Predicting stock prices using headlines and Google Trends: Final Report

G046 (s1837504, s1878104, s1883345)

Abstract

Financial time series prediction is a topic that has gained a lot of attention due to large potential profits. In order to approach this problem, non linear methods were applied. Among others, backpropagation networks and LSTMs are widely used for stock market prices prediction since they can capture non-linearities and both were used as baselines. Our idea is to investigate whether adding information about a particular stock such as headlines and Google Trends would improve the performance of our initial baseline. Our results indicate that in the best case scenario, the results are similar.

1. Introduction

Stock market prediction is a vastly researched, yet elusive, topic for both academia and industry. Apart from the obvious financial gains, a successful forecasting model can also provide more novel benefits such as alerting us of an upcoming financial crisis. However, the large amount of unknown factors which determine the price of stocks, e.g. news, render this problem challenging. One school of thought, technical analysis, advocates that all of the aforementioned factors are incorporated in the price and trading volume of the stocks (Cavalcante et al., 2016). Therefore, all the information required to estimate the future is stored inside past stock prices and for our prediction we can use several technical indicators. The opposite school of thought, fundamental analysis, claims that analysing the factors which cause the stock changes yields better results. In this paper we combine the above approaches by using technical indicators, the sentiment from financial headlines, and the search volume of the stock's name, in an attempt to predict Microsoft's stock data.

Our two baseline models are a simple backpropagation (BP) neural network which utilises only technical indicators as features, and a long short-term memory (LSTM) network with the same features. Since a spike in search volume may arise from either positive or negative news, in order to employ search volume as a feature, we have to calculate the sentiment regarding the company at each day. Text data is gathered from headlines of the New York Times and the sentiment values, derived from an LSTM network, are combined with the search volume to create a new feature we call **sentiment volume**. Our new feature is then fed into the baselines, along with the technical indicators, and the prediction score of our networks is re-evaluated.

Various researches have employed simple and LSTM networks in order to predict the stock market (Kara et al., 2011; Patel et al., 2015; Gers et al., 2001; Chen et al., 2015a; Qin et al., 2017; Elliot & Hsu, 2017; Nelson et al., 2017; Fischer & Krauss, 2018). In addition, several sources perform sentiment analysis in an attempt to achieve better accuracy (Zhai et al., 2007; Pagolu et al., 2016; Mittal & Goel). However, although Google Trends data, i.e. search volume, has been used for predictions in some fields such as the medical (The Flu Trends Team, 2015) and the housing market (Wu & Brynjolfsson, 2015), only a handful of papers are related to the stock market (Bank et al., 2011; Da et al., 2011; Dzielinski, 2012; Vlastakis & Markellos, 2012; Kristoufek, 2013), and only one utilises the data for developing a trading strategy (Preis et al., 2013) but without using neural networks. The main contribution of this paper is testing and quantifying the effect of including sentiment analysis and Google Trends in a hybrid machine learning approach to predicting a stock's closing price. In addition, we test the effectiveness of technical indicators, calculated from data of either 5 or 10 past days, in predicting the stock of either 1, 5, or 10 days ahead.

2. Data set and task

In this section we describe our datasets and the preprocessing we perform in each one. Our task is to identify whether combining the sentiment of financial news headlines and Google Trends decreases the mean squared error in predicting stock values. In our research we use both a BP neural network and an LSTM. The networks are tested in forecasting 1, 5, and 10 days ahead, using technical indicators, presented in table 1, from either 5 or 10 days ago.

2.1. Stocks

We obtain Microsoft's stock (MSFT) over a period of 5 years, from 31/12/2013 to 31/12/2018, which includes 1259 trading days. The daily open, high, low, and volume of the stock are collected using the AlphaVantage API (Alpha Vantage, 2017). Finally, the technical indicators of table 1 are calculated using finta (noa, 2019a), a technical analysis library for python.

2.2. Headlines

A total of 736 headlines related to Microsoft are retrieved from the New York Times using their API (The New York Times Developer Network). We define the same time window as the stock data, 31/12/2013 to 31/12/2018, and our

search query term is 'Microsoft'. In addition, their API allows filtering by tags, which the authors have provided for each article, so we also limit our request to articles which have 'Microsoft' as a tag in the organisation category.

