Predicting Stock Prices through Textual Analysis of Web News

Daniel Gallegos
dgalort@stanford.edu

Alice Hau
alicehau@stanford.edu

Problem Statement

Investors have access to a wealth of information through a variety of news channels to inform them on their decisions to buy and sell stocks when managing their portfolios. Our project's goal was to simulate and improve on this process and predict stock price fluctuations of specific companies through a supervised learning approach to textual analysis of recently published and relevant articles on the web.

Data Collection and Pre-processing

We limited our scope to examine four prominent tech companies: Apple, Microsoft, Amazon and Tesla. Using python, we scraped the sites of Bloomberg Businessweek, TechCrunch, and the Motley Fool for a month's worth of relevant news articles and press releases based on searches by the company name and ticker symbol. We also computed the stock fluctuation for each company for each day of news collected. We collected a total of 1,123 articles over 30 days. After collecting the articles, we wrote scripts to parse their content, convert all words to lower case, remove punctuation, and stem words to their base form using the Porter stemming algorithm.

Raw Text	Processed Text
HTCs One A9 has been criticized as an Apple iPhone knockoff because of the resemblance in the exterior design. However, the ambitious A9 will hit the store shelves today.	htcs one a9 has been critic as an appl iphon knockoff becaus of the resembl in the exterior design howev the ambiti a9 will hit the store shelv today

Methodology

We treated this as a natural language processing and discrete classification problem where stock price either improved or did not regardless of the percent change. To predict these two classes we had two types of features: word frequencies and numerical measures derived from sentiment analysis. This problem lends itself to supervised learning algorithms such as SVMs and logistic regression. For each method, we trained our models with holdout cross-validation by training on the data scraped for Microsoft, Tesla, and Apple and testing on the data scraped for Amazon.

SVM

SVMs are among the best "off-the-shelf" supervised learning algorithms. SVM's excel at efficiently handling high dimensional feature spaces because of their use of kernels. Also, their margin maximization usually creates very robust predictions. SVMs are a good starting point for almost any type of classification problem.

Logistic Regression

Logistic regression is the other natural choice for a classification problem. It assumes very little information about the training data, and it tends to do well even with small amounts of data. In addition, it is a very simple learning algorithm to implement, so it is considered a good first step.

