Stock market forecasting: ML and Fundamental analysis

a mixed model of sentiment analysis, clustering and Neural networks.

Ayoub Ben Chaliah

National School for Statistics and Information Analysis
ayoub1benchaliah@gmail.com

April 27, 2019

Abstract

Many market analysts believe that predicting market's stocks fluctuations is nearly impossible to achieve due to the number of variables involved, especially since many of these variables are based on irrational factors such as human sentiment, with intricately hard to model interactions between them. However, there's an alternative known as technical analysis, which unlike the aforementioned analysis only takes in consideration the values over time of a given stock to predict its patterns, unfortunately the downside of this approach is that the more reliable you require the predictions to be, the least frequent is the event of the stock fitting into a highly selective criteria that would ensure the predictability accuracy. A large part of the reasons why forecasting based on fundamental analysis is extremely challenging is the amount of data needed to be processed to come up with accurate forecasts, rendering the process itself unattainable by human means. but thanks to today's machine learning models we can rely on computers to process different forms of data. In this paper, I'll explain how Natural language processing and sentiment analysis are used with both unsupervised and supervised machine learning models to achieve promising results in the field of market stocks forecasting.

I. Introduction

Fundamental analysis attempts to measure a security's intrinsic value by examining related economic and financial factors, which can be both qualitative and quantitative in nature. Fundamental analysts study anything that can affect the security's value, including macroeconomic and microeconomic factors. The end goal of fundamental analysis is to produce a forecast indicating whether the security is undervalued or overvalued. Thus, profiting from future security's price variations.

Experts usually distinguish between two types of forecasting; short term and long term forecasts. In order to understand this segregation, we must first understand that most of the heavily traded stocks, for instance AAPL and MSFT, derive their price values from two things. First we have the intrinsic value, this one is based on the amount of revenue being generated on the current day. The other one is the speculative value, as the name may suggest, this one is based on future predictions of the company's performance. The latter is responsible for most short term fluctuations and is heavily impacted by the overall attitude of investors toward a particular security or financial market. Therefore, the model in this paper will mainly focus on the use of financial news to forecast future price fluctuations in the short term (less than a month).

For an algorithm to be capable of establishing a link between a batch of non-numeric statements, in our case, financial news sentences, and the variation in a stock's value. It first needs to disassemble these sentences, and process each word to extract numeric

attributes out of them. These attributes are the distance (a metric that will be thoroughly explained in the following sections), and the sentiment coefficient. After extracting each word's attributes the algorithm then proceeds to forming clouds of words with similar properties. This process is a combination of embedding and clustering, its main objective is to create a dataset of non-arbitrary numerical inputs. The algorithm then relies on deep learning to extract the appropriate weights of each word and combination of words. In theory, the use of deep learning for the purpose of forecasting should be very expensive on time, computing power and input data. Moreover, It's highly improbable for the AI to reach convergence. Yet through the clustering process, the algorithm manages to hugely optimize the learning process and reduce the overall time, data and computing power required to reach AI convergence.

II. Methodology

This model uses a combination of the following machine learning concepts:

- Natural language processing
- K-Means clustering
- Feedforward neural network (Regression)

For this model to be tested several tools were either used or created from scratch. The first step was to use web mining and REST API to gather sufficient financial news and market fluctuations. We'll assume a dataset with 3 main variables; the stock's symbol, the date of publication and the headline of a certain news event.

First of all, we need to filter the stopwords, punctuations and all unnecessary parts out of the headlines, then we proceed to disassemble the headline into words. We then insert these words into a database table, giving each word a unique ID and a value representing the number of appearances of each unique word. Each headline will be represented by a sequence of IDs, generating a numerical equivalent of the headline (i.e: "word1 word2..." will be converted to [234, 153, ...], With word1 and word2 IDs being respectively 234 and 153).

Secondly, since the count of words varies from headline to headline, and since we need to create a matrix with a static size for the deep learning phase. We considered picking only 8 words from each headline. Those words are selected based on how frequent they show up in the news (the frequency is retrieved from the table of words). The idea here is to pick words with the smallest frequency. The least likely a word is to show up in a given stock's related headlines, the more likely its unusual appearance is related to unusual fluctuations of said stock's price.

One of the main concerns this algorithm has to tackle is that slight variations in a word's composition would push the neural network into treating said word as a complete different word. Another concern is that if a group of headlines with similar impact on the market used different wordings. We cannot count on the network's weights likelihood of converging towards the appropriate values because of how heavy on resources such training would be, which would render this process nearly impossible. Hence the need to create clusters of words that have similar attributes. This way the network will be reasonably sensible to the word's attributes in a way that we can easily manipulate and enhance.

Clustering is the operation of partitioning observations into homogenous subsets, or clusters, to uncover subpopulation structures in a dataset through unsupervised learning. K-means is one of several clustering algorithms and the one used here for its ability to produce tight clusters and easily explainable results. We start by converting each word into a 2-D vector that contains a distance from a reference word and a sentiment score. The sentiment score is extracted from a lexicon called "Vader lexicon" made for NLTK (a

Python NLP library). While the distance is the squared distance between "word_X" and a constant reference word based on the alphabet order of each letter. To summarize the properties of the distance calculation, it's non commutative (sensible to the order of the letters) and the difference in the score given to each word is proportional to how different their compositions are (letter combinations and length).

