Predicting Stock Trends Using Natural Language Processing of Headlines

Team 11

1 Introduction

Do the headlines of news articles influence stock prices? Many people are very interested in the answer to this question and related questions because the money that can be made in the stock market. Determining the correlation between news headlines and stock prices would have a big impact on society. Many people depend on the successful investment of stocks for their financial future and knowing how stock prices will react to certain news headlines could greatly increase investment income.

We examine both the news headlines and the stock history relating to America's top 50 largest companies for a 15-week time period (1/1/2012 to 4/14/2012). The goal is to determine if a given headline will cause a stock to rise, fall, or remain steady during the trading day after the headline is released.

2 Related Work

Hidden Markov Models have a long history of application in the area of text classification, both in academic literature e.g. *AI: A Modern Approach* by Russel and Norvig [5] as well as more detailed treatments including "Text Categorization using N-grams and HMMs" by Mathew [3]. As presented in "Text Mining Systems for Predicting the Market Response to News" [4], prototype systems in this arena often use n-gram language models and Naïve Bayes or SVM classifiers to predict market responses over a time frame of minutes or hours.

We intended to focus more specifically on the application of HMMs and N-gram language modeling for classification of news headlines as presented by Hassan and Nath in "Stock Market Forecasting Using Hidden Markov Model: A New Approach" [2]. This work treats the previous day's closing price as the observed variable and predicts the following day's closing price. We sought to improve on the performance of such systems (typically 40-45% vs 33% with random guessing as presented in [4]) by instead using the tone of news headlines as emitted variables in our HMM.

3 Approach

3.1 Language Model

The first step in the implementation of this project was to obtain both news headlines and historic stock price data for the companies of interest for the given time frame. This information was gathered from the Yahoo Finance website, using Yahoo's URL syntax, and imported into a relational database.

It was impossible to gather data from the period originally planned since Yahoo does not provide historical headline data to the general public prior to about November 2011 as of the time this project was implemented. Consequently, data was gathered for the 15-week time period from roughly 1/1/2012 through 4/14/2012. In addition, version 1 of the Yahoo News REST API was deprecated in April 2011. Data import was eventually accomplished by issuing queries in Yahoo Query Language (YQL) via the current Yahoo News REST API using Java, extracting the relevant text using XPath, and finally exporting the data to a MySQL database for mining. Access to historical stock data is provided by simple HTTP interface, allowing this data to be retrieved using `wget` and a relatively simple bash shell script.

The headlines were then sorted at random into two data sets on a per-headline basis: training (60%) and validation (40%). The training set was used to develop the n-gram models and train the headline classifier. The evaluation data set was reserved for 5-fold random sub-select validation (see section 4).

3.2 Classifier

The classifier was intended to be a first-order HMM with hidden states consisting of the state of the relevant company, discretized into five different values ranging from strongest to weakest. HMM are thought to be a suitable method for this problem because it is assumed that both news headlines and stock prices are a reflection of the condition of a company. Therefore, this condition is the hidden state and the headlines and the stock prices are the observed emissions for an HMM. Moreover, it seems to be a valid assumption that the condition of a company can be assumed to only depend on its condition in the previous time period (i.e. first-order Markov assumption). Unfortunately, the expectation-maximization (EM) algorithm used to learn the transition probabilities and emission probabilities for the HMM did not converge for the several thousand emitted n-grams and stock price changes after running the algorithm for eight hours. Our implementation of the EM algorithm for solving HMMs was not optimized for such a large number of variables. The approach of using an HMM had to be abandoned in favor of an approach that learns weighted percent price changes for each headline n-gram. This learning weights approach assumes each n-gram has some affect on the company's stock price. The amount of this affect is captured in the weights, which are learned. As shown in the following section, this strategy was nonetheless a distinct improvement verse random guessing.

To learn the n-gram weights a sparse binary $m \times n$ matrix, A, is constructed for both the training and validation data. There is a row for every unique n-gram ($m = 367513$) and a column for each headline (training $n = 4958$, testing $n = 2479$). Each element of the matrix specifies which

n-grams are in each headline. In addition, for a given headline about a company there is a stock price change (as a percentage) for that company on the following trading day. For all the n-grams in the training data this percent price change is averaged across all headlines and stored in an $m \times 1$ vector, B. For both the training and validation data the true percent stock price changes for the following trading data are stored in an $n \times 1$ vector, C. In addition, the number of n-grams in each headline is stored in an $n \times n$ diagonal matrix, D. The weighting matrix that we solve for is an $m \times m$ diagonal matrix, W, such that

$$A_{train}^T W B_{train} = C_{train} D_{train}. \quad (1)$$

Obviously, (1) is underdetermined because there are m weights to learn and only n samples ($m > n$) for learning. Thus, W is solved for using a least squares regression fitting. Another words,

$$W = \underset{W}{\operatorname{argmin}} \|A_{train}^T W B_{train} - C_{train} D_{train}\|^2. \quad (2)$$

