A deep learning approach to stock price forecasting

Alejandro DANIEL NOEL *
adanielnoel@gmail.com

Paolo RIZZO *
paolo.rizzo@outlook.com

Anja MEUNIER *
anjameunier@yahoo.de

Abstract

Motivated by the great performance of the latest statistical learning methods in predicting time series, we venture into predicting the closing price of stocks one day ahead, a value which is believed by many to be too random to be predicted. To make sure our models have access to different sources of information we complement the historical market values of stocks with the sentiment analysis of related daily news. A simple and efficient sentiment analysis method is proposed, which we call Word-Graph, and benchmarked against the General Inquirer method with an improvement in accuracy of 6.2%. For stock price prediction we introduce a two-step approach in which the price is predicted together with an estimate of the error of that prediction. For this we compare the performance of linear regression, KNN and recurrent neural networks with LSTM architecture to the baseline performance of not predicting any change. The performance of the neural network model is consistently above the baseline, on average by 25 %, whereas it is not the case for the two simpler methods. For error prediction, both linear regression and RNN were able to beat the baseline for some stocks. On average, the neural network model outperformed the baseline by 10%. The price change and error estimate are used to compute the risk in using that prediction, which can be taken into account when designing trading strategies.

I. Introduction

With recent developments in machine learning, stock market forecasts have grown to constitute a challenging and rewarding problem in data science. However, heated debates continue to question the legitimacy of research efforts dedicated to "beating the market" by dissecting financial time series in the hope of generating alpha. Traditional economic theory describes the financial market as an informationally efficient construct [2] [8], and states that stock prices already reflect all past and current information. This implies that stock prices should react exclusively to news, which are inherently unpredictable, and are therefore bound to follow a random walk pattern.

But the concept of efficient markets came to be long before we could teach machines to beat us in chess, poker or Go, begging the question: can the efficient market hypothesis stand the test of time or will artificial intelligence uncover unforeseen subtleties in the way financial markets absorb information? The direction we are headed in seems to be the latter. Several attempts at predicting stock prices have achieved levels of accuracy above 50% by training models to internalize financial time series, challenging the random walk interpretation of markets. Moreover, the wealth of information dumped onto us by social networks and online news sources has made ample room to leverage crowd sentiment in the struggle to beat the market. Mining the web for unstructured insight allows us to predict the outcome of elections [3], box-office revenue for movies [4], and even hypothesize what future events of interest may be [19] [5], making sentiment data a natural candidate for stock price forecasting. Most notably, [6] demonstrates that tracking public mood via Twitter can be exploited to predict changes in the closing value of the DJIA with up to 87.6% accuracy.

What seems to be lacking from literature is an approach that successfully looks at both sides of the coin: stock price time series and crowd sentiment analysis. Given that both sets of predictors have proven to be successful, their intersection is a natural domain to explore moving forward.

This paper is devoted to the development of a comprehensive deep learning pipeline for stock price forecasting. We propose a novel sentiment analysis technique, the Word-Graph, and use it to process thousands of online news articles. Sentiment data is used together with stock price data to train a system of two deep recurrent neural networks with LSTM architecture to predict closing prices at market open, as well as the associated absolute errors. The underlying hypothesis is that not only stock prices are correlated to previous prices and crowd sentiment, but their predictability (quantified by error on price predictions) is also correlated to the same quantities and thus also predictable to a certain extent. Performance is benchmarked against models with higher interpretability such as regularized linear regression and K-Nearest Neighbors. A method to use both the closing price and error prediction as a measure of risk in trading is also proposed.

We have open-sourced all the code and database for this project [1].

II. Data Acquisition and Processing

Two main types of data are collected for each stock: market data and news articles.

The market data is obtained from the free Quandl database WIKI. This database contains stock price, dividend and splits data for 3000 public US companies. Obtaining tabulated data for a certain stock is very straight forward with their API. From price and volume data we use always adjusted values. A few extra columns are calculated, such as intraday relative price change, overnight relative price change and volume relative to the 10-day volume moving average. These three columns act as difference transforms of the original values, which is considered a good method of making a time series

*We thank Prof. Pun Chi Seng for his course on Statistical Learning and Data Mining (MH4510), offered at the Nanyang Technological University in Singapore. Fall 2017.