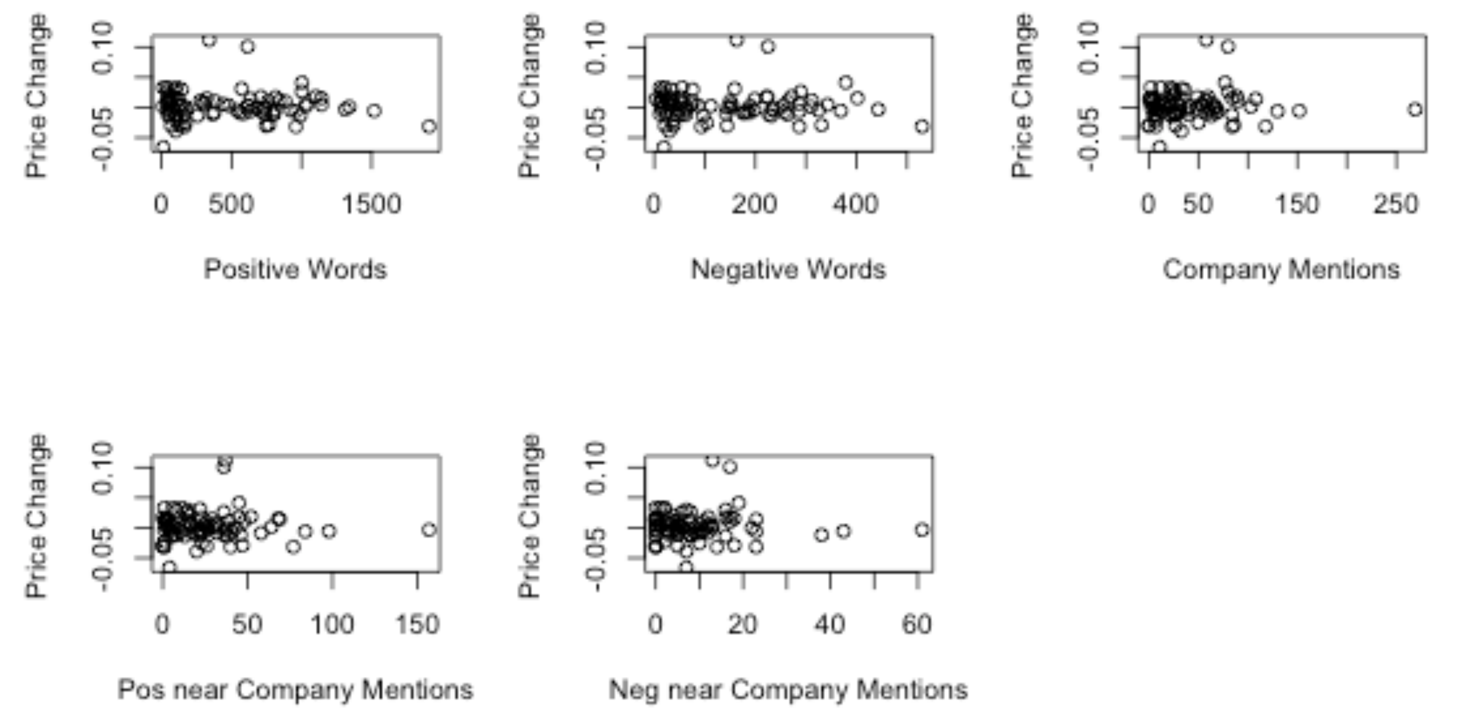
Results

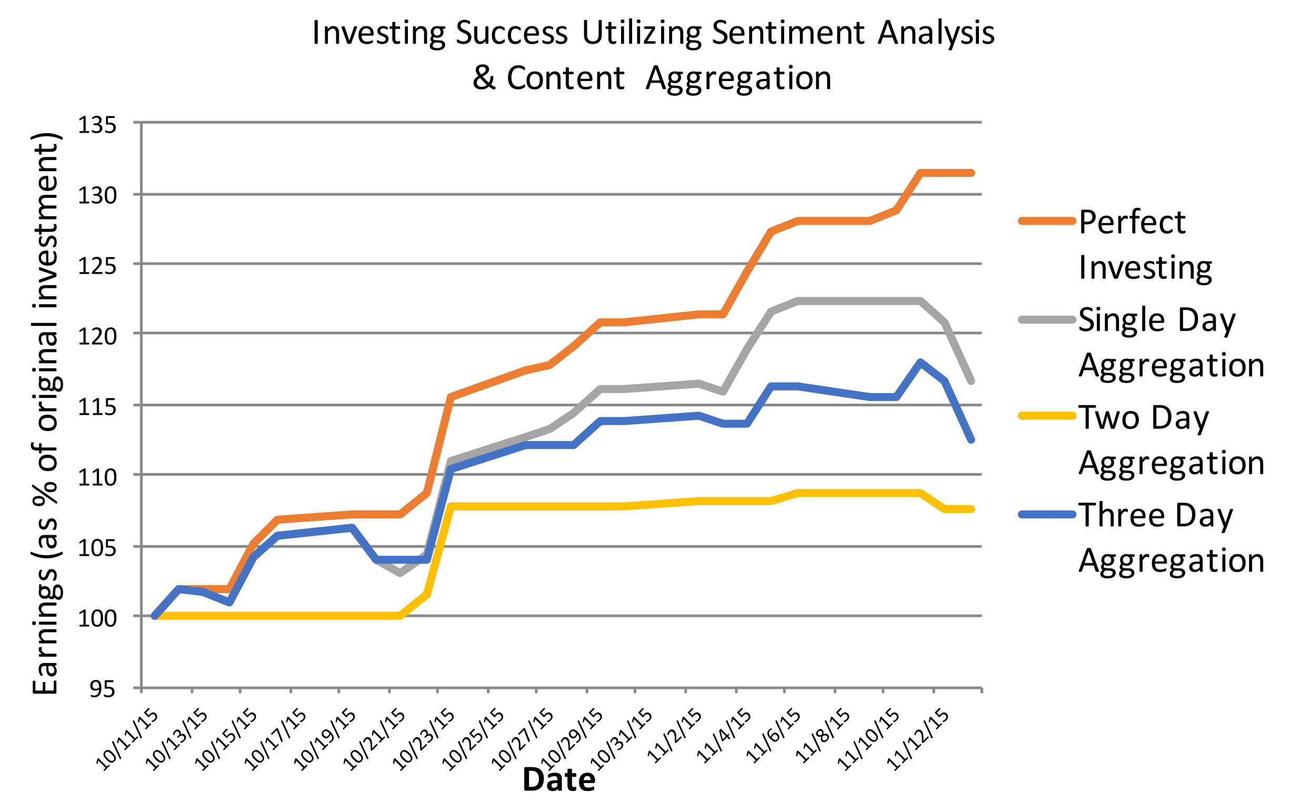
Holdout Cross-Validation Error Rates

Feature Selection (1st Approach)	SVM	Logistic Regression
250 Most Frequently Used Words	0.4821	0.4502
100 Most Frequently Used Positive and Negative Words	0.5179	0.3865

Feature Selection (2nd Approach)	SVM	Logistic Regression
Non-aggregated Content	0.5817	0.5498
Aggregated Content over 1 Day	0.4483	0.4138
Aggregated Content over 2 Days	0.5769	0.4230
Aggregated Content over 3 Days	0.4783	0.4783

Selected Feature (2nd Approach) Relationships





Perfect investing represents earnings with complete prior knowledge of stock price fluctuations, buying and selling for days where prices were to increase, and doing nothing for days where prices decreased or were stagnant. The aggregation plots represent earnings given the predictions made by our SVM with the feature subset created by our 2nd Approach to feature selection.

Feature Selection

We incrementally tweaked our selection of features with two approaches of choosing subsets of features. In our second approach, we also experimented with aggregating content.

First Approach

The first was a "bag of words" approach similar to how we implemented spam detection in class. We created a lexicon of words using frequency tables, and then ran our supervised learning algorithms. Limiting our features – in this case tokens – was our primary concern in this approach. Below are different methods we used to select features, which we then used holdout cross-validation to find test error estimates.

1. Created a lexicon of the 250 most frequently used stemmed tokens across all articles, ignoring numeric values and "stop" words such as "the," "a," "as," etc.,
2. Created a lexicon with sentiment analysis in mind by using an existing sentiment lexicon to include only the 100 most frequently used "positive" and "negative" tokens (ignoring "neutral" ones). We sought to test our hypothesis that some words were more consequential than others. Examples of positive and negative tokens (stemmed to be consistent with our processed data) are: "accomplish, amaz, matur, rich…" (positive) and "abolish, accus, lose, poor" (negative)

