtags and slang)
- Unigram (bi-gram and tri-gram yielded no noticeable improvement)

- Doc_freq = 5 (set minimum document frequency to 5, hopefully hides usernames and other proper nouns we don't want)
- Stopwords = 'English' (a list of stop words provided by sklearn retrieved from the Glasgow Information Retrieval Group.[2])

Trained Model Evaluation

The completed model was evaluated in a three-step process. First, accuracy scores were predicted using hold-out test (1/3 split) and a 3-fold cross validation scheme. Second, log ratios were calculated for the models most negative and most positive features. Finally, real twitter data about companies was annotated by hand after being fed to the model for sentiment scoring, allowing me to gauge the performance of the model with production data.

Predicted Accuracy. Hold-out tests were conducted using a 33% split of testing and 66% of training data leaving the training dataset with approximately 1 million observations evenly split with ~500,000 negative and ~500,000 positive. Given the binary classification task of assigning a 0 or 4 this model had a baseline accuracy of 50%. The most successful hold-out test predicted model accuracy at 77.29%. Additional cross-validation tests were conducted using the same data and model settings resulting in a 77.38% predicted accuracy. The confusion matrix below shows the prediction results of the holdout test and the precision and recall of the model. Interestingly both precision and recall were similar being differentiated by just 1% in all cases. This leads me to believe that the model is making mistakes recognizing both false positives and true positives.

[2] According to the source code for scikit-learn on GitHub the list contains about 300 words total.

P ->	0	4
0	202845	61171
4	58706	205278

	Precision	Recall
0	0.78	0.77
4	0.77	0.78

Feature Analysis. Log ratios were calculated for the features to determine what the model considered to be the most positive and most negative features. Additionally, the features which ended up in the middle were the most 'neutral' features. When calculated most of the top features made sense, e.g. the most negative features included words like "sad" or "miss" and the most positive features included "lol" or "thanks". Tokens identified as very low in either positive or negative sentiment mostly included names and other proper nouns. The table below shows the calculated log ratios and top features of the model.

Most Negative Features		Most Positive Features	
-5.47	sad	-5.58	time
-5.42	like	-5.51	going
-5.42	want	-5.49	like
-5.38	today	-5.48	lol
-5.35	day	-5.20	day
-5.29	miss	-5.13	thanks
-5.16	don't	-5.11	love
-5.10	just	-5.09	just
-5.02	work	-4.97	im
-4.64	im	-4.85	good

Real Data Performance. Following the model creation and evaluation it was necessary to see how it actually performed using the financial tweets I gathered.

I chose two files at random from the sample of 30 days and analyzed them for anomalies and incorrect predictions. Both sheets scored about 75% correct and ~79% correct for classifying sentiment however it was during this hand analysis in which two issues were uncovered.

1. Tweets were getting through to the dataset which had little to do with the company (e.g. Tesla had tweets about the inventor rather than the car company)
2. Certain tweets which were undoubtedly positive or negative (e.g mentioning Nazism or saying “I love x”) were being marked incorrectly.

I attribute these errors to poor quality training data. Because of the way the data was annotated it is possible that people were tweeting things most people would consider negative with positive emojis and sentiment. To fix this sentiment tweets for training should be curated from people who do not possess a dubious moral background. Additionally, the tweets should be annotated by hand as I believe this would increase the ‘common sense’ of the model.

Final Analysis

With a working model and the necessary data collected and transformed, the final process for testing my hypothesis was to choose some companies to analyze. I chose three companies: Tesla, Microsoft, and Barrick Gold. First, I used a python program to acquire and feed the model the 100-tweet dataset for each day in a 30-day window of the stock period. Because the model could only predict positive or negative the mean was calculated for each day leaving me with a table containing the date and

the mean sentiment score (between 0 and 4) for that date (based on the 100 tweets of the day). While mean was chosen to represent the days sentiment further analysis concluded that analyzing the ratio of positive to negative tweets could also show the overall sentiment of the company or product. Finally, this data was merged with the financial data based on the dates of both files, and regression was conducted to test for a relationship between either closing price and sentiment or daily return and sentiment. Microsoft is presented below as it proved to show the strongest correlation and the tweets requested were more or less all about Microsoft.

Microsoft Evaluation

Microsoft is one of the largest software and information technology developers in the world. With a strong brand worldwide, there were no shortage of people expressing their sentiment about it over social media, especially twitter. I conducted linear regression on both of the financial indicators of the stock and the results are shown in the two clusters below. Additionally, below is a timeseries showing Microsoft's mean sentiment across the month of November. Interestingly the uptick at the end of the month is associated with the Microsoft and US military HoloLens announcement. Overall no R^2 was too high so we can't make too many inferences from the data.

