Predicting the Stock Market with News Articles

Kari Lee and Ryan Timmons
CS224N Final Project

Introduction

Stock market prediction is an area of extreme importance to an entire industry. Stock price is determined by the behavior of human investors, and the investors determine stock prices by using publicly available information to predict how the market will act or react. Financial news articles can thus play a large role in influencing the movement of a stock as humans react to the information. Previous research has suggested that there is a relationship between news articles and stock price movement, as there is some lag between when the news article is released and when the market has moved to reflect this information. We built a text classification system to categorize news articles containing references to a list of companies and predict the impact of the news on the stock price. Our trading system to act on these predictions outperformed a baseline strategy of simply holding on to equal amounts of the stocks in question for the test time period.

Previous Research

Gidofalvi trained a naive Bayesian text classifier by scoring news articles using a function combining the change in price with the β-value, which measures the volatility of the stock. The articles were given one of three labels based on the stock's movement compared to its expected movement. Using this method, the predictive power of the classifier was limited, but there was a strong correlation between the news article and the stock price behavior within a 20 minute window around the news article's release time.

Fung, et al designed a system which used a t-test based split-and-merge piecewise linear approximation algorithm to filter out the noise of the movement over the stock time series. Features were the words in a document weighted using term frequency inverse document frequency, and the optimization problem was solved using a Support Vector Machine. The resulting system was most stable and appropriate for predictions within 3-5 days.

Algorithms

Bag of Words

We generated four different word distributions for each company in question based on proximity to the company reference in the article - the paragraph containing the reference, the paragraph preceding the reference, the paragraph after the reference, and the rest of the article. The word counts for each distribution were incremented for every reference in each article. The word counts were weighted by the stock price change for the article date as well as by the weight of the paragraph (e.g. weighting the paragraph containing the reference higher).

Algorithm to compute relevance for each word in training:

foreach (article *in* training set)
    foreach (word *in* article) word relevance = *Sum* over paragraphs (word count for paragraph * weight for paragraph)
    end foreach
    Total relevance = *Sum* over all words in article (word relevance)
    Multiple each word's relevance by (daily % change for stock / Total relevance)
    Add each word relevance to total score for word
end foreach
foreach (word *in* data)
    normalize - divide word's score by number of articles word has appeared in, to get an average score per article
end foreach

The output is a map of (words, scores). When making a prediction on a test article, the algorithm is:

score = 0
foreach (word in article)
    score += (word score) * Sum over paragraphs (word count for paragraph * weight for paragraph)
endforeach

The returned score is the predicted daily percentage change for the stock in question. We chose this way of weighting the scores so that if a predictor was trained on exactly one word bag and then asked to predict based on that word bag, it would return the actual change for that day exactly.

Implementation

We used New York Times articles from the EnglishGigaWord corpus, which spans the timeframe July 1994 – December 2004. There was no tag to mark financial articles specifically, but we separated out any articles of type "story" which contained a reference to a company name on a modified list of the Dow Jones Industrial Average (ignoring companies which were not publicly listed during the timeframe in question). We tokenized the articles by splitting on any non-word character and lowercased everything. Company name references were determined before the lowercasing to take advantage of proper-noun capitalization.

Stock price information was downloaded from Yahoo!Finance, which gives the historical opening and closing prices (adjusted for splits). Because this was the finest-grained data easily available to us, we used a daily window in attempting to correlate news articles to stock prices.

NewsReader – primary class for the project, reads in the data and applies the appropriate filters to the filenames, creates an instance of the appropriate algorithm and runs the predictor
NewsArticle – converts an article in which all the paragraphs are single strings into usable form for the algorithms
WordBag – contains the relevant word distribution and stock information for a particular article

Predictor – trains and simulates trading based on the Bag of Words algorithm
MaxEntPredictor – trains and simulates trading based on the MaxEnt algorithm
Stock – keeps track of the stock price information for a particular company
MarketData – loads the stock data and allows lookup based on ticker or company name

Testing Methodology

We created a separate test set by randomly choosing one or two months from each year. Because of memory limitations we could usually not run across the entire corpus, but we were able to run in sets of 1-4 years at a time. The ultimate purpose of the classification was to use the information to trade stocks and hopefully beat the market, so we evaluated based on the results of two simulated trading systems.