To actually compute W a Moore-Penrose pseudoinverse of A_{train} and B_{train} are calculate (A^+ and B^+, respectively) and multiplied to $C_{train} D_{train}$. This is gives a least squares estimate of W. Thus,

$$W = A^+ C_{train} D_{train} B^+. \quad (3)$$

4 Evaluation

4.1 Language Model

The language model used to train the classifier consisted of a set of unigram, bigram and trigram frequencies for each dataset, along with average stock price trends for all headlines in which each n-gram appears. The following tables are an excerpt of the most positively and negatively weighted bigrams for Dataset #2.

Table 1: Trigram Model - Falling Stock Prices

N-gram	Count	Avg % Price Change
freddie mac supported	1	-12.5000
puzzled over derivatives	1	-12.5000
supported industry with	1	-12.5000
<unk> holding back	1	-9.6774
to <unk> foreclosures	1	-9.0909

Note that the column of counts in the n-gram table presented above gives the number of occurences in unique headlines, not n-gram frequency in the corpus as a whole. This was done to avoid skewing the n-gram/price correlation when a particular headline is published multiple times. Consequently, it is unsuprisingly that many trigrams appear in only one headline.

In addition, each set of n-gram models was used to analyze the perplexity of the validation dataset corresponding to each training dataset. As defined in *AI: A Modern Approach* [5], perplexity is calculated as

Table 2: Dataset #2 Trigrams - Rising Stock Prices

N-gram	Count	Avg % Price Change
instinct to win	1	25.0000
obama proposes broader	2	25.0000
of housing assistance	3	25.0000
to broaden reach	3	25.0000
your next <unk>	1	25.0000

$$Perplexity(C_{1:N}) = P(C_{1:N})^{\frac{1}{N}} \quad (4)$$

The following perplexity measurements would prove insightful when evaluating the results of the classifier (see 5).

Table 3: Validation Dataset Model Perplexity

		Perplexity		
	N-grams	1-Gram	2-Gram	3-Gram
Dataset 1	525424	421.1	29.3	7.9
Dataset 2	523227	422.3	29.3	7.8
Dataset 3	521141	422.5	29.0	7.8
Dataset 4	522396	420.5	29.0	7.8
Dataset 5	521497	422.3	29.2	7.9
Average	522373	421.7	29.1	7.9

4.2 Classifier

To evaluate the performance of the classifier, the predicated stock price changes, $\hat{C}_{test}$, were estimated and compared against the true changes, C_{test}, from the validation dataset. To do this stock price changes were categorized into three classes: "short", "steady", or "long". If a stock price fell by more than 0.25% then it is a "short" stock. If a stock price rose by more than 0.25% then it is a "long" stock. Lastly, if the stock stayed within -0.25% and 0.25% it is a "steady" stock. These thresholds were chosen to give approximately equal number of samples in each category. To determine the class of the validation data predicted stock prices changes were calculated as

$$\hat{C}_{test} = A_{test}^T W B_{train} D_{train}^{-1}, \quad (5)$$

where W is the weights learned from the training data.

The results of all three n-gram classification models for all 50 companies were aggregate into a single confusion matrix shown in Figures 1 and 2. The training and testing process of the three n-gram classification model takes several hours to run. Due to time constraints we were not able to run N-fold cross-validation on 3-gram classification model. However, we were able to do it for just the 1-gram classification model because this ran fairly quickly. These results are summarized in Table 4.

5 Discussion

It is impossible to accurately compare these results with Fung et. al [1] for several important reasons. First, we were

Table 4: Mean and standard deviation for percent correct classification using 5-fold cross-validation on 1-gram data.

	Short	Steady	Long
Mean	58.9%	38.4%	64.5%
Standard Deviation	1.6%	0.7%	1.5%

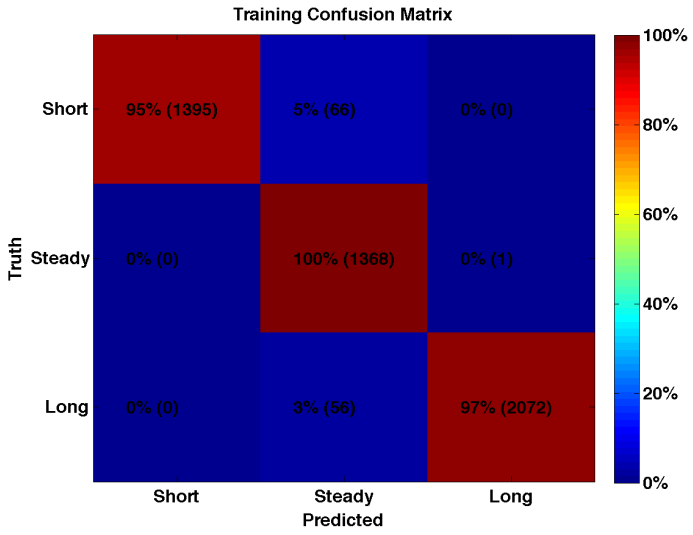

Figure 1: Confusion matrix of training data.