[1] Code repository: `https://github.com/adanielnoel/NeuroTraders`

stationary [7][18]. Non-stationary time series are otherwise hard to predict, since effects of seasonality and long-term trends make the base statistics drift over time, thus difficulting the task of regressors in finding patterns. Nonetheless, in our models we have found that introducing absolute variables or even predicting them did improve the precision.

The news articles are unfortunately much more complicated to get for free. Many news agencies sell access to their archives at a premium, others have them open to the public but not directly downloadable. We used news from Reuters, which belongs to the second type of agency, and circumvented the inconvenience by programming a web scraping routine that navigates the entire archive online and downloads the articles for the stock we are interested in. This requires the user to define keywords so that the articles can be identified. Google Finance has also been included, although we found that the service does not provide a large volume of news for dates that are older than a few months. The Reuters archive on the other hand offers news back until 2007.

The articles are cleaned from symbols and other irrelevant characters and tokenized into sentences and words using the popular NLTK package. The sentiment analysis methods discussed later convert the news articles into a 3-component vector per day, which provides confidence levels for positiveness, negativeness and uncertainty.

III. Methods

A. Sentiment analysis

Among the field of natural language processing sentiment analysis is one large focus of study. There is no general analysis method and those available generally perform poorly outside the scope they are designed for. It is therefore customary for projects in algorithmic finance that include news analysis to implement sentiment analysis trained in the scope of finance.

A statistical sentiment classification model consists of a feature extractor, which generates a vector of scores for a certain piece of text, and a classifier which maps those scores to the sentiment category. In this project we define the sentiment categories by discretizing the relative intraday price change into labels *positive*, *negative* and *uncertain* using thresholds points at +1.1% and -1.1%.

The probably simplest feature extractor is the General Inquirer method. This involves comparing words in a set of news with dictionaries of positive, negative and uncertain words, keeping a score that counts the number of words in each of the three categories. This is a statistical method and therefore it requires the corpus of news to be large in order to provide accurate results. As one could expect, the simplicity of this method does not yield a very impressive accuracy, as is discussed later in the results section. The dictionaries of words have been derived from the LoughranMcDonald Master Dictionary [14].

Driven by the motivation to find a more accurate method, yet not being able to find a simple and efficient technique in literature, we endeavoured in developing our own sentiment analysis method. We call this Word-Graph, it is also a statistical method and has shown some classification performance improvement over the General Inquirer model while being as fast. In Word-Graph we construct a graph in which each node is a word found in historical news articles. During training, the edges of the graph are reinforced each time the two words they join appear together. The edge weights are 3-component vectors (s_p, s_n, s_u), with each component counting the number of appearances in each of the three contexts.

To retrieve the feature vector from a news article we simply traverse the graph while reading the article and accumulate all the weights. We filter large amount of noisy pairs by skipping those in which the maximum component of the weight is less that 1.4 times the mean of that 3-component vector (see Figure 1) and those that appear in less than 0.1% of training cases. All thresholds are found by trial and error, with the goal of maximizing the test classification accuracy for a stock not used in training. The feature vector is finally multiplied by a normalization factor that counteracts the fact that we do not have the same amount of training cases for each of the three categories, thus reducing the bias towards the most frequent. We have also tested the impact of stemming the words in training the Word-Graph, but found it to reduce performance rather than improve it.

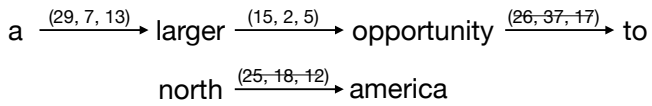

Figure 1. Sample nodes and edges from Word-Graph. The pairs *opportunity→to* and *north→america* are filtered as noise.

For both the General Inquirer and Word-Graph methods, we normalize the feature vectors by shrinking them by the ceil of standard deviations away from the statistical mean times a constant of 1.4 [2] (see Equation 1).

$$(X_c^i)_{norm} = \frac{(X_c^i - \mu)}{1.4\lceil sd \rceil} + \mu \qquad c \in \{p, n, u\} \tag{1}$$

The normalized feature vector is passed to a nearest neighbor classifier to retrieve the sentiment class. Other classifiers such as k-nearest neighbour with different k values and random forests have been tested, all resulting in lower performance.

[2] The two constants of 1.4 are unrelated. Their values being equal is coincidental.

