香　港　大　學

THE UNIVERSITY OF HONG KONG

FINA 4350: Text Analytics and Natural Language Processing in Finance and Fintech

December 2019

Professor Matthias Buehlmaier

Written by: **Fintech Gang**

Khoo Pei Jing 3035493050

Cho Ming Hang 3035377701

Dhruv Batra 3035248118

Tang Yuehua 3035534153

The Project

Our project was focused towards looking at a very common topic within NLP for finance – Earnings Releases. While the common analysis in this area is to analyze speeches and statements of senior executives of a firm to predict earnings, we chose to leave this part to individual investors and instead, analyze their opinions on certain social media platforms. The underlying idea is that markets are mostly efficient in the manner that publicly available information would be consumed by individual investors and the aggregate idea would be reflected in their posts on Twitter, StockTwits and Reddit.

The implication of such a study is that it potentially removes the need for an investor to collect a large number of sources that may provide an insight about the earnings releases of a firm. Instead, it restricts information sources to just a few and in a manner crowd-sources information.

The focus of the project is on predicting earnings surprises – the over or under performance as compared to the market expectation, as provided by Yahoo! Finance (we will take the market expectation as-is) – using textual data from Twitter, StockTwits and Reddit.

What we did

The Data Sources

We started with an Earnings Release Calendar from Yahoo! Finance which was the basis for further data collection. This dataset has a total of 1484 observations that span over 2018-2019. Each observation corresponds to one company-earnings release pair (there will be multiple observations for the same company and multiple observations for the same date). Each observation has an expected EPS along with the actual EPS that was released. Using these two data points, the earnings surprise is calculated and is used for the rest of the analysis as the dependent variable. We generated an "Index" for these rows by combining company tickers and the month and year of earnings release so that each row would have a unique index.

Next, we went through the three social media platforms to collect textual data that was posted within a few days before the earnings releases –

Twitter remains to be one of the most popular forms of social media in today's day and is a source of a large amount of user opinions and views. Consequently, using twitter is a natural choice for this project as it would be able to provide a big data set that could be relevant for the study.

StockTwits is similar to Twitter in some manners but is focused on posts about the stock markets. This makes it's textual data very relevant for conducting financial analysis. However with a such a niche in its userbase, it is not as popular as Twitter and cannot match it in terms of the quantity of data.

Reddit is another popular social media platform that has discussion forums – subreddits – about almost everything. Further, unlike many other platforms, Reddit allows the downvoting of posts and comments. This means top content on Reddit has been through a reviewal process that is essentially crowd-sourced in a democratic manner. This could create a lot of value given the content available on the platform as the accuracy of posts (in terms of majority's approval) is already weighted into posts.

Data Collection and Processing

Twitter – Although python has the Tweepy package, it only allows the user to retrieve data within the recent 7 days. Therefore, we are using the request package to scrape data using a GitHub open-source code. First, an object of the class TwitterCriteria is created which contains the stock ticker and the relevant dates. Next, a Scraper object is created using the Twitter Criteria object that can scrape through Twitter with the relevant search query. The Scraper returns textual data along with other attributes of twitter data such as the username, number of re-tweets and the number of likes. This process is run in a loop to collect data for all earnings releases. Within the loop, the data is pushed through the text-processing function (see below) and generates a dataframe that has the mean sentiment score for each ticker-date pair.

StockTwits – As Stocktwits doesn't allow scraping, we used its API to request our desired data. Given there is a rate limit for Stocktwits API, we referenced a code on Github that creates a proxy to get over the rate limit. For each given ticker, the code will get all historical twits related to that stock in a reverse chronological order indefinitely, starting from today. The data we really want for each stock is the twits that were posted in a specific interval before the company earning release dates. Consequently, the process becomes very time consuming as the filtering of twits happens only after all historical twits are collected.

Because Stocktwits will throttle the download speed and we added an extra filtering system, the code runs unexpectedly slow. Also, after running for a long time (i.e. 10 hours), the code started to write some empty files. It seems that the throttle renders the code ineffective. For this project, we did not manage to come up with a solution. We got approximately 70% of the data that we expected at the end.

The code outputs one file for each stock and within that one file is data for all four periods. We need to organize them into something that can be put into our processing function. Here we mainly use Pandas to manipulate and restructure our data. In terms of processing, one thing worth noticing is that for StockTwits data, if we put the raw textual data into our text processing function first, and then apply the "textbolb" function, all our sentiment scores will be 0. If we apply the "textbolb" function directly on the raw text, we will get mostly non-zero score. With limited coding knowledge, we cannot provide a thorough explanation of this. We believe that a lot of functions in "textblob" overlap with our text processing function, which renders such a problem.

Reddit – Reddit provides an easy to use API called PRAW that allows unlimited access to data from the platform. We created a custom class of objects that would be able to hold a reddit objects (a class defined by the PRAW API that contains several attributes of each reddit submission) and the corresponding company-earnings release index. Given the API's functionality, we were able to filter out submissions that were made in the desired time period at the time of requesting data. Therefore, the process was fairly quick and efficient. There are two rounds of search queries that are run – one using the ticker of the company and one using the company short name. While this covers most scenarios, it is not entirely perfect as certain tickers such as INFO (HIS Markit Ltd.) would return results that would be referring to entirely different topics.

Next, a dataframe with Index, textual data from the post and all top-level comments and certain reddit attributes was created. Our text processing function was run on each row's textual data and

consequently, a sentiment score (polarity) was generated using “textblob”. The dataframe was then grouped by Index and the mean sentiment score was used for further analysis.

