Predicting Stock Market Overreactions Using NLP

Neil Shweky NSHWEKY@SEAS.UPENN.EDU
Hersh Solanki HSOLANKI@WHARTON.UPENN.EDU
Kamal El Bayrouti EMOH@WHARTON.UPENN.EDU

Abstract

Our project aims to create a investment strategy that focuses on the relationship between the linguistic properties of companies' earning calls and consequent stock market overreactions. Using natural language processing (NLP) techniques, we trained several learners to read earnings call transcripts and predict whether or not the market was going to overreact. Our highest performing learner was a Multinomial Naive Bayes model. We believe that the predictions in our model can serve as a feature in a multi-factor investing strategy or as a stand-alone investing strategy.

1. Introduction

At the first market open after a company's quarterly earnings call, it is common to see a spike in the stock price followed by a correction a few hours or days later that reflects an initial overreaction to the earnings call. We had the intuition that the language company executives use in the earnings calls has a significant impact on how the market perceives the company's health. This intuition is nothing new. The novelty in our project lies in the use of NLP on earnings call transcripts to tease out undetected linguistic features that correlate with market overreaction. We reasoned that market overreactions are largely driven by market sentiment which in turn is largely driven by what the market hears and reads. Stock price movements are extremely difficult to predict; however, we believe that focusing on one-time events such as earnings calls and having a narrow prediction time horizon (2-3 days) gave us a good chance at meaningful predictions.

The current industry standard, for even the best long/short equity investment firms, is a correct prediction rate of 53-57%. Profits are generated despite the relatively low accuracy via the use of leverage. For our projected, this meant that a slight improvement beyond random choice (50% accuracy) would result in a meaningful investment strategy.

2. Data Collection and Processing

1. The first step was to amass a collection of earnings call transcripts. We did not have access to institutional-grade software that allowed us mass download transcripts; instead, we manually collected 378 transcripts across 11 companies dating back to 2009 from FactSet.

2. The transcripts were in a PDF format. We used the pdfminer Python library to parse through the transcripts and extract the text data.

3. Each transcript had to be labeled as having cause an over reaction or not. To do this, we pulled the relevant company's stock price at the last market close before the earnings call, the first market open following the earnings call (usually the next morning), the end of day (EOD) price following the earnings call,

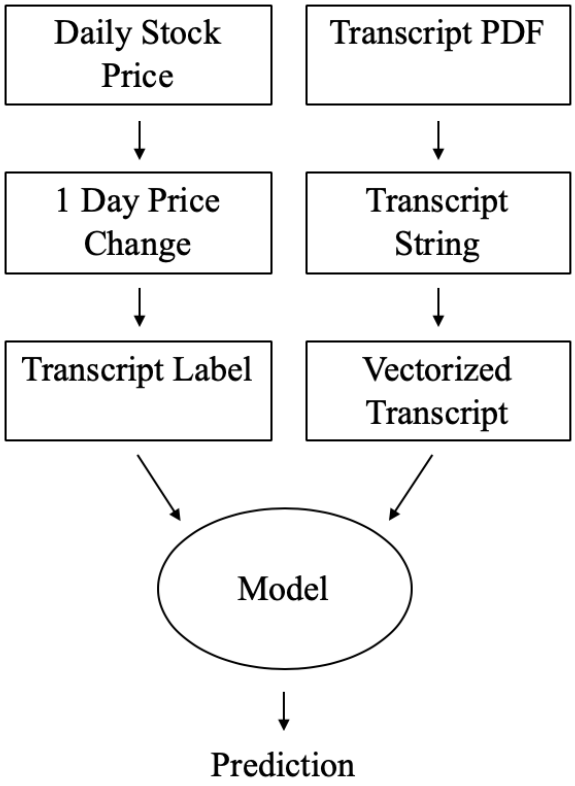
University of Pennsylvania, CIS 419/519 Course Project.

and the EOD price 3 days later. If the stock price reversed direction by more than 50% of the initial price change, we labeled the transcript as leading to a market overreaction (1). Otherwise, we labeled the transcript as neutral (0).