We use the Elbow method to get an approximate value of the optimal number of clusters needed for 50.000 english word (each word is a 2-D vector as explained in the previous paragraph). The process of computing the explained variance on such a large dataset for each possible number of clusters in order to apply the Elbow method takes a significant amount of time and is a quite costly process. Thus the need for employing the "Minibatch-kmeans" algorithm that takes random samples to estimate the explained variance. Statistically, this algorithm gives almost the same results much faster, which would greatly optimize the overall process. Consequently, we get N clusters and the coordinates of their centroids.

The final step of the preparation process is to create the deep learning input. We assign the coordinates of the closest centroid to each word. At that point, each word in a headline (words sequence) will have 3 main attributes, centroid's sentiment $CS(w)$, centroid's distance $CD(w)$ (1st & 2nd coordinates) and the word position $WP(h)$ in the sentence. Immediately we replace each word with the outcome of the following linear formula :

$$\frac{\alpha CS(w) + \beta CD(w) + \gamma WP(h)}{\alpha + \beta + \gamma} = I_j$$

Where α, β and γ are the coefficients that we can manipulate in order to highlight the most significant attributes in a word. We initially assign their values arbitrarily then vary them in order to optimize the accuracy of the predictions. Eventually, we'll get a dataframe with inputs and outputs, where the former is a matrix in which each row is an array and the latter is the variation in the stock price between the day (depending on the time interval used) before the headline's publication and the day after.

- Final step : Deep learning

Briefly, the structure of the network used for deep learning is a simple 3 layers, with a singular neuron as the output. Layers sizes starting with the input are 8x24, 24x24, 24x24 and 24x1. We use the common "sigmoid" function as an activation function.

While some would prefer a Recurrent neural network for such a problem. The reason behind choosing a basic structure, is the need to monitor and optimize the pre-learning process that create the inputs first. Additionally, with Non-feedforward structures retracing a prediction's causality could be unfeasible.

The optimization algorithm is an extension of the stochastic gradient descent known as "Adam optimization". It was used for its efficiency, low memory usage and because it's Invariant to diagonal rescale of the gradients.

III. Results

In order to evaluate the model's performance and efficiency, we need to first define what is considered to be a correct prediction. The objective of the model described in this paper is estimating the variance's magnitude of a stock price and its direction; increase or decrease. This signifies that the correct predictions are the ones that coincide with the real market's variation trends. In order to boost the model's accuracy, we can filter the outputs. For instance, we could only consider "evaluation-significant" predictions ($max(|y_i - y_{mean}|)$). We can also the apply such filters on the training data. For the

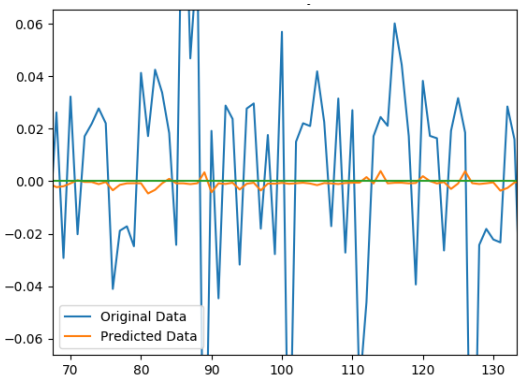
purpose of eliminating anomalies that could impact the training negatively.

Table 1: *Training and testing sets*

Unfiltered data		
Training size	Testing size	Accuracy
2000	500	54.12%
1000	357	56.02%
filtered data		
Training size	Testing size	Accuracy
2000	180	63.88%
1000	200	70.02%

• There're 2 filters applied to the data. First one is applied on the training set while the second is applied on the predictions. Both filters exist in order to remove rows that have relatively insignificant impact on the stock price from the dataset. This is done in order to avoid incertitudes that result from the neural network being unable to properly fit the most significant variations.

• The accuracy illustrated in the table above is the average accuracy ratio of several consecutive tests after each training.



Each point in the abscissa of the figure above is a particular stock and its associated ordinate is the variation of that stock price.

We can extract these informations from the last graphic:

• If both an orange dot and a blue dot were on the same side of the green line for a certain X, that prediction is correct.
• When the orange line and blue line intersect, it signifies that the model predicted almost the exact future value of the stock associated with X_i

IV. Discussion

Based on the results in the previous section, we can say that this model shows significant ability in predicting the market's variation trend based on financial news. In addition, there is a lot of space for improvements, for instance, extracting a third property from words to make 3-D clusters. Also we could improve sentiment estimation by replacing or optimizing the process. The results also show that in most cases, even when a prediction happen to have the same trend (positive/negative) as the actual market trend, there is a significant gap between the predicted variation and the market's. In theory, this can be improved by significantly increasing the size of the neural network and using a more personalised structure rather than the standard Feedforward one used in this model.