Predicting Quarterly Earnings using Sentiment Analysis and Regression Models on Reddit Comment Data

Max Liu | Spencer Ho | Jonathan Li
Spring 2017

Abstract:
In 2011 analysts noticed an interesting trend: Berkshire Hathaway stock seemed to rise when actor Anne Hathaway had increased mentions in the news. Although there may be no fundamental reason why this happens, there did seem to be a correlation. Our goal was to test a similar concept using Reddit comments. We were interested in seeing if sentiment towards words that were closely tied to a specific company would be able to predict how well that company would do. Using the immense amounts of data in the Reddit comments dataset, we aimed to see if we could spot similar correlations through word mentions and sentiment analysis. The first step of our project involved finding commonly mentioned companies and products in Reddit comments as well as associated words that occur frequently in comments containing a given company name. The second step was conducting sentiment analysis on comments containing these keywords and testing correlation between sentiment in Reddit comments and stock performance. We ran our analysis on two sets of companies: the first being the top 100 most mentioned companies in the comments from an initial list of the top 1000 largest companies on the NYSE, and the second being an index of 50 household name companies whose names are synonymous with their products. The analysis on the the first set aimed to test the Berkshire Hathaway/Anne Hathaway associative effect while the second set aimed to test the correlation of the actual company sentiment without the associative effect.

Data:
We chose to use the archived Reddit comments dataset for our project. This data set contains all Reddit comments from 2006 to 2017, but we chose to focus on a subset of the 2016 data due to size constraints. Our initial dataset was over 300 gigabytes compressed and over 1 terabyte uncompressed; however, we ended up conducting our analysis by sampling a subset that was greater than 100 GB. The raw data set was a text file containing lines of json strings. Due to the broad nature of our data, our aim was to find insight in areas that our analysis could benefit from the massive and varied data of Reddit comments. We decided to focus on company performance as large companies are likely to be mentioned across the many verticals of Reddit and seemed to be a promising topic for sentiment analysis.

We also used financial data from Thomson One for a list of the top hundred most commonly mentioned company names from an initial list of the top thousand largest companies on NASDAQ by market cap. We also used the same data on a list of "household name" companies that are mentioned frequently and naturally in everyday speech, in the hope that our results for these companies would be less variable. We ran multiple regression models to compare how differences in actual and expected earnings per share correlate with our sentiment analysis for the companies.

Methods and Assumptions:

1. We began by keeping track of an initial 1000 companies on NASDAQ as well as a "household name" company list* and searching for the most mentioned companies on Reddit and creating a bucket of the most commonly associated words for those companies.**
2. Next, using those buckets of words, we performed sentiment analysis for each word and tracked weekly and monthly changes in sentiment and mentions toward each individual word.
3. Finally, by using the bucket of words for each company we are able to create unique sentiment and mentions data points that we then analyzed. Our analysis involved multiple regression methods to find the best predictor for earnings per share.

*The company names in the list were data cleaned with regular expression to be both more usable and likely in comments (i.e. Honda Motors Corporation cleaned to Honda) while still taking into consideration and making adjustments for certain companies having higher appearances due to the nature of their names (i.e. Apple or Coach). This data cleaning portion proved to be far less problematic for our "household name" company list.

**We use this method because this will take into account industry performance (using Apple as an example again,

if technology has a negative sentiment, this will drag their score down but their score can still be positive if their other words have a positive sentiment).

Hypothesis:

Our initial hypothesis was that positive sentiment towards words in a company's bucket will most likely correlate with better earnings performance (positive sentiment towards iPod and Mac may result in better earnings for Apple).

We tested this hypothesis on two lists of companies: the top hundred mentioned companies in the NASDAQ largest thousand companies by market cap and a "household names" companies we compiled. Our compiled list is comprised of mostly large cap stocks from a few industries. We expected our hypothesis to be more accurate for our analysis with the "household name" companies as opposed to the other list where the companies are more likely to be spread very broadly across a variety of market caps and industries.

Algorithms:

The primary algorithms for our project were related to sentiment analysis on the comments dataset as well as multilinear regression to observe correlation between sentiment and earnings.

