Stock Market Prediction and Portfolio Management using ML techniques

Jaspreet Kaur Thethi
Information Technology Vidyavardhini's college of Engineering
and technology University of Mumbai,
Maharashtra, India.

Aditi Pandit
Information Technology Vidyavardhini's college of Engineering
and technology University of Mumbai,
Maharashtra, India.

Hitakshi Patel
Information Technology
Vidyavardhini's college of Engineering
and technology University of Mumbai, Maharashtra, India.

Prof. Vaishali Shirsath
Assistant Professor
in the Department of Information and Technology
Vidyavardhini's College of Engineering and Technology
University of Mumbai, Maharashtra, India.

***Abstract*—Predicting stock market prices is one of the most complex task that traditionally involves extensive human-computer interaction. Due to the correlated nature of stock prices, conventional batch processing methods cannot be utilized efficiently for stock market analysis. We propose an online learning algorithm that utilizes a kind of recurrent neural network (RNN) called Long Short Term Memory (LSTM), where the weights are adjusted for individual data points using stochastic gradient descent. This will provide more accurate results when compared to existing stock price prediction algorithms. The network is trained and evaluated for accuracy with various sizes of data, and the results are tabulated. A comparison with respect to accuracy is then performed against an Artificial Neural Network.**

***Index Terms*—Stock market prediction, Sentiment analysis, Machine learning, training dataset.**

I. INTRODUCTION

The stock market is a very vast array of investors and traders who buy and sell stock, pushing the price up or down to gain profit. The prices of stocks are governed by the principles of demand and supply, and the ultimate goal of buying shares is to make money by buying stocks in companies whose recognized value (i.e., share price) is expected to rise. Stock markets are closely linked with the world of economics —the rise and fall of share prices can be traced back to some ideology - Key Performance Indicators (KPI's). The five most commonly used KPI's are the opening stock price ('Open'), end-of-day price ('Close'), intraday low price ('Low'), intra-day peak price ('High'), and total volume of stocks traded during the day ('Volume').

Economics and stock prices are mainly reliant upon subjective perceptions about the stock market. It is near impossible to predict stock prices to the T, owing to the volatility of factors that play a major role in the movement of prices. However, it is possible to make an educated estimate of prices. Stock prices never vary in isolation: the movement of one tends to have an avalanche effect on several other stocks as well [2]. This aspect of stock price movement can be used as an important tool to predict the prices of many stocks at once. Due to the sheer volume of money involved and number of transactions that take place every minute, there comes a trade-off between the accuracy and the volume of predictions made; as such, most stock prediction systems are implemented in a distributed yet parallelized fashion [7]. These are some of the considerations and challenges faced in stock market analysis and solution to this is still being found.

II. LITERATURE SURVEY

The initial focus of our literature survey was to explore generic online learning algorithms and see if they could be adapted to our use case i.e., working on real-time stock price data which included Online AUC Maximization [8], Online Transfer Learning [9], and Online Feature Selection [1].

However, as we were not able to find any potential adaptation of these for stock price prediction, we then decided to look at the existing systems [2], analyze the major drawbacks of the same, and see if we could improve them in anyway. We zeroed in on the correlation between stock data (in the form of dynamic, long-term temporal dependencies between stock prices) as the key issue that we wished to solve. A brief search of generic solutions to the above problem led us to RNN's [4] and LSTM [3]. After deciding to use an LSTM neural network to perform stock prediction, we consulted a number of papers to study the concept of gradient descent and its various types. We concluded our literature survey by looking at how gradient descent can be used to tune the weights of an LSTM network [5] and how this process can be optimized. [6].