Since one of our subtasks is to perform sentiment analysis in the aforementioned 736 headlines, it is essential to obtain training data related to our topic. In the SemEval2017 conference, participants who opted to address task 5, subtask 2, had to conduct sentiment analysis in financial news headlines obtained from online news sources such as Yahoo Finance (Cortis et al., 2017). To that end, the organisers released two files containing headlines and their Gold Standard, which was a value from -1 to 1, with lower values indicating negative sentiment. The first file contains 1142 examples, the second 14, and we merge them in order to use them later, in 3.2, as training and evaluation datapoints.

2.3. Google Trends

Google Trends data is retrieved using pytrends (noa, 2019b), for the same time window as before, 31/12/2013 to 31/12/2018, using as search term Microsoft's stock symbol, 'msft'. The value retrieved for each day is the percentage search volume for that day, compared to the day with the highest search volume in the retrieved time window, i.e. if we retrieved data for 'msft' from 01/01/2015 to 10/01/2015 and the highest search volume occurred in 06/01/2015, then the value at that day would be 100, a value of 50 at 07/01/2017 would mean that on that day we had 50% of the search volume we had on 06/01/2015. Because the API allows the retrieval of daily data for up to 90 days, after which it returns weekly data, and because the values are relative to the time window, they have to be adjusted to a common scale. To achieve that, each request to the API retrieves 90 days with the starting date of each request being the same as the last date of the previous request. That way for the one common date we have two values, one value indicating the relative search volume for the first window and one for the second window. If the value for the first window is higher, we create an adjustment factor equal to the ratio of the value in the first window over the value in the second window. We then multiply each value in the second window with that factor in order to maintain a maximum value of 100 and have a similar scale for both. If the value in the second window is higher, the process is similar but the adjustment factor is reversed and we adjust the values of the first window.

3. Methodology

To evaluate our task we build two neural networks as baselines, a BP and an LSTM, which predict the stock price using 10 technical indicators as features. After performing sentiment analysis to our relevant news headlines we add one more feature which combines the sentiment with the sentiment volume and re-evaluate the performance of our models.

3.1. Baselines

3.1.1. BP

Our first baseline is a BP network which utilises as features 10 technical indicators, shown in table 1. Technical indicators are generally accepted by fund managers and researchers (Kim, 2003; Zhai et al., 2007; Kara et al., 2011; Patel et al., 2015) as a method to predict stock prices. We opt for the indicators used by (Patel et al., 2015) who also use a BP network as their baseline.

Indicator for n days	Formula
Average	$\frac{C_t+C_{t-1}+...+C_{t-n}}{n}$
Weighted average	$\frac{(n)C_t+(n-1)C_{t-1}+...+C_n}{n+(n-1)+...+1}$
Momentum	$C_t - C_{t-n}$
Stochastic K%	$\frac{C_t-LL_n}{HH_n-LL_n} \times 100$
Stochastic D%	$\frac{\sum_{i=0}^{n-1} K_{t-i}\%}{n}$
RSI	$100 - \frac{100}{1+\sum_{i=0}^{n-1}\frac{Up_{t-i}}{n} / \sum_{i=0}^{n-1}\frac{Dw_{t-i}}{n}}$
MACD	$MACD(n)_{t-1} + \frac{2}{n+1}(DIFF_t - MACD(n)_{t-1})$
Williams %R	$\frac{HH_n-C_t}{HH_n-LL_n} \times 100$
ADL	$\frac{H_t-C_{t-1}}{H_t-L_t}$
CCI	$\frac{M_t-SM_t}{0.015D_t}$

C_t is the closing price at day t, L_t is the lowest price at day t, H_t is the highest price at day t. LL_n and HH_n are respectively the lowest low and highest high of the last n days. K_t is the stochastic K% at day t. Up_t and Dw_t are respectively the upwards and downwards closing price movement from day $t-1$ to t. $DIFF_t = EMA(12) - EMA(26)$, where EMA is the exponential moving average. $M_t = \frac{H_t+L_t+C_t}{3}$, $SM_t = \frac{\sum_{i=1}^{n} M_{t-i+1}}{n}$, and $D_t = \frac{\sum_{i=1}^{n} |M_{t-i+1} - SM_t|}{n}$

Table 1. Technical indicators looking n days in the past.