For our project's sentiment analysis, we used a combination of NLTK (natural language toolkit) and TextBlob. Our analysis took advantage of NLTK tokenize to divide the reddit comments into usable pieces, NLTK stopwords to parse out language that did not add to the sentiment, and NLTK part of speech tagging to focus on only object words when considering associated products for the companies in our analysis. Once the comments were formatted with NLTK, we used TextBlob to conduct sentiment analysis on sentences within comments that contained the keywords of our interest. We found that TextBlob's polarity sentiment analysis gave us the most consistent and robust results compared to NLTK's Vader sentiment analysis or any other models we attempted to train using online training data. Furthermore, due to the broad and varied nature of our data, we found it best to use a similarly broader sentiment analysis tool that was less specialized in analyzing specific formats of language, such as tweets or movie reviews.

The second portion of our analysis used Pandas multilinear regression tools to find the best independent variables to fit the actual to expected earnings ratio. Examples of independent variable we put up for testing include total company mentions in a period, sentiment of company name for a period, and sentiment of company associated products for a period. We used the output of our multilinear regressions to test our above hypothesis on whether or not the sentiment of Reddit comments could have any predicting power on stock performance. We will elaborate on this later.

Big Data Approaches:

1. **MRJob**
   We used MRJob to do most of our data analysis over the raw data set. This involved first keeping track of word mentions and building out a list of each company's list of most mentioned words. We wrote this mapreduce function in 2 steps. The first step kept track of the most mentioned words and the second step reduced the list to only the top 100 mentioned companies. A list of the current top 100 companies was kept through each loop to prevent storing all of the data in memory.
   Our second mapreduce script took in the bucketed words associated with each company and performed sentiment analysis for each comment if any of those words were present in the comment. This required keeping a list of words that we were interested in analyzing and checking each comment for all of those words.
2. **Dataproc**
   In order to use mapreduce to its full extent, the mapreduce scripts were deployed through Google Dataproc. This involved setting up multiple nodes and allowing every node to work on the job in parallel. This step proved to be a huge challenge which will be elaborated on later.
3. **Random Sampling**
   Because our data set is so large we were unable to run all of the tests that we wanted to. In order to increase the breadth of data that we used we randomly sample parts of different months to test.

Run Times and Optimization:

The primary problem we encountered while trying to increase efficiency in run times was decreasing the amount of loops needed in order to accomplish our task:

1. **Keeping track of companies**
   In our first mapreduce we had to constantly search to see if any of the company names appeared in the comment. This required 1000 loops for every comment. Instead of taking this approach we decided to check if any of the words in the comment were in our company list. In order to decrease this loop further, we looped over only the noun words. This resulted in a speed increase but was not the fastest as we had to use NLTK to pre tag the comments which took more time. The fastest way we found was to only remove the stopwords, where we actually had a longer list than just nou words, which was very quick and loop over the rest.
   Looping over just nouns took about 0.6x as long as looping over everything, on the other hand, looping over everything but the stop words took about 0.4x as long as looping over everything.
2. **Searching faster**
   Soon after we realized that looping over a list was not the best way to search for words. We converted all of our lists to sets and saw a significant speed improvement. The resulting loops ran about 0.3x the original speed which was a significant speed improvement.

Data Analysis Tools:

1. **Multi-Linear regression**
   We used multi-linear regression as our main source of data analysis. Multi-linear regression allows us to see if multiple explanatory variables can predict an observatory better than a single explanatory variable. Our observatory variable is the percentage difference between predicted EPS (earnings per share) and actual EPS for a given company. Our explanatory variables were average sentiment and total count for weeks 1-9 and months 1-3. Using 50 companies, we randomly selected 40 to build the training model and used the other 10 as tests to see if our model could predict that a company would beat its expected earnings. We would run this test (randomly selecting testing companies and training companies and making multi-linear regression models), 10 times and took the average R^2 and success score to get more accurate results. Our initial data only looked at week 9, so we only had two explanatory variables: w9cnt and w9avg. We then used a "tailored" data set, giving us information for weeks 1, 5, 9 or the first week of January, February, and March 2016. Our results drastically improved.