Text Processing – A common set of text-processing functions were written that consisted of removing punctuations, links, stopwords, alpha-numeric words, special characters and tickers. Next, we changed all text to lower case and tokenized it. Finally, the “textblob” function was used for data of all three sources to produce sentiment scores.

What we found

1. Data preparation
    a. Dealing with the empty data
    When managing the data in excel files, we notice that there are much missing data for both sentiment scores and earnings release.
    For earnings release, it’s not possible to simply consider the missing data as any default value. Also, considering the missing scale as 22/1486 = 0.0148, which is regarded as not significant in statistics. Therefore, we will delete the empty earnings release data points.
    For sentiments data, as Reddit and Stocktwits are not widely used social media, there is a massive missing data, which implies no discussion on that particular stock in our stated time period. However, in this case, it is rational to simply consider those missing scores as default 0, which implies no or neutral effect on investor sentiments.
    b. Managing the input data
    The Textblob function outputs a sentiment score between -1 and 1, so the average of text sentiment should also score between -1 and 1. Since Naive Bayes Classifier requires non-negative data input, so we add 1 to all the sentiment scores, which should not affect the analysis of our data as it only changes the range of sentiment scores. Then, based on two average scores, we can denote the sentiment change by the percentage change between the two standardized scores. The final input data are the average sentiment scores for the 5 days before the earnings release and the percentage change in the sentiment when it’s closer to release.
    c. Labeling the surprise
    For model simplicity, we labeled the positive surprise as 1 and non-positive surprise as 0.
2. Model training and evaluation
    We train our model with a test size = 0.33 and two-dimension input data.
    To evaluate the model, we will use the accuracy score and classification report to measure model performances.
    a. Naive Bayes Classifier:
    Accuracy score: 0.7785

	precision	recall	f1-score	Support
negative	0.00	0.00	0.00	0.00
positive	0.78	0.78	0.78	483

Micro average	0.78	0.78	0.78	483
Macro average	0.50	0.39	0.44	483
Weighted average	1.00	0.78	0.88	483

b. Random Forest Regressor:
Accuracy Score: -0.2721

	precision	recall	f1-score	Support
negative	0.20	0.23	0.21	91
positive	0.81	0.78	0.80	392
Micro average	0.68	0.68	0.68	483
Macro average	0.51	0.51	0.50	483
Weighted average	0.70	0.68	0.69	483

c. Decision Tree Classifier:
Accuracy score: 0.6667

	precision	recall	f1-score	Support
negative	0.23	0.24	0.24	104
positive	0.79	0.78	0.79	379
Micro average	0.67	0.67	0.67	483
Macro average	0.51	0.51	0.51	483
Weighted average	0.67	0.67	0.67	483

d. Gradient Boosting Classifier:
Accuracy score: 0.7660

	precision	recall	f1-score	Support
negative	0.02	0.20	0.03	10
positive	0.98	0.78	0.87	473
Micro average	0.77	0.77	0.77	483
Macro average	0.50	0.49	0.45	483

Weighted average	0.96	0.77	0.85	483

Although this classifier provides the highest accuracy, the final data shows that 77.83% of the earnings surprise is positive, which is very close to our accuracy score. This closeness and the matrix result implies that the Naive Bayes Classifier predicts any input sentiment as 'positive'. Therefore, we can conclude that the Naive Bayes Classifier has no predictive power.

Also, the accuracy score for Random Forest Regressor is negative, which means extremely low accuracy. Compared to other classifiers, this may not be a good choice to predict the earnings surprise.

The Decision Tree Classifier and Gradient Boosting Classifier both provide acceptable accuracy scores. However, Gradient Boosting Classifier only predicts 2% of the samples to be negative, which is significantly lower than the expected rate 22.17%. For the Decision Tree Classifier, it predicts 21.53% of the samples to be negative, which is much closer to reality. Based on these analyses, we will choose the Decision Tree Classifier to be our predicting model.

What could be done better

Firstly, there are some shortcomings of our data that affect our analysis – we used data for a large number of companies (all S&P500 companies) which creates a problem as not all companies are popular enough to have people talking about these companies on social media. A potentially better analysis could be done with some of the more popular companies' data for a longer time duration.

Further, our textual data has some issues as it is difficult to effectively filter out irrelevant posts/submissions from social media. This is because social media posts about stocks do not always have a common identifier such as a ticker – a user might refer to Apple as Apple instead of AAPL or sometimes even with an apple emoji. Additionally, a search for the word Apple has a high probability of returning posts that are irrelevant. This problem exists for all data sources in different manners. Possibly, we could identify specific frequently used names of companies and use them to search for user posts.

Secondly, there is a lot of information that could be gained by making use of the number of likes, retweets, upvotes, comments etc. which would require a different set of analysis to be able to assign weights to posts and generate a synthetic sentiment score.

Conclusion

The project explored the relation between investor sentiment a few days before earnings releases and the consequent earnings surprises with the hypothesis that the aggregate investor sentiment would contain predictive power for earnings surprises. The Naïve Bayes Classifier, Random Forest Regressor, Decision Tree Classifier and Gradient Boosting Classifier were run on the data and only the Decision Tree Classifier returned results that agree with the hypothesis with an accuracy of 66.7%. This result can possibly allow us to predict earnings surprises by monitoring social media data for the days before the earnings release and hence, provide us with a profit-making strategy in the stock market.