B. Expanding Window Cross Validation

To avoid any look-ahead bias and to reflect the reality of using all available data to train the model, an expanding window approach [12] is used to train and test all models. To that end, the data sets of all individual stocks are divided into three parts, consisting of the first 60% (Set A), the next 25% (Set B) and the last 15% (Set C) of entries respectively.

In a first step the price prediction model is trained on data Set A, plus entries 1 to n from Set B and C, to predict the stock price on the next day and to calculate the squared residual. Repeatedly retraining the models to predict only the outcome of the following day, we obtain the test errors for all elements of Set B and Set C. The square root of the mean over the errors (RMSE) obtained this way is the error measure for our price prediction models.

In a second step we use the error data of the best performing price prediction model obtained during this training procedure, to train and test the error prediction model. The models are trained using Set B plus entries 1 to m from Set C, including the absolute value of the price prediction residuals as response variable, in order to predict the error of entry $m + 1$ of Set C.

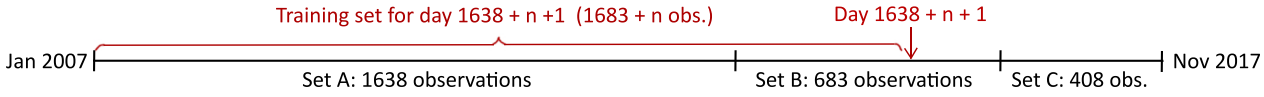

Figure 2. Illustration of Expanding Window Cross Validation

C. Linear Regression

Multi-linear regression models are a commonly used tool for the analysis of various kinds of data. A particular advantage of multi-linear regression models is their high interpretability. Although it became clear that linear regression has limited merit as a prediction model for stock price data, it was extremely useful as an indicator of significance of the various input variables in price and error modeling for each stock, and thus a useful preliminary analysis.

Since the number of observations significantly exceeds the number of predictors, we can assume a low variance and a high accuracy of the least squares predictors, which we use for linear regression models without shrinkage methods.

Price Prediction

In a first analysis, three different models were compared. The first model directly predicts the absolute value of the stock price with all available input. The other two models first predict the relative price change and calculate the absolute stock price based on it, one of them using all available predictors and the other using only the relative price changes and the sentiment scores as input. The latter model was included based on the hypothesis, that recent development and news sentiment would be the strongest indicators of near future behaviour.

Seeing that all three models failed to significantly outperform the benchmark model, a *least absolute shrinkage and selection operator* (LASSO) approach was used on the relative model with all predictors, in an effort to increase predictive power.

The optimal λ was determined using expanding window cross validation for each stock individually. Results ranged from 0.00001 for Amazon to 0.00594 for Walmart. This however only marginally improved outcomes.

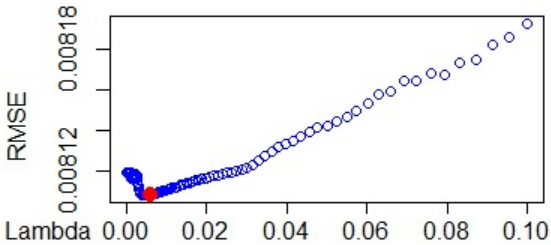

Figure 3. Finding the optimal λ for the Walmart stock with expanding window CV

Table 1. λ and variables selected by LASSO

	λ	Pos. S.	Neg. S.	Low	High	Open	Vol	Rel. Vol	Overnight	Intraday
AAPL	0.00118								x	x
GOOGL	0.00001	x	x		x	x	x	x	x	x
AMZN	0.00311	x								
VRX	0.00049						x	x	x	
WMT	0.00594	x								

As seen in Table 1, the variables selected by the LASSO are not robust across the different stocks. This makes a generalization of the model difficult and highlights, that there is no one-size-fits-all model when it comes to linear regression for stock price prediction.

Error Prediction

For error prediction, a model with six predictors proved to provide the best results, determined using a simplified backward stepwise selection. Positive and negative sentiment scores had a significant impact on the error prediction models. We hypothesize that this is due to the higher volatility of the stock after publication of relevant news, and the resulting higher potential for errors.

D. K Nearest Neighbors

In our analysis we also included a non-parametric method, K-nearest neighbors (KNN). KNN approaches are relatively simple methods, which are easy to implement. For the price prediction model, two standard unweighted KNN model were compared, one using all available predictors and the other using the five predictors which were most significant in the linear regression model. As distance measure, the Manhattan metric, which corresponds to a Minkowski distance parameter of 1, showed the best results.