1) Simulated trading based on our news article predictions – each day a news article referenced at least one company in question, the "money" for that day was divided among the stocks referenced. We assumed that we would either buy at the opening price and sell at the closing price if we predicted that the stock would go up, or short the stock at the opening price and buy at the closing price if we predicted that the stock would fall. In addition, we also assumed that broker fees would be negligible in comparison to the amount of money being traded. We tried three different methods of dividing the money among the stocks for each day.
    a. Equal amounts per stock reference – this would be equivalent to weighting a stock higher if it was discussed a lot in the news
    b. Equal amounts per stock – every stock mentioned per day received an equal allocation of money
    c. Division based on predicted profit – stocks were allocated a share of the day's money proportional to their predicted percentage change in price
2) Baseline trading – all the stocks on the list received an equal share at the beginning of the month, held for the entire period, and sold at the end regardless of any change in price.

In order to have a relevant comparison between the baseline trading and the simulated trading, the test articles had to be from a contiguous time period. Our test set consisted of various month-long time periods in which trading was simulated using both methods and then averaged across all the test-months.

Maximum Entropy Approach

We also attempted to use a prediction approach based on a maximum entropy classifier (implementation gleefully borrowed from the third homework, at the price of template-induced insanity to convert our data to the appropriate forms). For this we used three labels for words in articles, ones denoting positive, negative, or neutral change in a stock over the course of one day. The neutral label serves as the unlabeled class. When training, we look up the change in the stock on the corresponding day, and every word pulled from the article gets the same label; neutral if the absolute value of the percentage change is less than a threshold (generally we used), positive if above and negative if below the negative threshold. We construct a different maxent classifier for each company.

We experimented with different ways to extract word-label pairs from articles. One approach is to use only the paragraph in which the company appears. Using the entire article was generally not feasible as memory requirements ballooned, so efforts were focused more narrowly. Unlike the bag of words approach, precise weighting was not possible as the maximum entropy classifier treats all input pairs equally. However, a crude form of weighting is possible by repeating parts of the article; for example one could train and classify on the paragraph before the company reference and two copies of the paragraph with the reference. Due to time constraints involved in training about 25 classifiers, we limited the training to 40 iterations.

To make a prediction, we combined all articles for a given stock and date, then label each word in this combined test data using the maxent classifiers. We first tried a simple method of taking a simple plurality vote; buying the stock if the positive label has the most occurrences, shorting if the negative label is most prominent, and doing nothing if neutral is most common. In this system we would need to use equal amounts of money for each stock (equivalent to testing method 1b above) since the predictions are discrete and have no weighting. A second method using weighting computes a score as (# positive labels - # negative labels) / (# total labels), which will be a value between -1 and 1. The absolute value of this was used as a measure of confidence in our prediction and allowed us to allocate our daily investment based on predicted profit (Testing Methodology 1c). However, in contrast to the bag of words method this score is an abstract prediction, not a predicted percentage change.

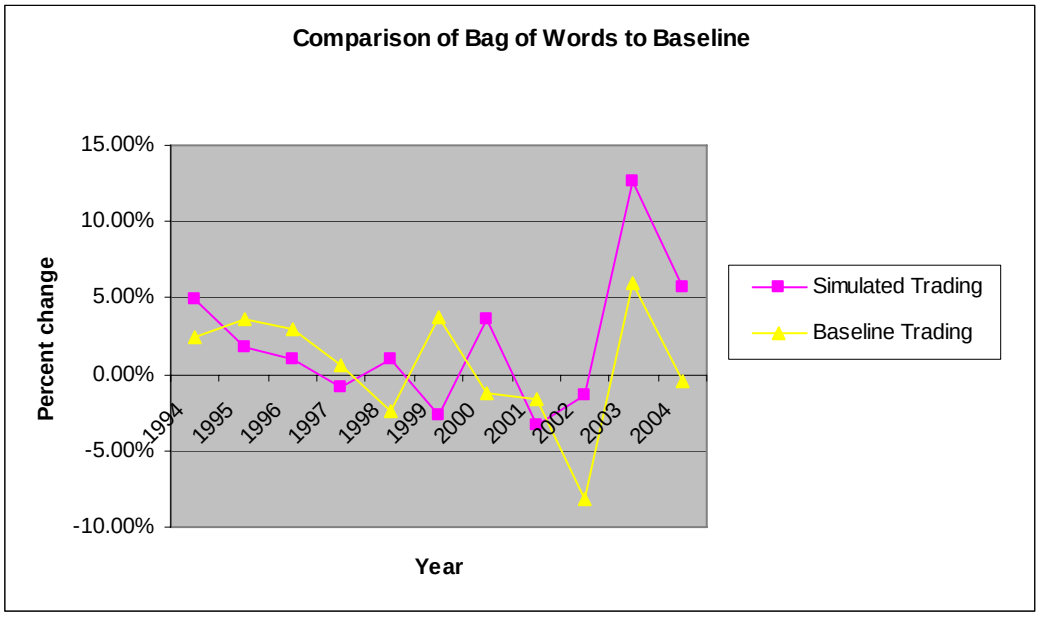
We experimented with different features a bit on the smaller test sets and came up with the following list: the word itself, adjacent words up to two positions away, beginning and inner capitals in the word, and if the word contained a number/digit. Unfortunately time did not allow extensive testing of different feature combinations on the whole dataset. Initially the label of the previous word was included in the features but we removed it for reasons that will be discussed in the results section.

