CS224n Final Project Stock Price Prediction Using News Articles

Qicheng Ma

June 10, 2008

1 Introduction

The basic form of efficient market hypothesis postulates that publicly available information is incorporated into stock prices. The stronger form of the hypothesis asserts that all information are immediately reflected in the market as soon as they become available; whereas the weaker form allows a certain lag period in which the information is digested by the public and stock prices move gradually towards the efficient target. Previous research suggests that there is correlation between news articles and future stock movement, and several methods have been proposed for making prediction on stock market and devising profit-generating trading strategies.

In this project, building on previous research by falvi and Elkan [1], Fung et al. [3] and a similar final project from previous year by Timmons and Lee [6], we build two models using Naive Bayes and Maximum Entropy classifiers to predict stock price movement based on archived news articles from the Wall Street Journal and the Reuters Financial corpus, and evaluate their effectiveness.

2 Algorithms

2.1 Problem Setting

We formulate the stock movement prediction problem as the following NLP classification task: Given a stock of interest s, for every trading day t we can assign to it a class label $c_{s,t}$, for example its discretized daily percentage return. For every new articles (or segments thereof) d_i that mentions stock s, we have a labeled instance $(d, c_{s,t})$. Our goal is to learn a model of $P(c_{s,t}|d_i)$ and use it to predict future (unobserved) classe labels based on observed news articles.

Once we have trained the classifiers $P_s(c|d)$ for a list of stock in interest, we can then specify a higher-level trading strategy that generates buy and sell signals. Although in practice the stock-movement classification task and the trading strategy is coupled together, we explicitly separate the two system so that we can develope and test the two components separately.

2 ALGORITHMS 2

2.2 Labeling Stock Movements

We used a modified list of 29 stocks from the components of the Dow Jones Indistrial Average, borrowed from [6], each with a list of hand-made names to pick up references in news articles. Historical stock quotes are downloaded from Yahoo! Finance [2] using a freely available batch tool [7]. Similar to [6] we assign to a (stock, day) one of the three classes Positive, Neutral, Negative based on its daily percentage return, with Neutral being the range [-1%,+1%]. We also considered labeling based on the percentage return on the next n-day trading window, or the relative position of that day's opening price in the [low, high] range of the n-day trading window, but due to time constraint we settled with the simplest approach which allows the simplest trading strategy - buy (sell-short) at market open a stock of predicted Positive (Negative) movement and sell (buy-cover) it at market close. The other two labeling rules would correspond to split-day tradig rules that generate buy and sell signals on different days. Most of the stock data importing and pre-processing code is borrowed from [6] with minor modifications.

2.3 Processing News Articles

We re-implemented the news-article processing code to work with the North-American-News corpus, in particular the Wall Street Journal and the Reuters Financial subset¹. These contain articles in the date range July 1994 - Dec 1996. Our implementation also has the ability to work with other corpus (like the EnglishGigaword/nyt_eng corpus, which was used by Timmons and Lee [6]), but we eventually did not use it in our experiments due to time constraints.

Like [6] we identified paragraphs in the articles that contain references to our list of stock using regular expression matching on a hand-made list of names (with some bug-fixes and optimization from their code). Then the articles were tokenized at all non-word boundaries and converted to lowercases. Recognizing that one article may talk about many topics and may involve several stocks, we form (paragraph, label) as our basic unit of instance instead of (article, label). Out of the 190544 articles from the combined WSJ and Reuter corpus, we find 31493 articles containing references to the stocks in interest.

Notice that once the class labeling method is fixed, we can develope and test our classification system iteratively using feature engineering techniques similar to Assignment 3 by purely evaluating the prediction performance, taking the determined labels as ground truth. Such intrinsic evaluation process is later on supplemented with extrinsic evaluation process of scoring simulated trading profit using the classifiers as subsystems.