From a negative tweet, the classifier would be more likely to classify other tweets containing the word "bad" as negative. Likewise, a bigram is a N-gram of size two and a trigram is a N-gram of size three. That means that in the case of

bigrams the feature vector for the classifier is made of a two-word combinations and in the case of trigrams is made of a three-word combinations respectively. For example, if a negative tweet contains the combination “not perfect”, in the case of the bigram feature extraction it would be classified as a negative tweet. Instead, if only unigram features were used, the tweet would have been classified as positive since the term “not” has a neutral sentiment and the word “perfect” a positive one. B. Feature Filtering With the method described above, the feature set grows larger and larger as the dataset increases leading to the point were it becomes difficult and unnecessary to use every single unigram, bigram, and trigram as a property to train our classifier. So we decided to use only the n most significant features for training. We used a chi-squared test, Pearson’s chi-squared test in particular, to score each unigram, bigram, and trigram in our training set. NLTK helped us to determine the frequency of each feature. Having, now, the features ordered by score, we selected the top-10000 to use for training and classification. This method undeniably speeded up our classifiers and reduced the amount of memory used.Traditional approaches to stock market analysis and stock price prediction include fundamental analysis, which looks at a stock’s past performance and the general credibility of the company itself, and statistical analysis, which is solely concerned with crunching of numbers and to identifying patterns in stock price variation. The latter is commonly achieved with the help of Genetic Algorithms (GA) or Artificial Neural Networks (ANN’s), but these fail to capture correlation between stock prices in the form of long-term temporal dependencies. Another major issue with using simple ANNs for stock prediction is the phenomenon of exploding / vanishing gradient[4], where the weights of a large network either become too large or too small (respectively), drastically slowing their convergence to the required optimal values. This is typically caused by two factors: weights are initialized randomly, and the weights closer to the end of the network also tend to change a lot more than those at the beginning. An alternative approach to stock market analysis is to reduce the dimensionality of the input data [2] and apply algorithms such as feature selection to shortlist a core set of features (such as GDP, oil price, inflation rate, etc.) that have the greatest impact on stock values or currency exchange rates across markets [10]. However, this method does not consider long term trading strategies as it fails to take the entire history of trends into account i.e. it fails to process all historical data; furthermore, there is no provision for outlier detection.

A. DATASET

We extracted tweets using Twitter’s Search API and processed them for further analysis, which included Natural Language Processing (NLP) and Sentiment Analysis. Thereafter, we applied neural network to predict each tweet’s sentiment. By evaluating each model for its proper sentiment classification, we discovered that neural network give higher accuracy through cross validation. In spite of this fact, we continued to take into consideration both techniques and compare every time their accuracy. After predicting every tweet’s sentiment, we extracted historical stock data using Yahoo finance API. We then created a matrix for stock market prediction using sentiment score and stock price’s change for each day and at the end we proposed our own trading strategy.

III. IMPLEMENTATION

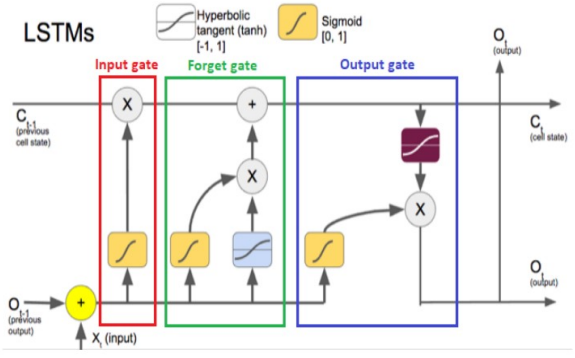
We used two live datasets: Stock Prices obtained using Yahoo! Finance API. This dataset consists of the Open, Close, High and Low values for each day. • We obtained also a collection of tweets using Twitter’s Search API. For each tweet these records provide a tweet id, the timestamp and tweet text, which is by design limited to 140 characters and needs to be filterated from noise. Since we perform our prediction and analysis on a daily basis, we saperate the tweets by days using the timestamp as the main index of the dataframe. Stock Data Preprocessing The data we collected from Yahoo! Finance had to be preprocessed in order to become suitable for further reliable analysis. The main problem we faced was that while the Twitter data were available for each day of the week, including weekends, the stock values were not present for weekends and other holidays when the market is closed. In order to fill the missing values, we overcame this problem by using a simple statistical function. If the value that is missing is y, the prior known value is xprevious and the next known value is xnext, then the value y will be: y = (xprevious + xnext)/2. This approximation works most of the time very well except in cases of unexpected rapidly rise and fall. Furthermore, we create two additional fundamental metrics: HLP CT = HighLow Low P CT change = CloseOpen Open HLPCT stands for ”High-Low Percentage” and PCTchange for ”Percentage change”. Both metrics are important for the machine learning algorithm which we applied in order to find the correlation between tweets and stock market. Twitter Data Preprocessing For the process of collecting tweets, Twitter provides two possible ways to gather Tweets: the Streaming API or the Search API. The Streaming API allows users to obtain realtime access to tweets from an input query. The user first requests a connection to a stream of tweets from the server. Then, the server opens a streaming connection and tweets are streamed accordingly. However, there are a few drawbacks of the Streaming API. First, language is not specified, resulting in a stream that contains Tweets of all languages, including a few non-Latin based alphabets, that complicates further analysis. Because of these problems, we decided to go with the Twitter Search API instead. The Search API is a REST API which allows users to request specific queries of up to date tweets. The Search API allows filtering based on language, region, geolocation and time. There is a rate limit linked with the query, but we handle it in the code. The request give a list of JSON objects that contain the tweets and their metadata. This includes a variety of information, which includes username, time, location, retweets, and more. For our needs, we mainly focus on the time and tweet text. We filter out the unnecessary metadata