4. We used a bag of words model to represent the transcripts. We experimented with both term frequency and TF-IDF to weight the significance of the terms.

5. After running a few test trials, we found that comparing the EOD price 1 day later to the initial price yielded the most meaningful results. As such, we narrowed our labeling scope to only look at the change after 3 days.

Figure 1. Model Process



Model Selection

Our primary evaluation metric was prediction accuracy over 5-fold validation. Our first set of models are all variations of Naive Bayes. NB was the most logical first option given its ubiquity and success in text-classification. We then used an SVM given its ability to separate non-linearly-separable data. Finally, we used a logistic regression model as additional competition to the Naive Bayes models.

1. We decided to use Multinomial Naive Bayes with a Term Frequency Vectorizer as our first training model. These are the parameters (and the ranges) we used:

   - ngram=(1, 1-5)
   - alpha=.01-.11 in .01 increments
   - stop_words='english' and None
   - max_features=2000-20000 in 500 increments

   After training the model, out best parameters were: 25,000 max_features, stop_words=english, alpha=0.03, ngram=(1,2). Our best evaluation metrics where: accuracy = 58.88%, recall = 58.88%, precision=60.78%

2. Next, we used Multinomial Naive Bayes with TF-IDF features to train our model. These are the parameters (and the ranges) we used:

   - ngram=(1, 1-5)
   - alpha=.01-.11 in .01 increments
   - stop_words='english' and None
   - max_features=2000-20000 in 500 increments

   After training the model, out best parameters were the same as for term frequency: 25,000 max_features, stop_words=english, alpha=0.03, ngram=(1,2). Our best evaluation metrics, however, were: accuracy = 58.85%, recall = 58.85%, precision=59.83%.

3. We also used Gaussian Naive Bayes with both Term Frequency and TF-IDF features. These are the parameters (and the ranges) we used:

   - ngram=(1, 1-5)
   - alpha=.1-1.1 in .1 increments

- stop_words='english' and None
- max_features=2000-20000 in 500 increments

After training the model, the best parameters did not have a higher frequency than that of Multinomial Naive Bayes, so we decided not to use this in our model.

4. Next, we decided to use Support Vector Machines with both Term Frequency vectors and TF-IDF vecors. These are the parameters (and the ranges) we used:

   - ngram=(1, 1-5)
   - C=1-101 in 10 increments
   - stop_words='english' and None
   - max_features=2000-20000 in 500 increments

   After training the model, the accuracies we got ranged from 49% to 52%. This is an absolutely terrible model as it is about as good as guessing randomly. We also used different kernels for SVMs, including cosine-similarity, linear, polynomial, rbf, and sigmoid. We did not use SVMs for our project.

5. Next, we decided to use Logistic Regression with both Term Frequency vectors and TF-IDF vecors. These are the parameters (and the ranges) we used:

   - ngram=(1, 1-5)
   - stop_words='english' and None
   - max_features=2000-20000 in 500 increments

   After training the model, the best parameters did not have a higher frequency than that of Multinomial Naive Bayes, so we decided not to use this in our model.

3. Results

Below are the summarized results for our tested models. Our best performing learner was the Multinomial NB model with TF weighting.

Table 1. Classification Accuracy for Each Model

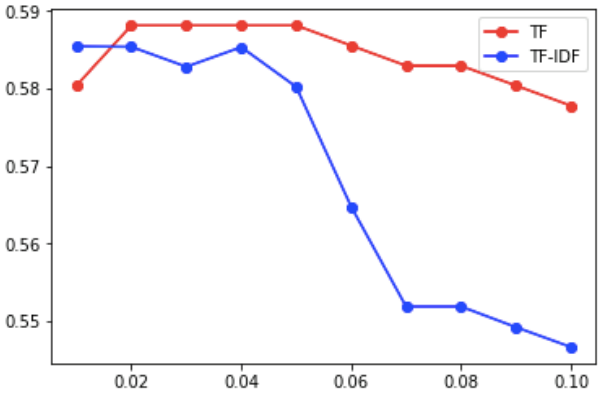
MODEL	ACCURACY
MULTINOMIAL NB TF	0.59
MULTINOMIAL NB TF-IDF	0.58
GAUSSIAN NB TF	0.54
GAUSSIAN NB TF-IDF	0.53
SVM TF	0.49
SVM TF-IDF	0.52
LOG. REGRESSION TF	0.51
LOG. REGRESSION TF-IDF	0.52