Input data was scaled using the built-in standardization method of the *kknn* package. The optimal K was determined using expanding window cross validation, minimizing the RMSE for each stock individually, resulting in parameters ranging from 24 to 100.

Both models first predict the relative change in price and then calculate the absolute stock price prediction based on this information. This approach ensures that the models are able to predict stock price values outside of the range of the training set values, and thus ensuring that even a stock with a rising or falling mean can be reasonably predicted.

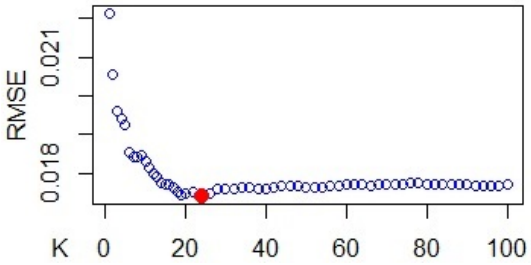

Figure 4. Finding the optimal K for the Valeant stock with expanding window CV

Table 2. Optimal K for each stock

	AAPL	GOOGL	AMZN	VRX	WMT
All Predictors	86	54	44	100	54
Selected Predictors	46	90	30	24	42

E. Recurrent Neural Network with LSTM architecture

Long Short-Term Memory (LSTM) networks have recently emerged as a popular and very effective alternative for learning problems related to sequential data. First introduced by Hochreiter & Schmidhuber [11] and refined in several other publications [1][9][10], they are designed as a solution to the problem of long-term dependencies that affects traditional recurrent neural network (RNN) architectures and prevents them from learning over large time intervals in practice. While the repeating modules of LSTMs come in several variants, the one used in this paper is the three-gated architecture, with four layers of neurons per module and no peephole connections (pioneered in [1]).

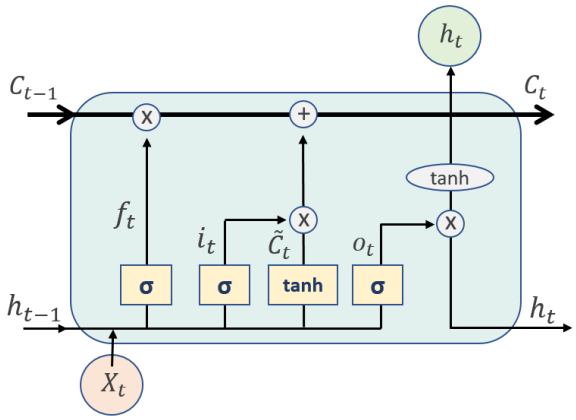

Figure 5. A diagram of a single LSTM memory unit as used in this paper, with 2 to 5 summarizing the role of the forget, input and output gates. The illustration and equations are adapted from [15].

Forget Gate:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, X_t] + b_f\right) \quad (2)$$

Input Gate:

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, X_t] + b_i\right) \quad (3)$$

$$\tilde{C}_t = \tanh\left(W_c \cdot [h_{t-1}, X_t] + b_c\right) \quad (4)$$

Output Gate:

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, X_t] + b_o\right) \quad (5)$$

As shown in Figure 5 each memory unit contains three gates. The first tanh layer constitutes the *forget gate*, which conditionally decides what information to discard from the unit. Next is the *input gate*: a sigmoid layer decides what state variables to update and a tanh layer builds a vector of possible candidate values $\tilde{C}_t$ to be added to the state. At each recurrence, we forget $f_t \cdot C_{t-1}$ and acquire $i_t \cdot \tilde{C}_t$ new information. The state at time t is thus: $C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$.

The *output gate* is made up of a sigmoid layer that decides what to output from the cell state. This output is squashed by a tanh operator and then passed through in depth (to the next layer) and in time (laterally in the same layer).

Hyperparameter Tuning

The number of stacked LSTM layers and the number of neurons in each of the component layers of a single memory unit (i.e: the size of the state vector $C(t)$) were tuned to minimize test RMSE. While stacking more layers enhances hierarchical learning capacity, and recurring a larger state vector through time allows for more features to be learned, a more complex model is significantly more likely to overfit the training data and suffer from poor generalization ability.