3.1.2. LSTM

LSTM networks are a special case of recurrent neural networks (RNN) which are capable of handling long term dependencies (Karpathy, 2015). The input at timestep t, in a RNN layer, is the output of timestep $t-1$ and the features in t, combined as $\mathbf{g}(t) = \mathbf{W}_{hx}\mathbf{x}(t) + \mathbf{W}_{hh}\mathbf{h}(t-1) + \mathbf{b}_h$ with $\mathbf{h}(t) = \tanh(\mathbf{g}(t))$. LSTMs also incorporate input and forget gates which allow the units to keep an internal state, not necessarily used in that cell's prediction, but capable to be passed forward in the next cells which may use it. This mechanism enables LSTMs to pass information from cells early in the chain to later cells, e.g. in word prediction where the pronoun to be predicted, she/he, must be inferred by a noun which occurred as the first word in the sentence. In our case, each timestep is one day and the input features of each timestep are the 10 technical indicators combined with the output of the previous unit. We choose LSTMs since they have been used in various instances as methods to predict price movements (Gers et al., 2001; Chen et al.,

2015a; Qin et al., 2017; Elliot & Hsu, 2017; Nelson et al., 2017; Fischer & Krauss, 2018), but also because using technical indicators as input is researched only in one paper (Nelson et al., 2017). We use a version of LSTMs called many to one, figure 1, where the output of all units, except from the last, is only used as input to the next since from one sequence of days we want to predict only one value.

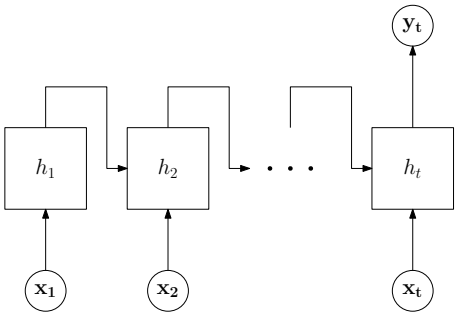

Figure 1. Many to one LSTM model

3.2. Sentiment analysis

In order to extract the sentiment from the 736 New York Times headlines we try three different approaches and pick the one which performs better in a subset of the annotated dataset from SemEval2017 (Cortis et al., 2017). The approaches tested are TextBlob (Steven Loria, 2018), the attempt from the University of Düsseldorf at SemEval2017 (Cabanski et al., 2017), and an attempt to perform sentiment analysis on Tweets from Kaggle (Nagy, 2017). The code for the last two models is available online and was used with minor adjustments.

3.2.1. TEXTBLOB

Textblob, a Python library which performs text processing, is our first and easiest approach. It is an off the shelf solution which allows instant sentiment analysis by passing the headline as an argument to a constructor and subsequently retrieving the polarity attribute, i.e. `TextBlob(headline).sentiment.polarity`. The result is a value in the range [-1, 1], with lower numbers indicating negative sentiment, similar to the convention of our annotated data from SemEval2017.

3.2.2. HHU SEMEVAL2017

The University of Düsseldorf participated at task 5, subtask 2, of SemEval2017, and placed 7th among the 29 participants in the second subtask which was about sentiment analysis of financial headlines. Since they also addressed subtask 1, twitter sentiment analysis, they used a single methodology for both. Therefore, the preprocessing task included steps unnecessary for subtask 2 such as URL and emoticon removal. They tested two different models, support-vector machines (SVM) and LSTM. We select the LSTM approach, available at GitHub (Cabanski et al., 2017), since it produced better results. Their approach generates two feature sets for each word/timestep at the LSTM. The first set generates a vector for each word in the text using Word2vec (Mikolov et al., 2013) by employing a pre-trained model (Levy & Goldberg, 2014) from spaCy (spaCy) in combination with one built from training data. The second feature set comes from several lexica with sentiment, namely SentiWordNet (Mikolov et al., 2013), VADER (Hutto & Gilbert, 2014), Opinion Lexicon (Hu & Liu, 2004), MaxDiff Twitter Sentiment Lexicon (Kiritchenko et al., 2014), and one they built from the training data.