Results

Bag of Words

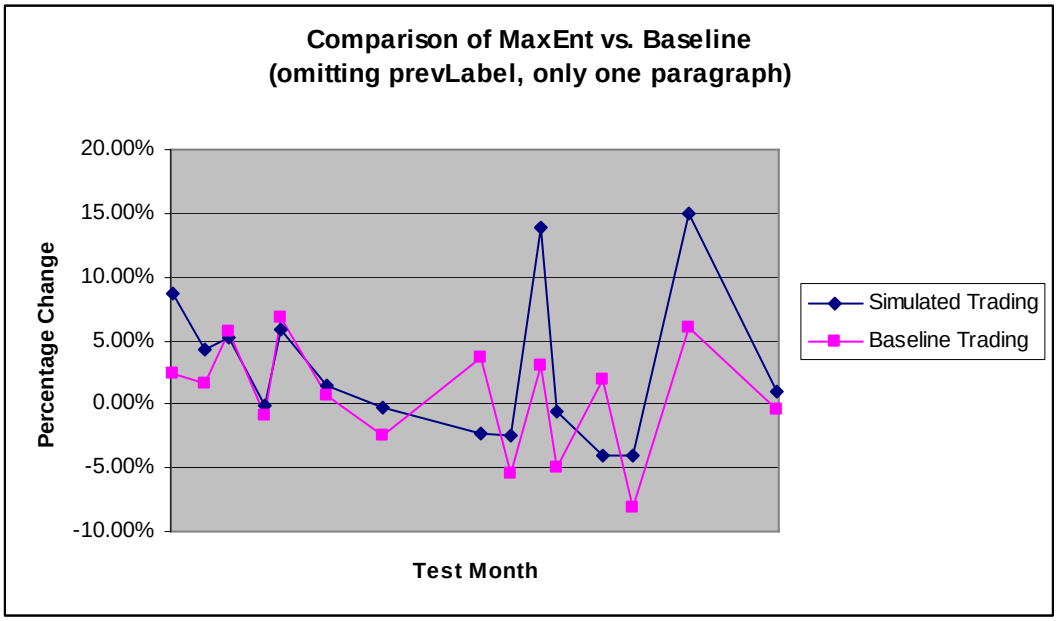
We were able to run the bag of words on one year at a time (training on all train data for that year and test on the one-two months of test data) as memory constraints prevented training on the whole set simultaneously. Combining these results gave an average of 2.05 % return per test month for the bag of words, vs. 0.615 % per month for the baseline method. Fig.1 is a graph of the yearly comparison between the simulation and the baseline (see next page). It is interesting that while the average for the simulation is significantly better than the baseline, the year-to-year comparison is very inconsistent, with the simulation underperforming the baseline as often as not. Unfortunately, this makes experimenting with parameters such as the weights for the paragraphs difficult as the entire training and test sets must be tried; it would seem dangerous to tune parameters to a subset of the training set with validation data given that performance is so variable.

