Stock Prediction From News Media Reactions

Hawkeye Asset Management
Parker Woodworth
Teddy Knox
Will Potter

Abstract

The interest of the financial sector in automated trading algorithms has given rise to entirely new fields of practical study in the disciplines of computer science and quantitative finance. In most circumstances, the trajectory of the stock market is correlated with the investor sentiment, which is typically derived from analysis of financial news and metrics. We aimed to model market sentiment by the supervised analysis of financial news. We use Twitter feeds to aggregate stock-specific news and then label these stories with their associated market response with the Yahoo Finance API. We then applied a variety of classier models to our data, such as naive bayes and kernel-based support vector machine, to classify aggregated articles by their associated market response. We achieved the most accuracy with a Bernoulli Naive Bayes model, with a peak F-1 score of 76%.

Introduction

Information dictates stock performance. As with any market driven good, accurate and clear information is the most important resource for any investor. In the case of trading stocks and commodities, the global perception of information is the most important single set of data; stock prices do not change as a result of news, but rather the perceived implication of that news. In recent years, social media sites have allowed users world wide to serve as crowd-sourced curators of news media. Articles that are shared on social media are most likely to represent the global perception of media.

While certain firms, like the U.K. based DCM Capital, have begun exploring social media analysis to inform investment decisions, the practice has not become widespread, and most investors simply rely on traditional news sources for their great authority. In our approach, we hoped to maintain the legitimacy of authoritative news sources while leveraging the pace, breadth, and crowd-sourced curation provided by social media.

Our goal is to not replace the traditional methods of analysis, but rather to complement extant methods with another technique that fully takes advantage of the true breadth and depth of information available.

Algorithm

Our high-level strategy was to pull news related to a particular stock from various online sources and track the performance of the stock from the time period corresponding to the articles release. We used the stock performance to classify the articles as indicative of positive or negative performance thereby reducing complexity by converting the tasking into a supervised machine learning problem. For testing we split our data set into training and testing portions to evaluate various classification alorithms. In particular, we employed a series of classification techniques provided by the scikit-learn Python module as well as a custom Naive Bayes implemenation. The scikit-learn techniques employed were Gaussian Naive Bayes, Linear SVM, Linear SVC, Multinomial Naive Bayes, Bernoulli Naive Bayes, Stochastic Gradient Descent, and the Gradient Boosting classifier.

Our custom implementation of Naive Bayes was motivated by the particular circumstances of the type of data we analyzed. While we reduced the problem to a supervised learning approach, in reality the problem is much more nuanced than that. An article published on a day with near-average performance cannot necessarily be classified by a marginal positive or negative performance, and indeed can be dismissed as insigicant as it did not correspond to significant market change. Similarly, many features will be present in articles regardless of sentiment or market implications.

To address both of these problems, we employed a statistical method whereby we first only considered articles from days that exhibited stock performance from outside of one standard deviation of the mean performance. We employed the same method in reducing feature noise by only considering features that were strongly linked to positive or negative articles with a weight of, again, at least one standard deviation.

This technique was based on some sweeping assumptions about the data set and the nature of naive bayes. After tweaking the classifier, we were not able to outperform a random classifier, and decided to focus on further implementation of scikit-learn algorithms.

Data

In order to successfully leverage social media for data curation and acquisition, we explored a number of strategies. The paucity of freely available historical news, particularly

financial news, poised a problem for us immediately. We experimented with a number of news sources, and found the most success with Twitter.

We used Twitter as our primary aggregator for building a database of news articles. In addition to using standard "hashtags" to link tweets together, Twitter also includes a stock tag (ex. `$GOOG`), that allows users to easily tag tweets with a particular ticker symbol. In turn, we used the stock tag feeds to pull content from all websites linked from tweets. For example, a user might tweet about a blog post or include a business article. Rather than using the user's tweet for content, we use the linked article as our primary source of content. In order to sort through the HTML, we use http://www.readability.comReadability's API to extract only relevant content and ignore unimportant features like site navigation and tangential snippets.

To retrieve financial data about certain stocks, we databased output from a Bloomberg terminal. While Yahoo and Google both provide powerful financial information services, neither one of them has an officially supported API. Indeed, it seems that there are no freely available robust APIs for financial data, so we were left with no alternative but manual input from Bloomberg.

Unfortunately, data proved to be a substantial limitation to forming any conclusive results. While we continuously collected data over the time period of the project, our data set remained quite small, and was undoubtedly a limitation to our conclusions. However, we decided to cut our losses on compiling data in order to focus on classification algorithms.