First, we will discuss our poorly performing models. It can be clearly seen in the table above that SVM and Logistic Regression models performed almost no better than random chance, regardless of the term weighting used. This performance was expected given that these models are unsuitable to the bag of words feature inputs they received. In our estimation, these models can only viably become useful if different features were extracted and fed into them.

The Naive Bayes models performed surprisingly well, at best yielding a 9% improvement over random choice. Multinomial NB performed better than Gaussian NB under both term weightings. This makes sense given that terms in earnings calls are unlikley to be normally distributed.

An important insight in the data is that TF weighting performed better than TF-IDF weighting under all models. Furthermore, when focusing on the Multinomial Bayes model, we noticed that TF responded better to changes in the alpha parameter (Laplace smoothing parameter). As show in the graph below, TF-IDF performed better with alpha=0; however, TF performed reached peak performance when alpha was between 0.02 and 0.05, beating the peak performance of TF-IDF. Furthermore, we noticed a drastic increase in performance as alpha was increased.

Figure 2. TF vs TF-IDF Accuracy versus Alpha



Although it is not entirely clear why TF would perform significantly better than TF-IDF in all scenarios. Our main hypothesis is that TF-IDF might weight words that are idiosyncratic to a specific earnings transcript (economic conditions at the time of earnings call etc...) which are not important for generalizing a rule as to what qualifies as an over reaction or not.

4. Limitations and Future Experiments

1. Data collection: an obvious constraint of our project was the tedious nature of collecting earnings call transcripts. There are paid software packages that allow us to download transcripts in bulk and greatly reduce data-collection time. This would allow us to increase our data set by orders of magnitude.

2. Single company training: all models were trained on the entirety of data-set. Due to a lack of available transcripts, we were unable to create a learner for each individual company without risking over-fitting. Predictions on a company by company basis may yield better results.

3. Criteria for over-correction: our current criteria for an over-reaction is if the stock price moved by a certain amount in the opposite direction 3 days after the initial spike. We can make the criteria more nuanced by looking at multiple time-frames, and also looking at non-price factors such as trading volumes.

4. Correcting for market movements: the current stock price changes are not normalized for changes in the overall market. As such, the models may be accounting for price variation that arise independently of the earnings calls.

5. Market consensus as a feature: there are several indicators that reflect the market's sentiment on a given stock. These can be incorporated as a feature in any of the models to improve prediction performance.

6. Neural Network: a neural network with an embedding layer is a viable model type that we did not implement. This type of model has proven succesful with text classification and may yield improved performance.

7. Back testing: we can analyze the profitability of a strategy entirely based on the model by looking at the profits that we would have made had we followed the model's predictions in the past, while accounting for trading and leverage costs.

5. Conclusion

Our project forms a solid foundation for future development. The current results are promising, and indicate that there may be a chance to generate a profitable investment strategy using our model. With the implementation of the ideas in the section above, we believe that we can create a robust predictive model. At minimum, the model could serve as an indicator to help guide an investor in his investments. In the best case, the model could serve as a stand-alone algorithmic investor that can generate profits.