¹/afs/ir/data/linguistic-data/North-American-News/{WSJ,REUFF}

2 ALGORITHMS 3

2.4 Naive Bayes and Maximum Entropy classifier

Inspired by [3] and [6], we tried the Naive Bayes and Maximum Entropy classifier models. Instead of using Max Ent implementation from Assignment 3, we used the more complete Stanford-classifier-v2.0 package [4] developed by Manning and Klein [5]. Both classifiers can be accessed in the StockPredictor class by calling the useNBClassifier() and useMaxEntClassifier() methods.

Similar to [3], the Naive Bayes classifier learns the conditional class distribution P(c|d) based on every word occurred w occurred in d. Here, d is actually restricted to the paragraph containing the stock reference, rather than the entire article:

$$P(c|d) \sim \prod_{w \in d} P(w|c)P(c)$$

Unlike [6], in which the Max Ent classifier is used to label each word occured in a relavent paragraph, and then taking the pularity votes or weighted averages across all words to determin the class of a stock, our implementation is to use Max Ent to classifier each paragraph as a whole into one of the Positive, Neutral, Negative labels. The intuition is that unlike tasks like POS-tagging, where the class label is defined on the word level and depends more on word-level features like shapes, etc., our classification task is more at the paragraph/topic level. Hence we won't use word-shape-like features, but instead use unigram (word) and bigram as features.²

Things we tried that seem to improve classification performance:

- Pruning away very spare features (occur less than 5 times), which seems to
 help prevent overly attributing the weights to words that almost uniquely
 occur in one instance.
- Remove purely numerical tokens. Due to the way tokenization is done, a lot of broken numerical tokens are present, and are getting random weights that do not have real explanatory value. We expect a single number like "42" would not be meaningful without comparative context. Hence we remove all numberical tokens.
- Tuning the regularization constaint (setSigma() in classifier training). In limited point-testing, 2.0 seems best. Using cross-validation to set this parameter is needed in future work.

Finally, we show in Figure 1 as an example a list of top feature weights from a Max Ent classifier instance trained for the Microsoft stock. We can rationalize many of the top features, like gained(+), settlement(+), off(-), piracy(-), subpoena(-), while others may not make intuitive sense.

²Bigram features were turned off in later testing due to time complexity.

+++	weights	000	weights		weights
court	0.24	wordperfect	0.25	off	0.42
language	0. 23	explorer	0.24	pointcast	0.33
chicago	0.22	video	0.24	notes	0.21
gained	0. 22	while	0.23	express	0.21
practices	0.21	large	0.21	money	0.20
agreed	0.21	retail	0.20	start	0.19
europe	0.20	san	0.20	jackson	0.19
sporkin	0.20	communications	0.20	travel	0.18
javascript	0.20	netscape	0.19	piracy	0.18
appeals	0.19	ms	0.19	nbc	0.18
settlement	0.19	battle	0.19	free	0.17
rose	0.18	wintel	0.18	says	0.17
caldera	0.17	letter	0.18	buy	0.17
logo	0.17	borland	0.18	japan	0.17
specifications	0.17	cents	0.18	preferred	0.16
up	0.17	telekom	0.17	pentium	0.16
results	0.17	interactive	0.17	gates	0.15
earnings	0.17	written	0.17	few	0.15
plan	0.17	networking	0.16	american	0.15
vermeer	0.16	ellison	0.16	subpoena	0.15
venture	0.16	corel	0.16	didn	0.15
european	0.16	dispute	0.15	runs	0.15
buying	0.16	group	0.15	designed	0.15
amp	0.16	bell	0.15	network	0.15
intuit	0.15	mass	0.15	fiscal	0.15
wide	0.15	egghead	0.15	fell	0.14
weeks	0.15	multimedia	0.14	among	0.14
frontpage	0.15	tactics	0.14	president	0.14
global	0.15	ballmer	0.14	another	0.14
player	0. 15	msn	0.14	aug	0.14

Figure 1: Top features and weights for MSFT classifier.