Remarkably, the network architecture that minimized test RMSE was the same for all stocks analyzed: 2 stacked LSTM layers with 120 neurons per component layer of a single memory unit in both LSTM layers (i.e: $C(t)$ is a 120-dimensional vector). More complex architectures led to severe overfitting of the training data, but would be interesting to explore with a larger training set at hand.

All features were normalized before being fed to the first LSTM layer, to ensure equal weighting and avoid prioritization of predictors based on their original scale. Normalization consisted in rescaling all features linearly into the range [0,1] via the mapping: $X \rightarrow \frac{X-\min}{\max-\min}$

The learning process was monitored closely to identify the learning rate and number of training epochs that minimized test loss, so that the network could be trained as much as possible without overfitting the training data. Understandably, training on different datasets led to different optimal training parameters, so these were tuned individually for each stock. Optimal learning rates were all found to be around $1 \cdot 10^{-4}$, but the associated optimal number of training epochs varied significantly, ranging from 106 for AAPL to 400 for GOOGL in the price-predicting model. Training to predict errors, instead, required considerably more epochs, ranging from 900 for AMZN to 3500 for WMT. For most stocks, test loss increased steadily after the epoch at which the global minimum was observed.

The use of dropout was also investigated with a broad range of drop probabilities starting from 10%, but ultimately resulted in under-learning, decreasing both training and test performance. This suggests that, given the large number of tunable parameters contained in the model, there are too few training points to allow for dropout to be anything but a disruption to the learning process for drop probabilities higher than 10%. Different regularization techniques were also explored, namely the implementation of L1, L2 and L1-L2 weight regularization penalties, but none appeared to increase model skill.

During training, weights are updated by means of an Adam (adaptive moment estimation) optimizer, first proposed in [13] and inspired by Adadelta [20] and Hinton's RMSProp (unpublished). Adam provides a framework to compute adaptive learning rates per parameter via bias-corrected estimates of the first and second moments of the gradients. Default values proposed by the authors are used, namely: $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$; the method has been shown to work very well in practice with the suggested settings, comparing favorably to other stochastic optimization routines.

IV. Results

A. Sentiment Analysis

The two sentiment analysis methods are benchmarked. The training data for the benchmark is composed of the news corpus for the stocks amd, amzn, intc, jpm, mmm, googl, nflx, nvda, tsla, vrx and wmt. The test data consists of the news for msft and aapl. In total there are 969 data points for training and 376 data points for testing. The results are summarised in Table 3

Table 3. Performance comparison between sentiment analysis methods

Classification accuracy	General Inquirer	Word-Graph
Positive	36.3%	42.2%
Negative	36.2%	37.8%
Uncertain	27.2%	38.1%
Overall	33.2%	39.4%

The results on the General Inquirer method show a performance that is barely above that of a 3-class random classifier. The performance obtained with the Word-Graph method is slightly better, although clearly limited by the simplicity of the feature extractor.

A paper from 2016 [17] reports accuracy of 70.2% for Word2Vec and 70.5% N-gram models, both using a random forest classifier, which is comparable to the degree of consensus in sentiment classification by humans, according to [16]. The results obtained in our research lead us to think that an accuracy beyond 40% may only be achieved with feature extractors that can extract semantic information in more generalized contexts and from larger text structures, rather than statistically comparing with a corpus. The classification method instead appears to not be of great importance.

B. Closing Price Prediction

The closing price predictions of all models are compared. Performance is benchmarked against the predictions of a purely persistent model, one that takes the opening price (closest available data point) as a prediction for the closing price of the same day. This is equivalent to constantly predicting zero intraday change, and should constitute the best possible prediction if stock prices are best described as a random walk (as implied by the Efficient Market Hypothesis).

Table 4. Test performance of all models in predicting closing price. The baseline RMSE reflects the performance of a purely persistent model that merely copies the closest datapoint.

	RMSE [USD]			
	Baseline	**Linear Regression**	**KNN**	**LSTM Neural Network**
AAPL	1.22	1.21	1.23	0.94
GOOGL	7.10	7.10	7.12	5.05
AMZN	7.50	7.55	7.60	5.79
VRX	3.02	3.07	3.03	2.40
WMT	0.59	0.59	0.59	0.43

While the regularized linear regression and KNN models generally fail to consistently outperform the baseline, test performance of the LSTM recurrent neural network beats the baseline for all stocks analyzed. The average error of the neural network model is **0.57%** of the stock price and predictions are **25%** more accurate than baseline. Moreover, performance is well balanced as the degree to which the model outperforms the baseline is similar for all stocks. This stands to confirm the ability of LSTMs to internalize complex time series significantly better than standard regression models. We conclude that a machine learning model can be trained to output a prediction at market open for today's closing price that is considerably more accurate than today's opening price, at least within the scope of the five stocks considered.