and store both the tweet text and its timestamp in a .txt file. We use query the ticker of the company in front of which we add a dollar sign to gather the most "financial" tweets. Both of these APIs require an API key for authentication by the user. Once authenticated, we were able to easily access the API through a library called "Tweepy", which is a wrapper for the Twitter API.

IV. EXISTING SYSTEMS AND THEIR DRAWBACKS

Traditional approaches to stock market analysis and stock price prediction include fundamental analysis, which looks at a stock's past performance and the general credibility of the company itself, and statistical analysis, which is solely concerned with number crunching and identifying patterns in stock price variation. The latter is commonly achieved with the help of Genetic Algorithms (GA) or Artificial Neural Networks (ANN's), but these fail to capture correlation between stock prices in the form of long-term temporal dependencies. Another major issue with using simple ANNs for stock prediction is the phenomenon of exploding / vanishing gradient[4], where the weights of a large network either become too large or too small (respectively), drastically slowing their convergence to the optimal value. This is typically caused by two factors: weights are initialized randomly, and the weights closer to the end of the network also tend to change a lot more than those at the beginning. An alternative approach to stock market analysis is to reduce the dimensionality of the input data [2] and apply feature selection algorithms to shortlist a core set of features (such as GDP, oil price, inflation rate, etc.) that have the greatest impact on stock prices or currency exchange rates across markets [10]. However, this method does not consider longterm trading strategies as it fails to take the entire history of trends into account; furthermore, there is no provision for outlier detection.

V. PROPOSED SYSTEM



A. LSTM Memory cell

LSTM's are a special subset of RNN's that can capture context-specific temporal dependencies for very large periods of time. Each LSTM neuron is a memory cell that can store other information i.e., it has and maintains its own cell state. While neurons in normal RNN's merely take in their previous hidden state and the current input to output a new hidden state, an LSTM neuron also takes in its old cell state and gives outputs of its new cell state. An LSTM memory cell, as depicted in Figure 1, has the following three major components, or which are frequently referred to as gates:

1) Forget gate: the forget gate decides when specific portions of the cell state are to be updated with more recent information. It outputs values close to 1 for parts of the cell defines that should be retained, and zero for values that should be neglected and hence removed.
2) Input gate : based on the input (i.e., previous output o(t-1), input x(t), and previous cell state c(t-1)), this section of the network learns the conditions under which any information should be stored (or updated) in the cell state
3) Output gate: depending on the input and cell state, this portion decides what information is propagated forward (i.e., output o(t) and cell state c(t)) to the next node in the network.