Support Vector Machines

In an effort to more appropriately test the feasibility of a social media-derived news market indicator, we sought to employ multiple classification techniques. The python module scikit-learn proved to be an excellent third party to add to our investigations. In particular, this third-party addition enabled us to experiment with support vector machine classification. Previous researchers in this field have found more success with SVMs than NB, and as such it seemed appropriate for us to test the success of SVM with our social-media derived data.

Results

Our initial custom naive bayes implementation returned 35% across multiple trials. By comparison to the Bayes classifiers in scikit-learn, the assumption of a Gaussian distribution is not appropriate for this classification task; both our NB implementation and the scikit-learn Gaussian NB implementation underperformed.

Our other techniques proved to be much more effective. In particular, Stochastic Gradient Descent proved to be a very effective classifier, as did Bernoulli Naive Bayes as shown below. A full table of our results is available in Appendix A.

Summary	Accuracy	Precision	Recall	F1 Score
Random	0.462	0.776	0.462	0.545
Gaussian NB	0.328	0.742	0.328	0.398
Linear SVM	0.277	0.694	0.277	0.341
Multinomial NB	0.168	0.765	0.168	0.122
Bernoulli NB	0.630	0.770	0.630	0.687
Stochastic Grad.	0.832	0.783	0.832	0.804
Gradient Boost.	0.328	0.781	0.328	0.387
Random Forest	0.277	0.732	0.277	0.328
Decision Tree	0.437	0.769	0.437	0.520

With up to 83% accuracy, we are quite intrigued about the capabilities of our classification technique on a larger data set. While we are employing a freely available library for classification, the integration of social media aggregation may resulted in improvements to our output.

Conclusion

While we explored several promising and cutting edge applications in our project, our research still has a way to go before it is ready to hit the trading floors. Our largest limitation remains data sources. While we have seen our classifiers improve as our database has grown, it is still too small to make any conclusive claims about our success rate.

Unlike many areas in software development, the world of financial programming doesn't include many free or open source tools to employ. While this is unsurprising, it made research more laborious than in other fields of computer science.

We intend to automate ongoing collection of data for a number of high trading volume stocks and continue running our training algorithms to observe the capability of our algorithms as our data set grows. Given the academic nature of this inquiry, we feel it would be appropriate to release the results of that ongoing classifier on a simple web interface, free of charge. It may yield some interesting results.

Given the high accuracy we were able to achieve with some of our classification techniques, results with a larger data set would certainly be interesting.

Appendix A: Detailed Results

Random	Accuracy	Precision	Recall	F1 Score
Negative	0.580	0.170	0.600	0.265
Positive	0.580	0.909	0.577	0.706
Average	0.580	0.816	0.580	0.650

Gaussian NB	Accuracy	Precision	Recall	F1 Score
Negative	0.328	0.108	0.600	0.184
Positive	0.328	0.833	0.288	0.429
Average	0.328	0.742	0.328	0.398

Linear SVM	Accuracy	Precision	Recall	F1 Score
Negative	0.277	0.092	0.533	0.157
Positive	0.277	0.781	0.240	0.368
Average	0.277	0.694	0.277	0.341

Multinomial NB	Accuracy	Precision	Recall	F1 Score
Negative	0.168	0.125	0.933	0.220
Positive	0.168	0.857	0.058	0.108
Average	0.168	0.765	0.168	0.122

Bernoulli NB	Accuracy	Precision	Recall	F1 Score
Negative	0.630	0.108	0.267	0.154
Positive	0.630	0.866	0.683	0.763
Average	0.630	0.770	0.630	0.687

Stochastic Grad.	Accuracy	Precision	Recall	F1 Score
Negative	0.832	0.143	0.067	0.091
Positive	0.832	0.875	0.942	0.907
Average	0.832	0.783	0.832	0.804

Gradient Boost.	Accuracy	Precision	Recall	F1 Score
Negative	0.328	0.126	0.733	0.216
Positive	0.328	0.875	0.269	0.412
Average	0.328	0.781	0.328	0.387

Random Forest	Accuracy	Precision	Recall	F1 Score
Negative	0.277	0.101	0.600	0.173
Positive	0.277	0.800	0.231	0.358
Average	0.277	0.712	0.277	0.335

Decision Tree	Accuracy	Precision	Recall	F1 Score
Negative	0.437	0.118	0.533	0.193
Positive	0.437	0.863	0.423	0.568
Average	0.437	0.769	0.437	0.520

References

Zhang, W. and Skiena, S. 2010. Trading Strategies To Exploit Blog and News Sentiment. In *Proceedings of the Twenty-Fourth National Conference on Artificial Intelligence,* 43-52. Menlo Park, Calif.: AAAI Press.