C. Absolute Error Prediction

The predictions of the absolute error on the predicted closing price are discussed. The errors that are predicted are those made by the LSTM Neural Network, which is the model with the best test performance and the one we intend to use in practice. In this case, there is no obvious choice for a baseline model to benchmark performance against. We choose to consider a baseline model that outputs a constant prediction for the error, equal to the training RMSE, and thus has no skill in identifying patterns in time that will lead to more or less accurate closing price predictions.

Table 5. Test performance of all models in predicting errors on predicted closing price. The baseline RMSE reflects the performance of a model that outputs a constant error prediction equal to the training RMSE for all time.

	RMSE [USD]		
	Baseline	**Linear Regression**	**LSTM Neural Network**
AAPL	0.67	0.63	0.63
GOOGL	4.06	3.35	3.50
AMZN	5.02	4.84	4.76
VRX	0.56	0.88	0.42
WMT	0.31	0.31	0.31

As expected, estimating the error on the predicted closing price appears to be significantly more difficult. While the neural network model never performs worse than baseline, the improvement in accuracy over the baseline is generally low and null for WMT and AAPL. The average improvement over the baseline is of **10%**. Also, the improvement in performance over the linear model is less evident, with the linear model even performing better on the GOOGL dataset. Nonetheless, the fact that the neural network consistently performs at least as well as baseline suggests that there is some predictability to the errors of the first model. While they are not captured nearly as accurately as the patterns that move the closing price, results suggest that the circumstances that make for a more accurate closing price prediction exist and can be loosely identified.

V. Using the price and error predictions to trade

This section is a proposal on how to use the results of our methods to create or improve a trading strategy. Rather than directly heeding the price change to decide an investment we can use the estimated error to evaluate the risk. If we assume the model predictions are normally distributed around the real price change, we may roughly approximate the variance of this distribution with the error that was estimated. This way, we may take the right-tailed cumulative probability as an estimate for the confidence of a positive change, as shown in Figure 6.

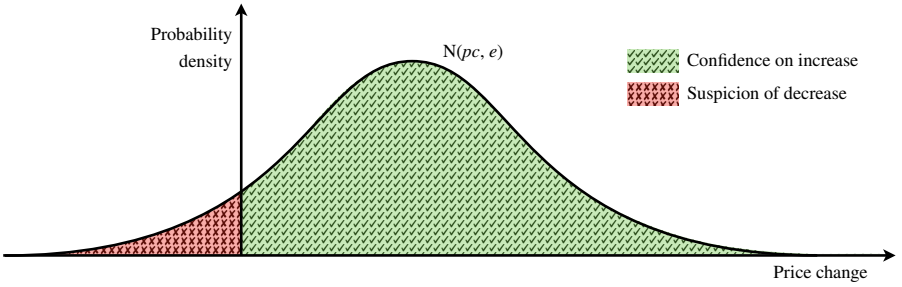

Figure 6. Estimate of investment risk based on predicted data. The symbols pc and e refer to the predicted change and the estimated error, respectively

VI. Conclusion

We develop a statistical learning pipeline for stock price forecasting that uses historical stock price data and sentiment data retrieved from thousands of online news articles. Closing prices are predicted at market open, and so is the error on said closing price prediction. We propose a method to use both outputs to measure the confidence of a price increase or decrease.

We introduce a novel sentiment analysis method, the Word-Graph, and use it to extract sentiment data from the news corpus. Despite its simplicity, Word-graph compares favorably to other sentiment analysis methods and outperforms the General Inquirer method by **6.2%**.

Out of the models considered to predict closing prices and errors, the LSTM recurrent neural network is by far the most accurate, outperforming standard regression models such as regularized linear regression and KNN. Remarkably, the neural network model is found to be **25%** more accurate than baseline in predicting closing prices. These results contradict the Efficient Market Hypothesis and show that stock prices are not a random walk, at least within the span of a day and for the stocks considered. Estimating the error on the predicted stock price is found to be a more difficult task. However, the LSTM recurrent neural network manages to consistently perform at least as accurately as baseline, outperforming it by **10%** on average. This suggests that the circumstances that allow for a more accurate closing price prediction can, albeit loosely, be determined from our predictors. In light of this result, it would be interesting to explore new predictors, such as volatility or beta, for the task of error prediction.