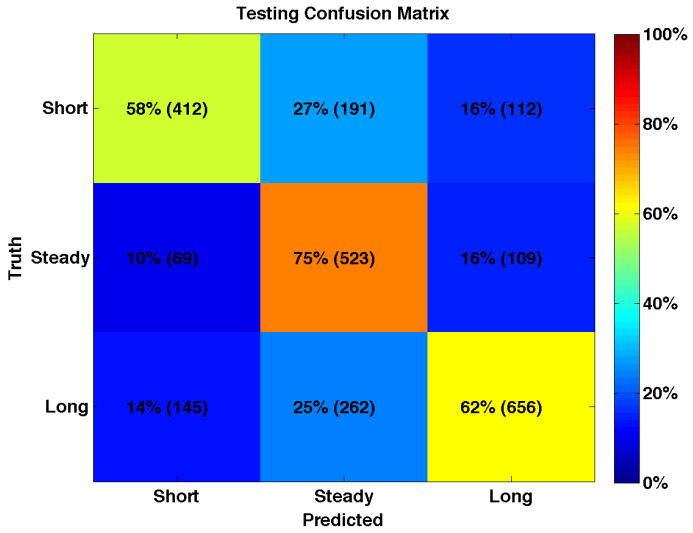

Figure 2: Confusion matrix of validation data.

unable to use headlines from the same time period since, as noted previously, Yahoo News no longer provides these headlines. Second, our study evaluates only the effectiveness of the classifier, whereas Fung et. al also evaluated an automated trading system based on SVM and Naïve Bayes classifiers. Although it is possible that the excellent performance of the classifier was a statistical outlier given the small number of cross-validation runs, we initially suspected over-training of the n-gram models. Table 3 shows the perplexity of the validation data for 1, 2 and 3-gram models trained of 5 randomly selected subsets of headline data.

As described in [5], perplexity is in essence a measurement of the branching factor of the language model. A 1 in 8 chance of picking the next trigram in a sentence given a predecessor n-gram is clearly an over-specified model for most languages. We attribute this result to substantial redundancy in the dataset - although headlines were randomly assigned to training and validation corpora, some headlines were repeated, with minor variations, more than a dozen times in Yahoo News over a period of days. Second, given the space constraints of a news headline, the sentences are often shorter and the vocabulary condensed as compared to prose or spoken English. These factors likely reduce the variability in the language studied here.

An unexpected pitfall of this algorithm's design was the substantial time necessary to generate the sparse matrix of headline/n-gram counts prior to training the classifier. The language model/classifier integration could be improved by simultaneously generating the headline/n-gram sparse matrix while building the n-gram language model, instead of repeating the process of parsing n-grams from the headlines during the classifier stage of the algorithm. We did not have time to experiment with this refinement, choosing instead to perform random subselect validation of the classifier built using the unigram model.

Using principle component analysis (PCA) or linear discriminate analysis (LDA) are a couple other approaches that might have been appropriate for this problem. This is because essentially we are trying to classify n-grams into one of the three (i.e. "short", "steady, "long"). It has been shown that PCA and LDA can perform well with this type of classification problem.

These caveats aside, it is clear that the classifier developed for this project dramatically outperforms random guessing. This type of tool could be of great use to a stock trader because it takes only seconds to highlight the most influential positive and negative news stories among thousands.

References

[1] Xu J. Fung, P. and W. Lam. Stock prediction: Integrating text mining approach using real-time news. *Proceedings of the 2003 IEEE International Conference on Computational Intelligence for Financial Engineering*, pages 395 – 402, 2003.

[2] M.R. Hassan and B Nath. Stock market forecasting using hidden Markov model: a new approach. *Proceedings of the 2005 5th International Conference on Intelligent Systems Design and Applications*, 2005.

[3] T. Mathew. Text Categorization using N-grams and Hidden-Markov-Models. http://www.slideshare.net/thomas_a_mathew/text-categorization-using-ngrams-and-hiddenmarkovmodels, 2006. [Online; accessed 2-Apr-2012].

[4] M. Mittermayer and G. Knolmayer. Text Mining Systems for Predicting Market Response to News. *IADIS European Conference on Data Mining*, 2007.

[5] S. Russel and P. Norvig. *Artificial Intelligence; A Modern Approach*. Prentice Hall, third edition, 2010.