Thus, LSTM networks are ideal for exploring how variation in one stock's price can affect the prices of several other stocks over a large period of time. They can also decide (in a dynamic fashion) for how long information about specific past trends in stock price movement needs to be retained in order to more precisely predict future trends in the variations of stock and its values. The main benefit of an LSTM is its ability to learn context specific temporal dependence. Each LSTM unit remembers and stores information for either a large or a small period of time (hence the name) without explicitly using an activation function within the recurrent components. An important fact to note is that any cell state is multiplied only by the result of the forget gate, which varies between 0 and 1. That is, the forget gate in an LSTM cell is responsible for both the weights and the activation function of the cell for current state. Therefore, information from cell at previous state can pass through a cell unchanged instead of varying i.e. increasing or decreasing exponentially at each time-step or layer, and the weights can converge to their most optimal values in a very reasonable amount of time. This allows LSTM's to solve the vanishing gradient problem – since the value stored in a memory cell isn't iteratively modified, the gradient does not vanish when trained with back propagation. Additionally, LSTM's are also relatively insensitive to gaps (i.e., time lags between input data points) compared to other RNN's.

Terminologies used below is a brief summary of the diffrent terminologies relating to our proposed stock prediction system:

1) Training set : subsection of the original data that is used to train the network model for predicting the output values
2) Test set : part of the original data that is used to make

predictions of the output value, which are then compared with the actual values to evaluate the performance of the model

3) Validation set : part of the original data that is used to tune the parameters of the neural network model
4) Activation function: in a neural network, the activation function of a node gives the output of that node as a weighted sum of inputs.
5) Batch size : number of samples that must be changed by the model before updating the weights of the parameters
6) Epoch : a complete pass through the given dataset by the training algorithm that is applied.
7) Dropout: a technique where randomly selected neurons are ignored during training i.e., they are "dropped out" randomly. Hence, their help to the activation of down-stream neurons is temporally taken out on the forward pass, and any weight updates are not applied to the neuron on the backward pass.
8) Loss function : a function, defined on a data point, prediction and label, that measures a penalty such as square loss which is mathematically explained as follows:

$$l(f(x_i),(y_i)) = (f(x_i) - (y_i))^2 \quad (1)$$

9) Cost function: a sum of loss functions over the training dataset. Mean Squared Error (MSE) is an example, which is mathematically explained as follows:

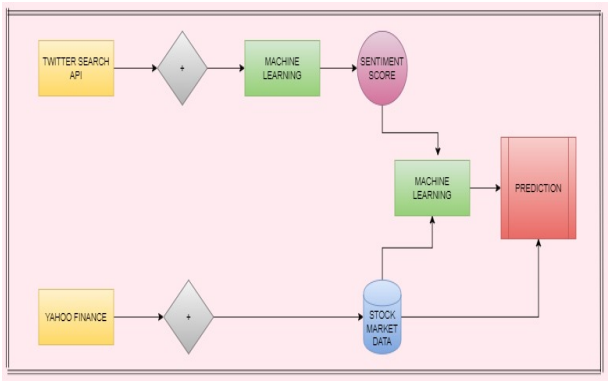
$$MSE() = N_i = l(f(x_i),(y_i)) = (f(x_i) - (y_i))^2 \quad (2)$$

10) Root Mean Square Error (RMSE): measure of the difference between values predicted by a model and the values actually observed. It is calculated by taking the summation of the squares of the differences between the predicted value and actual value, and dividing it by the number of samples. It is mathematically expressed as follows:

$$\sqrt{\frac{\sum(y_{predicted} - y_{actual})^2}{n}} \quad (3)$$

VI. SENTIMENT ANALYSIS

The text of each tweet includes a lot of words that are unrelated to its sentiment. For example, some tweets contain URLS, tags to other users, or symbols that have no interpretation. In order to better determine a tweet's sentiment score, before anything else we had not consider the "noise" that occurred because of these words. For this to happen, we are dependent on a variety of techniques using the Natural Language ToolKit (NLTK)



A. Tokenization

Firstly, we divided the text by spaces, thus forming a list of separate words per tweet. We then used each word in the tweet as features to train the classifier.