3.2.3. KAGGLE MODEL

Our third model is one created to classify tweets sentiment from the first 2016 GOP presidential debate. The model has an embedding layer of dimension 128 which reduces the dimensionality by vector encoding the text. The embedding layer is followed by a spatial dropout 1D layer, and one hidden LSTM layer with dropout set to 0.2 in order to further prevent overfitting (Gal & Ghahramani, 2015).

3.3. Sentiment and Google Trends as features

Since we do not have headlines related to Microsoft for each day in our time window, we do not have a sentiment value for each stock data. Once we obtain the initial sentiment, we complement our sentiment values using a concave function (Mittal & Goel). For a time window with n continuous missing values, where the first available previous value is x and the next available is y, we calculate the first missing value as $\frac{x+y}{2}$ and iteratively we calculate the next $n-1$ values.

As we mention in the introduction, in order to utilise the search volume from Google Trends we need to combine it with the sentiment since a higher search volume may occur both from positive news, e.g. a product launch, and from negative news, e.g. a scandal. For example, the highest search volume for Volkswagen in the last five years occurred in the week 20/09/2015-26/09/2015, when the emissions scandal broke (Hotten, 2015) and its stock plummeted by 40%. Google Trends data is in the range [0, 100], while sentiment data is in the range [-1, 1], hence a simple way to combine them which presumably assists in stock prediction is to multiply them. Therefore, we insert their product as a new feature, **sentiment volume**, in our two baselines and re-evaluate their performance in predicting future stock prices.

4. Experiments

In this section we present our experimental set-up, our results, and their interpretation. We perform each experiment, apart from sentiment analysis, three times, with different seeds with values 0, 1, and 2. We report the average of the three runs as well as the standard error of the mean (SEM) which is defined as $\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$, where σ is the standard deviation, and n is the population size. In order to keep track

of results and hyper-parameters, we utilise Sacred (Greff et al., 2017), a framework which eases the procedure of running experiments, and stores the progress and results in a database. Our BP and LSTM networks are implemented using Keras (Chollet et al., 2015). All experiments are performed in a computer with i7-7700HQ CPU @ 2.80GHz, GeForce GTX 1060, and 8GB of RAM.

4.1. Baselines

Our stock dataset includes 1259 rows, which after calculating the 10 technical indicator (table 1) are further reduced, depending on for how many past days, n, we calculated the technical indicators. We experiment with $n = 5$ and $n = 10$ and also predict for $d = 1$, $d = 5$, and $d = 10$ days ahead. In addition, we scale our features between 0 and 1. Initial experiments indicate that when we scale the stock prices we get faster and better results so we also scale our outputs. However, the outputs of the test set are scaled back after the prediction, in order to get a better grasp of the effectiveness of our models. The final step before performing the experiments is to shift the columns with the technical indicators d rows downwards so that the stock price at date t is predicted using the technical indicators from day $t - d$. In all cases, the first 70% of the resulting dataset is used for training, the following 15% for validation, and the remaining 15% for testing. As best configuration for each model we define the one with the lowest mean squared error (MSE).

4.1.1. BP

Our baseline BP model has 10 input neurons, one for each technical indicator, the activation function of the neurons in the hidden layers is sigmoid, and the output activation function is linear. Our loss function and reported result is the mean squared error of our predictions, $\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$, where N is the sample size, y_i is the the true stock value, and $\hat{y}_i$ is our prediction. The above settings are the same as the ones used by Patel et al. (2015).

Although Patel et al. (2015) mention that they experimented with various number of epochs, from 1,000 to 10,000, they do not define which one performed better, as well as which learning rate they used. Because of this omission, and in order to save computational resources, we implement early stopping. Early stopping is a mechanism which monitors the MSE of the validation set and stops the training when a specified number of epochs has passed with no improvement in the MSE. The threshold we set is 20 epochs. Furthermore, early stopping avoids overfitting so we opt not to experiment with regularisers in both networks.