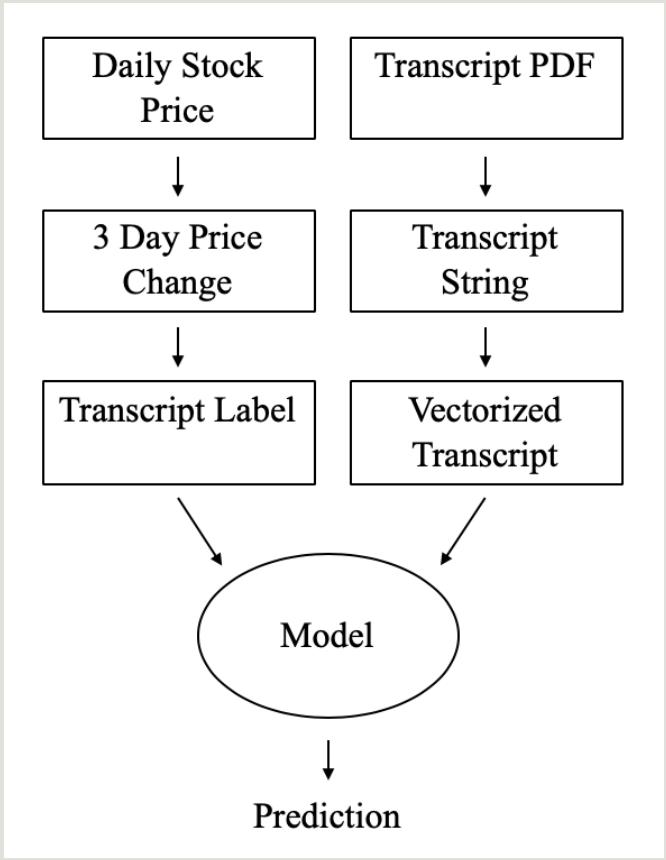
Understanding Market Overreactions using NLP

Overview: We wanted to see how text analysis could help predict overreactions/underreactions of stock prices based on corresponding Earnings Calls

Data: 378 Transcripts off Factset for 11 companies + corresponding share prices for 10 years. Utilized 5 fold cross validation to increase training size.

Final goal: Develop the basis of a trading strategy that may be rule based or discretionary

Process



Classification timeframe: Delta of price at market open and price at market close

- 1 if positive, 0 if negative

Classifiers Used:

- SVM with multiple kernels
- Logistic Regression
- Gaussian Naive Bayes
- **Multinomial Naive Bayes* (Best)**
- Adagrad
- Adaboost
- Random Forest

MNB Parameters

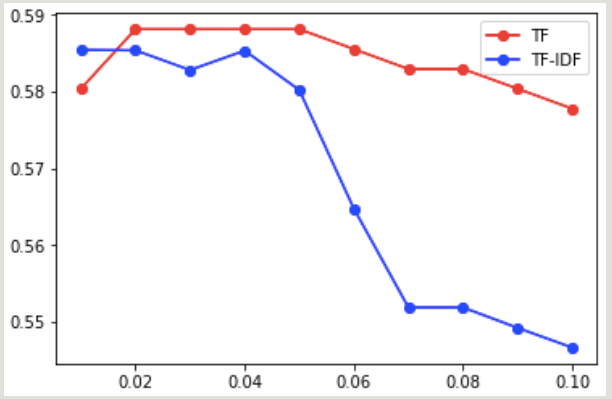
- Ngram range (1-2)
- Max features - 25,000
- Stop words - English
- Alpha – 0.03

Results

Count Vectorizer

Classifier	Accuracy %
MNB	58.81%
SVM	49.50%
LOG REG	51.04%
GNB	53.64%

Count Vectorizer outperforms TF IDF on nearly all alpha levels



TF IDF Vectorizer

Classifier	Accuracy %
MNB	58.28%
SVM	52.34%
LOG REG	51.06%
GNB	52.61%

Analysis of Results

- At first, anything below 60% accuracy may not seem fascinating. However, in the financial markets, every basis point (0.01%) can mean millions.
- A 58% accuracy is game changing. A simple systematic trading bot that uses leverage can consistently beat benchmarks.
- A trading strategy that plays off these results could be implemented via Quantopain.

Limitations and Future Research

- Understand the impact of trading costs on this trading strategy to make it realistic.
- Use only a singular companies to predict that companies reactions. This allows for standardization of speech.
- Train on a greater set of transcripts for a better accuracy number
- Test out different types of time periods, rather than the strict 1 day, 3 day, etc.
- Understand the effect of leverage on returns
- Sentiment as a feature
- Neural nets, especially converting words to vectors