B. Removing Stopwords

Next, we removed stopwords from the list of words Natural Language ToolKit library contains a stopwords dictionary, which is a list of words that have neutral meaning and are inappropriate for sentiment analysis. To remove the stopwords from each text, we simply inspect each word in the list of words against the dictionary. If a word was in the list, we excluded it from the tweet's text. The list of stopwords contains articles, some prepositions, and other words that add no sentiment value (able, also, on, behind etc.)

C. Twitter Symbols

It is common that tweets may contain extra symbols such as "@" or "" as well as URLs. On Twitter, the word following an "@" (mentions) symbol is always a username, which we also not include because it adds no value at all to the text. Words following "" (hashtags) are not filtered out because they may contain critical information about the tweet's sentiment. They are also specifically useful for categorization since Twitter creates new databases that are collections of similar tweets, by using hashtags. URLs are filterated entirely, as they add no sentiment meaning to the text and could also be spams.

D. Training set Collection

To train a sentiment analyser and obtain data, we were in need of a system that could gather tweets. Therefore, we first collected a large amount of tweets that would serve as training data for our sentiment analyser. At the start , we considered manually tagging tweets with a "positive" or "negative" label. Thus we created a list of 1000 hand-classified tweets but because it was difficult and time-consuming, we decided to look for a database with previously sentiment-classified tweets. Surprisingly, we found that our search for other tweet corpuses returned no results, as Twitter had recently modified its terms of service to not allow public hosting of old tweets. Under these circumstances, we turned to other methods in order to form a training set. Precisely, we had two main ideas on how to classify tweets as training data. According to the first idea we created a "positive" and a "negative" dataset

for training, by using Twitter's Search API. Each dataset was created programmatically and was followed by positive and negative queries on emoticons and keywords: Any tweet that included one or more of these keywords or emoticons was most likely to be of that corresponding sentiment. This resulted in a training set of "positive" and "negative" tweets which was nearly as good as tagging tweets by hand. The second idea was that we could maybe utilize a sentiment lexicon in order to classify the gathered tweets. The one we selected was the external lexicon AFINN[4], which is a list of English words rated for valence with an integer between minus five and plus five.

E. Training the classifiers

Once we had collected a large tweet corpus as training data, we were able to build and train a classifier. Within this project we used two types of classifiers: We chose to focus on these algorithms because according to [5], they are the state of the art for Sentiment Analysis. For both classifiers, we took out the same features from the tweets to classify on. A. Feature Extraction. A unigram is a N-gram of size one. For each exclusive tokenized word in a tweet, a unigram feature is created for the classifier. In case , if a negative tweet contains the word "bad", a feature for classification would be whether or not a tweet contains the word "bad". Since the feature came from a negative tweet, the classifier would be more likely to classify other tweets containing the word "bad" as negative. Likewise, a bigram is a N-gram of size two and a trigram is a N-gram of size three. That means that in the case of bigrams the feature vector for the classifier is made of a two-word combinations and in the case of trigrams is made of a three-word combinations respectively. For example, if a negative tweet contains the combination "not perfect", in the case of the bigram feature extraction it would be classified as a negative tweet. Instead, if only unigram features were used, the tweet would have been classified as positive since the term "not" has a neutral sentiment and the word "perfect" a positive one. B. Feature Filtering With the method described above, the feature set grows larger and larger as the dataset increases leading to the point were it becomes difficult and unnecessary to use every single unigram, bigram, and trigram as a property to train our classifier. So we decided to use only the n most significant features for training. We used a chi-squared test, Pearson's chi-squared test in particular, to score each unigram, bigram, and trigram in our training set. NLTK helped us to determine the frequency of each feature. Having, now, the features ordered by score, we selected the top-10000 to use for training and classification. This method undeniably speeded up our classifiers and reduced the amount of memory used.