In addition, they do not mention their batch size so we experimented with sizes of 8, 16, and 32. Since the batch size is relatively independent of the other hyper-parameters (Nielsen, 2015) we find the best one before tuning anything else. The other parameters used in that experiment are: indicators with n=5, predicting one day ahead, 1 hidden layer, 100 hidden neurons, adam optimiser (Kingma & Ba, 2014), and learning rate of 0.001, the default for adam in Keras. Any parameters not mentioned above are the default by Keras. The results are displayed in table 2 and we choose batch size equal to 16 since it marginally produces the best MSE and lowest training time.

Batch size	Validation MSE (10^{-4})	Training time (s)
8	1.5966 ± 0.0155	13.67 ± 0.60
16	1.5963 ± 0.0093	12.66 ± 0.85
32	1.6097 ± 0.0059	10.48 ± 0.52

Table 2. BP baseline scores ± SEM for different batch size

In order to further reduce our search space we assume that the number of past days used for the technical indicators, and the number of days ahead we are predicting, are independent of the remaining hyper-parameters. The best configuration for $n = 5$ days and predicting one day ahead is assumed to produce the best results for and n and days ahead. We proceed by training with every combination of the values found in table 3 and keeping the number of past days at 5 and days ahead at 1. We have to mention that when we use the Adaptive Gradient (adagrad) optimiser we multiply the learning rate by 10, in order to check both optimisers for $\times 10$ and $\times 10^{-1}$ times different from their default Keras values. The best validation score of $1.5775 \times 10^{-4} \pm 0.0095 \times 10^{-4}$, achieved with training for 45.85 ± 4.42 seconds, comes from using 1 hidden layer, 150 neurons, learning rate of 0.0001, and the adam optimiser.

Hyper-parameter	Values
Hidden layers	1, 2, 3
Neurons per hidden layer	50, 100, 150
Learning rate	0.01, 0.001, 0.0001
Optimiser	adagrad, adam

Table 3. Hyper-params search space for BP baseline

4.1.2. LSTM

In order to produce a fair comparison between the BP network and the LSTM, we use the same set of input features for each timestep, i.e. the 10 technical indicators. The activation function for the hidden layers is the hyperbolic tangent, the output activation function is linear, and the loss function is MSE. The reported results are the same with the BP model, namely the validation set's MSE ± SEM. For reasons we mentioned in 4.1.1, we utilise early stopping with the threshold set to 20 epochs.

In our hyper-parameter tuning, we make the same independence assumptions as before. We consider the batch size, the number of days n in the technical indicators, and how far in the future we are predicting, independent of the remaining hyper-parameters. Although many papers employ LSTM networks to predict stock prices, only Nelson et al. (2017) use technical indicators and they also omit to mention their hyper-parameters. We therefore search for an appropriate batch size using the same settings as the

ones used in the batch size tuning of 4.1.1. We calculate the technical indicators with $n = 5$, predict the closing price of the next day, use one hidden layer, 100 neurons per hidden layer, adam optimiser, and a learning rate of 0.001. The length of the sequences we use, i.e. the number of timesteps, is set to 5 since our belief is that for short term predictions, i.e. 1-2 days, the longer in the past we look, the more irrelevant our data is. The results are displayed at table 4. Although batch size of 8 produces better results, because its SEM bars overlap with the ones of batch size 16, it is not clear whether its population mean is lower. However, it is clear that the training time required to train with batch size of 16 is lower than the one required for 8. For those reasons, and in order to speed up our experimentation, we opt for the batch size of 16 samples.