2.5 Trading Strategy

We did not have enough time to explore trading strategies other than the simplest one: buy (sell-short) at market opening some value of a stock for every predicted Positive (Negative) labeled instance and sell (buy-cover) it at market close. Since our instances is one paragraph, we can use voting as a simple form of distributing investment among stocks such that stocks with more trading signals (with more paragraphs being classified as non-Neutral) gets higher portion of the total investment, similar to method (1c) mentioned in [6]. Formally, suppose our classifiers emsamble generated a list of s_i, c_i signals, where s_i is stock ticker and c_i is +1for Positive (buy) or -1 or Negative (short). Then we poll all signals for the same stock to get the trading weights for each stock $w_s = \sum_{i:s_i=s} c_i$, and then distribute the fixed investment amount for that day in proportion to the normalized weights $\tilde{w}_s = |w_s|/\sum_s |w_s|$ Note that the absolute value is neccessary because we assume shorting a stock needs the same value of collateral. Then the daily return rate is simply $r = \sum_s w_s r_s / \sum_s |w_s|$ where r_s is the return for stock s. Since we assume a fixed daily investment budget, the monthly return is the sum of return over all days, resulting in a percentage in terms of the fixed daily budget. The baseline method is a simple buy-and-hold for each stock, hence the comparable return is the average return of all stocks in the list over the testing horizon.

2.6 Code Organization

- cs224n.contrib/ data import code borrowed from Timmons and Lee [6] with modifications.³
 - Stock reads in individual .csv quotes downloaded from Yahoo!Finance.
 - MarketData manages all stocks and annotates stock references.
 - NewsArticle read in and process news article corpus.
- cs224n.newcontrib/ all new code
 - NewsArticleDatum represents our instance datum.
 - StockPredictor builds both NaiveBayes and maxEnt classifier models and run trading simulations and evaluations.
 - StockPredictorTester is the main class to run tests on the system.
- starnford-classifier.jar classifier library from [5].
- Other code from PA3 base.

3 Experiments and Results

To evaluate system performance, we use cross-validation style simulations. For each simulation run, we put aside 1 month data (stock quotes and articles) from the 3-year window as the test set, train the classifiers for each stock over the rest of data, and test the classification performance and trading performance in the one month window. This is repeated for each month in the 3 year. We perform intrinsic evaluation of the classification task as well as extrinsic evaluation by recording the average monthly return. Table 1 shows the average Precision, Recall, FB1 score, average monthly return, the 95% confidence interval of the return based on 30 observations, and the volatlity (standard deviation) of the strategy. Figure 2 shows the similated monthly return for each month.

The precision and recall is defined by treating Neutral classes as unlabeled, i.e.

$$Precision = \frac{N_{+/+} + N_{-/-}}{N_{*/+} + N_{*/-}}$$

$$Recall = \frac{N_{+/+} + N_{-/-}}{N_{+/*} + N_{-/*}}$$

where $N_{T/P}$ means number of instances with true label T but predicted label P.

Note that according to Table 1, MaxEnt improves Precision but degrades Recall drastically compared to NaiveBayes, resulting a lower FB1 score. However, the

³Special thanks to them for sharing the code

	Р	R	FB1	return	confint	volatility
Baseline				0.026	0.009	0.026
Naïve Bayes	0.246	0.386	0.299	0.026	0.013	0.036
MaxEnt	0.279	0.184	0.220	0.052	0.015	0.041

Table 1: Summary statistics.