Second Approach

Our second approach used as features the percentage of positive tokens in an article, percentage of negative tokens, percentage of the relevant company's name or ticker symbol mentions, the percentage of positive tokens in the sentences immediately surrounding a company name/symbol mention, and percentage of negative tokens in the sentences immediately surrounding a company name/symbol mention. Here, we also took into account bigrams such as "not profitable" or "no revenues" to make our sentiment analysis more sensitive to the meaning of pairs of words in context.

Aggregating Content

We found that, on average, the length of any given article was approximately 600 words. Given that many of those words would be ignored by our feature selection of either most frequently occurring words, positive or negative words, and non-stop-words, we hypothesized that a single article did not have enough information to make an accurate prediction. So, we took three approaches to aggregating content to make predictions not on the information of a single article, but on the information of the content of multiple articles aggregated together:

1. Aggregate all article content for a company for each day. We did this by summing up all of the frequencies of positive words, negative words, and company mentions over all of the articles for each day and for each company. Make predictions for the next day given all content for the current day.
2. Aggregate all article content for a company for each day and the previous day, then make each prediction for the next day given its previous two days of content.
3. Aggregate all content for a company for each day and the previous two days, then make each prediction given three total days of news content.

Caveat: Aggregating data greatly shrank our training data set by roughly a magnitude of 10 since it reduced our number of observations to the number of days over which we collected data. Hence, our results from the aggregated data had higher variance than the other methods.

STANFORD UNIVERSITY