Fig. 1



Maximum Entropy

Fig. 2



First, let us discuss one alternative to the features list for the maximum entropy classifier. We tried removing the label of the previous word. This was motivated by our training data labeling method; since every word in an article got the same label the previous label would be a 100% accuracy method of predicting the current label in the training data. This struck us as bad, and not surprisingly the results of the trading simulation were considerably better when this feature

was removed. When trained over the entire data set, the maxEnt system (using just the paragraph with the company reference as the data), averaged over all months in the test set, produced an average return of 0.63 % per month, slightly higher than the average baseline return of 0.615 % per month. However, with the previous label removed from the features list, the same trial produced an average return of 2.77 %, quite an improvement (and better than the bag of words approach). Fig. 2 on the previous page shows comparisons of the simulated return vs. the baseline return for this version of the maxent classifier method. Notice that while omitting prevLabel makes it much better on a month-by-month basis to the bag of words, on certain months it still does worse than the baseline. Also notice that Fig.1 contains data averaged by year (years generally had 1 or 2 test months) so there are slightly fewer data points than Fig. 2.

We also experimented with using more than just the paragraph surrounding the company reference. As previously mentioned, using the whole article was too memory intensive to be practical, but we were able to run with one copy of the paragraph before the reference, two copies of the paragraph with the reference and one copy of the paragraph after. However, even this greatly increased the memory requirements and it was no longer possible to run the whole set at once, so we tried 1994-1995 and 2001-2004 (time constraints prevented more complete testing). Here are the results:

	Average Performance per Month		
Year range	MaxEnt, 1 paragraph	MaxEnt, 1-2-1	Baseline
1994-95	0.06035	0.02921	0.03221
2001-2004	0.01479	0.03695	-0.01136

The variability curse strikes again, as we have one result where 1-2-1 is significantly better and one where a single paragraph is.

Conclusion

Of the two methods we tried, the maximum entropy classifier was superior. It is unclear whether a single-paragraph method or 1-2-1 method is better, more testing should be done. However, there are significant tradeoffs between the systems. As more paragraphs are used for the maxEnt method, the memory use goes up quite a bit and the convergence time does as well. In addition, to make a detailed comparison of single paragraphs vs. multiple paragraphs in the maxEnt method, it would be good to hold a constant-size memory comparison (so the single paragraph, which uses less memory and processing time per article, gets more training data). The bag of words is generally faster to run (primarily since it's training method does not need to iterate like the classifiers does) but takes more memory, so there is worse hard limit on the size of data that can be trained at once. Both methods give substantial improvements over the baseline trading system, but for reasons discussed in the trading methodology section and in the final section, that may not transfer to real-world success.

Future Work

Our research could almost certainly benefit from more work with the corpus. As was previously mentioned, our news corpus was not separated into financial news, and while we separated the articles based on our company names there are many types of articles that could get in without

actually relating to business at all (such as any article mentioning people drinking a Coke or eating McDonald's food). There's also the danger of picking up after-the-fact stories like "Microsoft stock rose $2.50 today." Also our assignment of the training and test sets were broken up by months; given more time we could break up the test set on a daily basis.

Another improvement to the data would be to get price data for stocks at greater resolution than the daily opening and close along with articles with time stamps as well as dates. This would allow us to both measure and predict the price changes in the immediate vicinity of when a news article appears, like Gidofalvi's 20 minute window. In order to do this, it would probably be necessary to write scripts to pull articles from news websites and prices from financial websites as information with this time detail does not appear to be readily available. This also means waiting for a few months to build up data instead of accessing an existing corpus. All these limitations definitely limit the extent to which we think our system would be successful in the real world.

Yet another thing to try would be to train the paragraph weights for the bag of words method; again due to the length of the training we had to experiment with a few different choices and couldn't try very many. The bag of words is pretty quick to train (the major program time is reading the files), so it may be possible to have a gradient descent on the 4 weights in a reasonable amount of time but we didn't have the programming time to try this.

Other methods of baseline comparison could be used also, although the current one is a reasonable benchmark since many people invest in funds that simply index a major set of companies like the Dow. It would be nice to get the variety of the S&P 500, but given that we ran up against memory issues on the 30-company collection that's not likely to be practical. It could be possible if we didn't use a separate classifier for each company; a study would need to be done on the current data set to see how much performance is lost by using the same classifiers for multiple companies. On the other hand, it would be interesting to apply this to non-blue-chip companies, but selecting an appropriate set with enough news data to train on could also be problematic.

References

Gidofalvi, Gyozo. *Using News Articles to Predict Stock Price Movements*. Department of Computer Science and Engineering, University of California, San Diego. 2001.

Fung, Gabriel, et. al. *The Predicting Power of Textual Information on Financial Markets*. IEEE Intelligent Informatics Bulletin. Vol. 5. No. 1. June 2005.