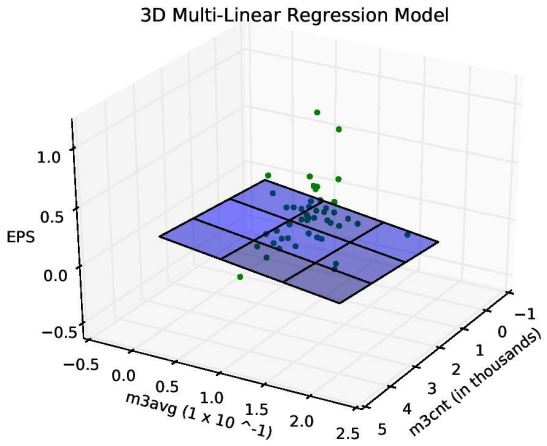
Challenges:

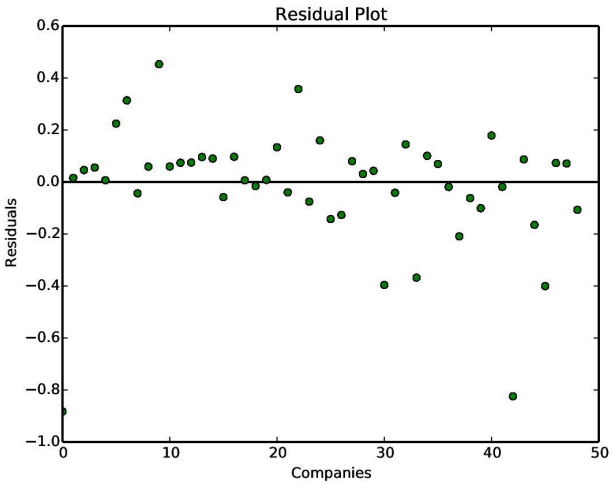
1. **Dataproc**
   Perhaps our biggest challenge was running our mapreduce scripts over dataproc successfully. The main problem we faced with dataproc was installing textblob successfully on each node. In order to install textblob, textblob needed to also download external dependencies that did not come with the initial package. We found that the external files were downloaded into an unknown directory that textblob could not find. Ultimately the following command was able to install the packages and move them into the correct directory:
   ```
   --bootstrap='sudo pip3 install textblob; python3 -m textblob.download_corpora; mv /root/nltk_data /usr/local/lib/; chmod -R a+rX /usr/local/lib/nltk_data'
   ```
2. **Filtering common words**
   In order for our models to be more accurate, it would be better if each company's bucket is different from each other but if we have too many common words that do not mean anything this will hurt our results.
3. **Data Cleaning Company Names**
   As our analysis depended on picking up company mentions, cleaning the names of large companies into more usable formats had a great effect on our results. Simple data cleaning methods included dropping suffixes such as "corporation" or "company" however this still remained imperfect for certain niche cases causing company names to either remain too specific or too broad. This difficulty also motivated running our analysis on a household name set as referential ambiguity in comments was much easier to avoid with companies such as Microsoft or Disney.

Results:

Using our first dataset (just week 9), our R^2 results were less than 1% and were predicting successful eps around 50% of the time, meaning our model could not predict the EPS of companies. Using the tailored dataset, the R^2 for all months averaged between 15-20% and success scores around 70%. We found the best predictors were

March's counts and sentiment analysis average having a combined R^2 score between 5-10% and having a success rate similar to the success rate using all months. February was the second best predictor and January was the worst predictor. Given that most quarterly earnings reports come out in March, the data suggests that comments made closer to the earnings are better predictors of whether a company beats earnings. Taking the best two predictors, we created a 3D linear regression model with its respective residual graph. These graphs are pictured below:





Conclusion and Future Analysis:

Ultimately, we were able to get some meaningful results out of our analysis despite being forced to work with only subsets of our data.

For future analysis, it would be interesting to use all the data for all 3 months. It would be interesting to see if our R^2 and success predictor averages increase and if March comments remain the best predictors for the model. Eventually, we would like to test this on the whole year to see if our hypothesis that the month of earnings is the best predictor for a company beating earnings.