Stock Prediction From News Media Reactions

Hawkeye Asset Management
Parker Woodworth
Teddy Knox
Will Potter

Abstract

In today's age of the rapid transfer of information, accurately predicting the future performance of a publicly traded equity can be challenging. Given the wide range of inputs and speed at which a stocks performance can change, we decided to analyze the effect of news sources on the performance of a stock. Given how public perception is often viewed as the largest factor in determining the price of a stock, news should influence the performance of a stock immediately following its release. We use Twitter feeds to aggregate stock-specific news and then track the market response through the Yahoo Finance API. We then employ both a Nave Bayes and Scalable Vector Machine approach to classify certain articles as likely to cause a positive or negative response.

Introduction

Information dictates stock performance. As is true with any market driven good, accurate and clear information is arguably the most important resource for any investor. With the invention of the internet, information spreads in greater volumes with unprecedented velocity. Social media sites, like Twitter and Facebook, democratize the spread of information and allow anyone to share the latest popular story with the rest of the world before a reporter has arrived on scene.

Unfortunately, following the latest occurrences can be tedious due to the sheer volume of news that we are exposed to on a daily basis. Investors, averse to operating on unclear information, still rely on traditional news and media outlets as well as official releases from companies and in person interactions to make actionable decisions.

While certain firms, like the U.K. based DCM Capital, have begun exploring social media analysis to inform investment decisions, there exists no method of using social media and traditional media sources to predict the future performance of stocks. We sought out analyze a wide array of news sources relating to particular stocks to try and predict the future performance of a stock based off of historical media and its subsequent performance.

Copyright © 2013, Hawkeye Asset Management. All rights reserved.

We wanted to leverage the rapid transfer and breadth of information seen with sites like Twitter, while still using on the content written by authoritative media sources. In using both types of media, we hoped to craft the most accurate perspective possible regarding the future performance. This way, investors would not be excluded from the large unfelt benefits of social media in the equity analysis world and be able to maximize return on their portfolios. After all, if perception influences a stock's performance and media influences perception, shouldn't the media influence a stock's performance?

Our goal is to not replace the traditional methods of analysis, but rather to complement existing methods and to provide another source of information that synthesizes and classifies an a stock based on the response from similar articles.

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 immediately following the release of the article. The system would ideally train a classifier from previous data and then be able to look at realtime news and classify the performance of a stock in the immediate future.

Data

We hoped to generate the most informed opinion of a stock by using crowdsourced content (personal links, weblogs, informal sites) as well as more authoritative news (Business Insider, Bloomberg, Wall Street Journal, etc). Initially, we investigated using RSS feeds for their ubiquity and compatibility with many python libraries. Unfortunately, RSS feeds do not store historical data, so building a training set from many popular feeds was impossible.

We then resorted to scraping data from the Google Finance's aggregated web feed. Upon pull information from their feed, we discovered that the articles included only went back 6 months. Additionally, the coverage of news from the 6 month period was more sparse and infrequent than we would have liked.

We then 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.), 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 to 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. While Twitter's API only retrieves up to 5 days of Tweets, the density of coverage was much higher and we were able to scrape and store more articles than Google Finance's feed would have allowed.

To retrieve financial data about certain stocks, we used a community build Yahoo API connector. While Yahoo and Google both provide powerful financial information services, neither one of them has an officially supported API. Community members have built up libraries to help connect to services. Ultimately, we resorted to using Yahoo's YQL API with a fallback to if the YQL tables are past capacity. While Yahoo supports historical data, historical data is limited to the opening and closing prices for each day. This made it difficult to precisely track the response immediately after an article was published.

Even though we managed to connect data, we knew that the collection of articles we were retrieving from Twitter was far from ideal. While the articles we found were quality articles, we simply did not have enough time to build up a large enough database. We also could have used a more comprehensive financial data feed. Getting hourly quotes from a historical dataset would have given us more information as to the response of the company's performance. A group with more resources might have spent more money on Thomson Reuters' news feeds and market data feeds.

Naive Bayes

Classifying articles of this nature poses an interesting challenge. The articles used in training are not explicitly labeled, but since an article will have a particular sentiment with regard to stock performance, the problem can be treated as supervised. However, extracting the sentiment is a non-trivial problem. Two approaches present themselves.

The first is to extract sentiment from each individual article. Unfortunately, however, to create a suitable training set this would require human labeling; it is not possible to label individual articles against the actual stock market performance as not all articles will indicate the correct trend. Therefore, we instead used a second approach of aggregating all of the news for a given day, labeling the entire set based on the stock markets actual performance for that day and finding the most common features across all articles in the set.

Our training algorithm for naive bayes continued even further into this line of thinking and aggregated all articles from dates with outside of one standard deviate from the mean stock performance for a given date range into two sets, one linked to positive change, one linked to negative. In this way, we removed the noise caused by days that only experience very slight change on the market.

Support Vector Machines

In an effort to reduce our project's dependency on homemade code and to allow ourselves the flexibility of algorithm experimentation, we decided to incorporate a 3rd party machine learning package into our project. We decided to use Scikit-Learn, an open-source python package developed under the support of Google, to guide us through the process of building a Support Vector Machine. Their framework provides extremely helpful functions for document feature extraction as well as the mathematics and overhead of SVM model training and testing. We plan to test a variety of feature extraction methods implemented with Scikit learn, as well as implement our own. One such method is called bag-of-words, where we use the word frequencies within the corpus to construct sparse vectors representing the documents. Given that SVMs have shown to be one of the most effective approaches to textual analysis, we are confident that we will find some success with this approach if our dataset allows us.

Results

Unfortunately, our data set remains quite small, so it is difficult to draw accurate conclusions on the success of our classifiers, but our initial testing returned 40% accuracy using a naive bayes classifier and the techniques illustrated above. While this performance is below random, our data set makes any conclusions about its efficacy untenable.

To attempt better results we implemented a SVM approach that we believe will show much better accuracy. Unfortunately, the unreliability of Yahoo's Finance API servers has not allowed us to test this method. When we do test this method, we hope for at least %60 accurate article prediction.

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. Unfortunately, we never were able to fully test our hypothesis as we hit logistical issues with building this system. Regarding our news data, we had problems amassing a collection of news that corresponds to a particular company. While paid historic news APIs exist, our budget unfortunately would not let us run complete tests. Additionally, connecting to an official market data API would help us view a more detailed market response of a stock following the introduction of a piece of news.

Unlike many areas in software development, the world of financial programming doesn't include many free or open source tools to employ. While our model didn't have the most promising result statistics, the poor output most likely stemmed from an incomplete dataset rather than a broken model. From reading several papers and researching others' work, Bayesian classifiers and Scalable Vector Machines have been able to hit accuracy rates of at least 70

If we had two more months work on this project, we would spend most of our time cultivating and improving the dataset while also discussing the feature extraction component. While the machine learning piece of the equation is

easy to implement, managing a large dataset can be cumbersome and designing the best way to choose features for analysis in the articles would most likely require a good bit of strategy and testing.

Our results from this project may not have been exemplary, however, we feel that we have made promising steps into a fragmented area with poor API documentation and proprietary services for a team with a nonexistent budget.