VII. CONCLUSION AND FUTURE WORK

The results of comparison between Long Short Term Memory (LSTM) and Artificial Neural Network (ANN) show that LSTM has a better prediction accuracy than ANN. Stock markets are very difficult to monitor and require plenty of context when trying to interpret the movement and predict prices. In ANN, each hidden node is simply a node with a single activation function, while in LSTM, each node is a memory cell that can store contextual information. Thus LSTMs perform better as they are able to keep track of the context-specific temporal dependencies between stock prices for a longer period of time while performing predictions.Also an analysis of the results also indicates that both models give better accuracy when the size of the input dataset increases. With more data, more patterns can be produced by the model, and the weights of the layers can be better adjusted. Thus at its core, the stock market is a reflection of human emotions. We also investigated whether public sentiment, as measured from tweets, is correlated or even predictive of stock values and specifically for 16 of the most popular tech companies according to Yahoo! Finance. Our results show that changes in the public sentiment can affect the stock market. That means that we can indeed predict the stock market with high chances. Furthermore, it is worth mentioning that our analysis does not take into consideration many factors. Pure numbers crunching and analysis have their own drawbacks; a possible extension of this stock prediction system would be to augment it with a news feed analysis from social media platforms such as Twitter, where emotions are gauged from the articles. This sentiment analysis can be linked with the LSTM to better train weights and further improve accuracy. There are many areas in which this work could be expanded in the future. With a longer period of time and more resources, there is much potential in the field. Finally, in the future we could create a stock lexicon based on the most common words used.

REFERENCES

[1] W. Antweiler and M. Frank. Do US stock markets typically overreact to corporate news stories? Working Paper, (1998):1–22, 2006.
[2] a. Mittal and a. Goel. Stock Prediction Using Twitter Sentiment Analysis. Tomx.Inf.Elte.Hu, (June), 2012.
[3] F. A. Nielsen. A new ANEW: evaluation of a word list for sentiment analysis in microblogs. In M. Rowe, M. Stankovic, A.-S. Dadzie, and M. Hardey, editors, Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages, volume 718 of CEUR Workshop Proceedings, pages 93–98, May 2011.
[4] A. Pak and P. Paroubek. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. Lrec, pages 1320–1326, 2010.
[5] J. Bollen and H. Mao. Twitter mood as a stock market predictor. Computer, 44(10):91–94, 2011.
[6] T. O. Sprenger, A. Tumasjan, P. G. Sandner, and I. M. Welpe. Tweets and trades: The information content of stock microblogs. European Financial Management, 20(5):926–957, 2014.
[7] P. D. Wysocki. Cheap talk on the web The determinants of postings on stock message boards. 1998.
[8] "Recurrent neural Network - Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Recurrentneuralnetwork.
[9] S. Hochreiter, "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions," International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 06, no. 02, 1998.
[10] J. S. Sepp Hochreiter, "Long Short Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.
[11] "Understanding LSTM networks," [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.
[12] "Recurrent neural network," [Online]. Available: https://en.wikipedia.org/wiki/Recurrent.neural.network.

[13] F. J. P. Tom Fawcett, ”Activity monitoring: Noticing interesting changes in behavior,” in Proceedings of 5th International Conference on Knowledge Discovery and Data Mining., 1999.
[14]] J. X. Y. e. a. Gabriel Pui Cheong Fung, ”News sensitive stock trend prediction,” in 6th Pacijic-Asia Knowledge Discovery in Data Mining, Beijing, 2002. STOCK PRICE PREDICTION USING DEEP LEARNING 52
[15] B. D. T. Yahya Eru Cakra, ”Stock Price Prediction using Linear Regression based on Sentiment Analysis”.
[16] W. Bao, J. Yue and Y. Rao, ”A deep learning framework for financial time series using stacked autoencoders and longshort term memory,” PLoS ONE, vol. 12, no. 7, 2017.
[17] P. C. Fung, X. Yu and W. Lam, ”Stock Prediction: Integrating Text Mining Approach using Real-Time News,” in Computational Intelligence for Financial Engineering, Hong Kong, 2003.