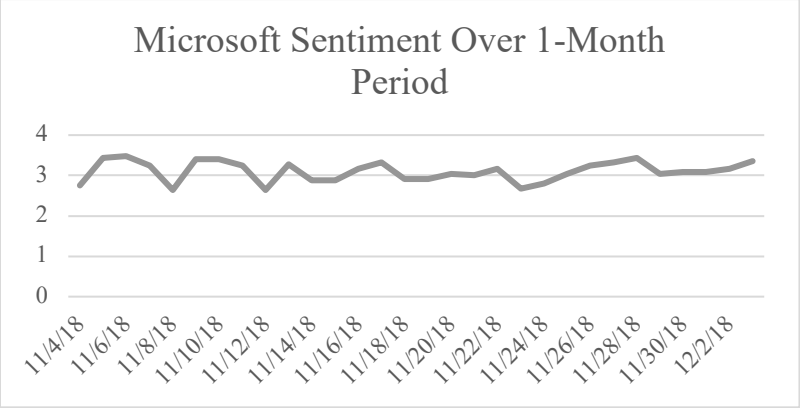

Figure 1- Microsoft Monthly Sentiment

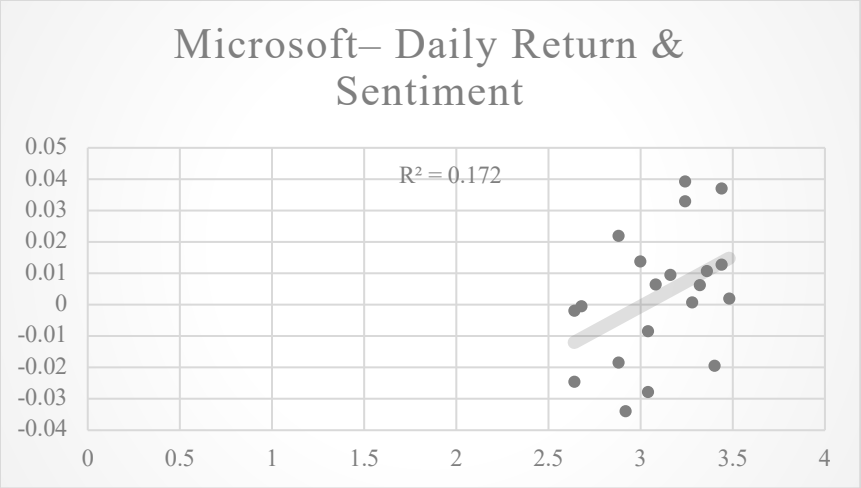

Figure 2 – Microsoft Daily Return & Sentiment

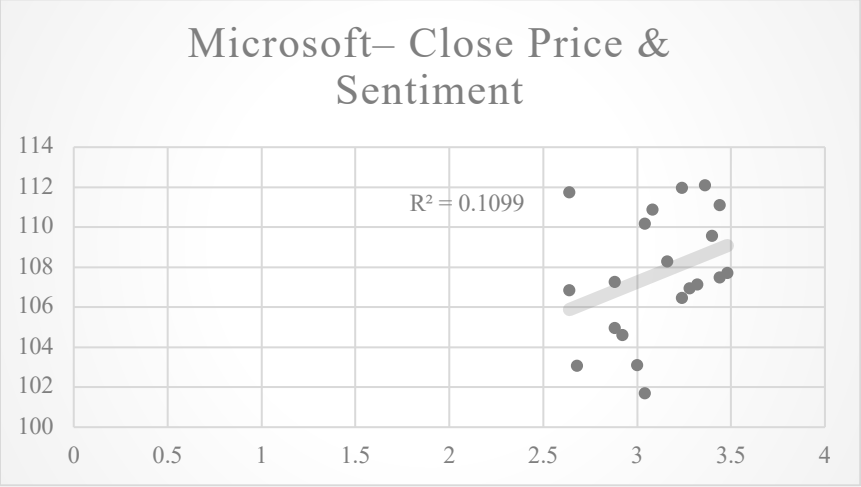

Figure 3 - Microsoft Close Price & Sentiment

Conclusions

No strong trend was readily apparent however I believe that further analysis and modification of the methods used in this project could yield interesting results. Perhaps an improved approach to this task would be to restrict tweets to influential people and organizations. If a list of approved twitters could be compiled, then only tweets from those sources would be collected and analyzed for their sentiment about a particular stock or industry. This could improve data quality drastically perhaps increasing the

clarity with which we can see the trend between social media and market performance. additionally, new methods for calculating daily sentiment should be experimented with.

References

[1]https://money.cnn.com/2018/02/22/technology/snapchat-update-kylie-jenner/index.html

[2]http://help.sentiment140.com/for-students

[3]https://github.com/scikitlearn/scikitlearn/blob/master/sklearn/feature_extraction/stop_words.py