A recommendation for future work is to try to get access to a larger database of news, since we found ours to be rather sparse. Moreover, given the advances in sentiment analysis, it would be interesting to investigate the impact on the price prediction performance from choosing a more accurate method such as Word2Vec or N-gram.

On the other hand, company statistics, events and indicators from external events such as political, climatic and from resources and development of related fields could be incorporated into the model as features. After all, it is a matter of how much insight your model gains quicker than the market does. To this end, predictions more than just one day ahead could also be an interesting direction of study.

References

[1] Felix A.Gers, Jurgen Schmidhuber, and Fred Cummins. "Learning to Forget: Continual Prediction with LSTM". In: *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)* (1999). DOI: 10.1162/089976600300015015.

[2] E.F. Fama et al. "The adjustment of stock price to new information". In: *International Economic Review* (1969).

[3] Barfar Arash and Padmanabhan Balaji. "Predicting Presidential Election Outcomes from What People Watch". In: *Big Data* (2017). DOI: https://doi.org/10.1089/big.2017.0013.

[4] Sitaram Asur and Bernardo A. Huberman. "Predicting the Future With Social Media". In: *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology* (2010). DOI: https://arxiv.org/pdf/1003.5699.pdf.

[5] Roja Bandari, Sitaram Asur, and Bernardo Huberman. "The Pulse of News in Social Media: Forecasting Popularity". In: *Proceedings of the Seventh International Conference on Weblogs and Social Media* (2012). DOI: https://arxiv.org/abs/1202.0332.

[6] Johan Bollen1, Huina Mao, and Xiao-Jun Zeng. "Twitter mood predicts the stock market". In: *Journal of Computational Science* (2011). DOI: https://arxiv.org/pdf/1010.3003.pdf.

[7] Jason Brownlee. *How to Remove Trends and Seasonality with a Difference Transform in Python*. URL: https://machinelearningmastery.com/remove-trends-seasonality-difference-transform-python/ (visited on 11/19/2017).

[8] E.F. Fama. "Efficient Capital Markets". In: *Journal of Finance* (1991).

[9] Felix A. Gers and Jurgen Schmidhuber. "Recurrent nets that time and count". In: *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference* (2000).

[10] Alex Graves and Jurgen Schmidhuber. "Framewise phoneme classification with bidirectional LSTM and other neural network architectures". In: *Neural Networks, 18(5–6):602–610* (2005). DOI: 10.1016/j.neunet.2005.06.042.

[11] Sepp Hochreiter and Jurgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* (1997). DOI: http://www.bioinf.jku.at/publications/older/2604.pdf.

[12] Rob J Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, 2013. ISBN: 978-0987507105.

[13] Diederik Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference for Learning Representations, San Diego* (2015). DOI: arXiv:1412.6980.

[14] *LoughranMcDonald Master Dictionary*. URL: https://www3.nd.edu/~mcdonald/Word_Lists.html (visited on 11/19/2017).

[15] Christopher Olah. *Understanding LSTM Networksn*. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (visited on 11/19/2017).

[16] *On social sentiment and sentiment analysis*. URL: http://brnrd.me/social-sentiment-sentiment-analysis (visited on 11/17/2017).

[17] Venkata Sasank Pagolu et al. "Sentiment Analysis of Twitter Data for Predicting Stock Market Movements". In: *International conference on Signal Processing, Communication, Power and Embedded System* (2016). DOI: https://arxiv.org/pdf/1610.09225.pdf.

[18] M. Qi and G. P. Zhang. "Trend Time-Series Modeling and Forecasting With Neural Networks". In: (2008). DOI: 10.1109/TNN.2007.912308.

[19] Kira Radinsky and Eric Horvitz. "Mining the Web to Predict Future Events". In: *Proceedings of the sixth ACM international conference on Web search and data mining* (2013). DOI: 10.1145/2433396.2433431.

[20] Matthew D. Zeiler. "ADADELTA: An Adaptive Learning Rate Method". In: *3rd International Conference for Learning Representations, San Diego* (2012). DOI: arXiv:1212.5701.