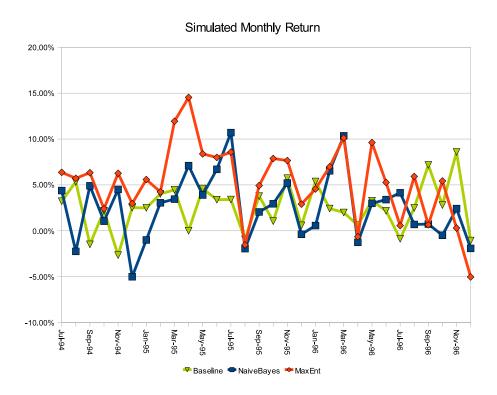


Figure 2: Simulated monthly return of Naive Bayes and Max Ent vs Baseline.

trading performance is actually much better (5.2% average return vs 2.6%). This suggests that in this application domain, Precision is more important than Recall, because it is more important that most trading signals result in profitable actions, than discovering more profitable opportunities with the risk of false positives.

Note that Naive Bayes is actually worse than Baseline in terms of trading performance, achieving the same average return but with higher volatility in the return curve. In contrast Max Ent gets much higher return despite slighly higher volatility. Using the 95% confidence intervals on the average return⁴ we can claim that difference in average return between MaxEnt and Baseline are statistically significant.

⁴They should combined in a root-sum-of-square way.

4 Conclusion and Future Works

In summary, we built and tested two systems to predict and trade stocks based on Naive Bayes and Maximum Entropy classifiers of new articles, inspired by previous research work combined with new approaches. The Max Ent system significantly outperforms the baseline buy-and-hold strategy. This is a proof-of-concept of what could become a larger real time trading system that generates signals based on a stream of news articles to extract higher-than market returns from the lag in the efficient market hypothesis. Due to time constraint, we couldn't out many other new ideas or perform more extensive evaluation. Let us conclude by listing some improvement points and untried ideas perfect for future work.

- Instead of building a classifier from scratch for each stock, we may try building a generic prior model by training on articles with individual stock reference replaced by generic STOCK. Such generic model would learn general rules like "STOCK quarterly earning exceeds expectation" that apply across the board. Then we could build either a separate stock-specific component combined with that, or train based on the generic prior to pick up stock-specific features.
- In our evaluation process that test on one month, there is still the danger of using articles "from the future" to learn that some keyword when it first appeared can trigger a big stock movement. Ideally we should only use past articles as training data and have a sliding window going forward with the test month being always the last.
- For simplicity we were trying to predict absolute returns of stocks. most financial literature suggest to use excess return (return of stock minus return of market index), or the log excess return. Also, further financial and numerical preprocessing and analysis needs to be done, e.g to take into account the variable votability across stocks, so that the range of Neutral is not fixed a-prior.
- We have tried to predict the discrete three-class movement of stock, mostly due to the limited amount of data. We probably want to predict the real continuous value of any return signal using real-valued models.
- More data, more frequent data, and better data! We want to get higher
 frequency stock quotes, preferably real time, and more streaming news source
 to get up-to-date articles, since the essence of the system is beating the general
 public to digest stock-moving information. We could try real time online
 quotes and news articles feeds from search engines as a next step.

REFERENCES 8

References

[1] G. Gidó falvi and C. Elkan. Using news articles to predict stock price movements. Technical report, Department of Computer Science and Engineering, University of California, 2003.

- [2] Yahoo! Finance. http://finance.yahoo.com/.
- [3] Gabriel Pui Cheong Fung, Jeffrey Xu Yu, and Hongjun Lu. The predicting power of textual information on financial markets. *IEEE Intelligent Informatics Bulletin*, 5(1):1–10, 2005.
- [4] Christopher Manning and Dan Klein. Stanford-classifier-2.0. http://nlp.stanford.edu/software/classifier.shtml.
- [5] Christopher Manning and Dan Klein. Optimization, maxent models, and conditional estimation without magic. In NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, pages 8–8, Morristown, NJ, USA, 2003. Association for Computational Linguistics.
- [6] Ryan Timmons and Kari Lee. Predicting the stock market with news articles. CS224n Final Project, 2007.
- [7] Tradery.com. Yahoo google quotes downloader. http://www.download.com/Yahoo-Google-Historical-Quotes-Downloader/.