Batch size	Validation MSE (10^{-4})	Training time (s)
8	1.5588 ± 0.0380	48.52 ± 10.85
16	1.5909 ± 0.0153	29.43 ± 3.18
32	1.6216 ± 0.0636	31.99 ± 3.68

Table 4. LSTM baseline scores ± SEM for different batch size

We continue in the same way as before and test all combinations of the hyper-parameters mentioned in table 5. Once more, the learning rate is multiplied by 10 when using adagrad. The best result is obtained using 5 timesteps, 2 hidden layers, 150 neurons per layer, adam optimiser, and learning rate of 0.001. Its validation MSE is $1.5481 \times 10^{-4} \pm 0.0213 \times 10^{-4}$ and it requires 74.10 ± 5.77 seconds of training.

Hyper-parameter	Values
Hidden layers	1, 2
Neurons per hidden layer	50, 100, 150
Learning rate	0.01, 0.001, 0.0001
Optimiser	adagrad, adam
Timesteps	2, 5, 10

Table 5. Hyper-params search space for LSTM baseline

4.2. Sentiment Analysis

To evaluate the performance of the three sentiment analysis methods, we combine the two files from SemEval2017, mentioned in 2.2, inserting the file with the 14 headlines second. Our final dataset has a total of 1156 annotated headlines, the first 1000 are used as training, and the remaining 156 as test data. We do not require an evaluation set since the hyper-parameters are already tuned by their developers. However, since the Kaggle model was created for the sentiment analysis of tweets, and the language of micro-blogs differs from the one used in headlines, hyper-parameter tuning for that model may yield better results. To compare the three approaches we calculate the cosine similarity, defined in equation 1, where y is the headline's Golden Standard, and $\hat{y}$ is the predicted sentiment. The same metric was used by the organisers of SemEval2017 in order to evaluate the performance of the participants (Cortis et al., 2017).

$$cosine(y, \hat{y}) = \frac{\sum_{i=1}^{N} y_i \times \hat{y}_i}{\sqrt{\sum_{i=1}^{N} y_i^2} \times \sqrt{\sum_{i=1}^{N} \hat{y}_i^2}} \quad (1)$$

The sentiment values from TextBlob are obtained by simply passing as arguments the 156 test headlines, as discussed in 3.2.1. Both the HHU SemEval2017 model, and the Kaggle model, are trained in the training set and their test scores are derived from the test set. Since the Kaggle model was built to predict binary sentiment, we must make two adjustments in order for its output to be in the range of [-1, 1]. The first one is to use as loss function MSE, instead of categorical cross entropy. The second one is to use as output activation function the hyperbolic tangent instead of softmax. The test set results are presented in 6. It is evident that TextBlob is inferior to the other two methods which score similarly. Since the HHU model scores marginally higher, it is the one we use in the next subsection to perform sentiment analysis in our 736 headlines related to Microsoft.

Method	Test cosine similarity
TextBlob	0.1362
HHU SentEval2017	0.5317
Kaggle Model	0.5272

Table 6. Test set cosine similarity for the three sentiment analysis approaches

4.3. Sentiment and Google Trends as features

After selecting the HHU SentEval2017 model, we utilise it to extract the sentiment from our relevant New York Times headlines. Since we don't have headlines related to Microsoft for every day, we also don't have daily sentiment. As explained in subsection 3.3, we fill the empty dates using a concave function. Afterwards we obtain the sentiment volume by multiplying the sentiment, a value from -1 to 1, with the Google search volume data which takes values from 0 to 100. In order to quantify the predictive power of our new metric, we add it as our 11_{th} feature, re-tuning and re-evaluating our two baselines in an identical manner. Since the hyper-parameter tuning approach is the same, we concisely present our results and any hyper-parameter we omit is equal to the one from the respective baseline. Our final model is a hybrid, two staged neural network, with the first stage being an LSTM which performs sentiment analysis, and the second stage a BP or LSTM which utilises the new metric to predict stock market movements.

4.3.1. BP

The results of tuning the batch size are displayed in table 7. We choose batch size equal to 16 because it has the lowest training time and error slightly higher than using a batch size of 8.

Batch size	Validation MSE (10^{-4})	Training time (s)
8	1.5817 ± 0.0041	17.34 ± 2.13
16	1.5881 ± 0.0065	14.87 ± 1.49
32	1.6061 ± 0.0085	17.12 ± 1.66

Table 7. BP scores ± SEM for different batch size

Searching in the hyper-parameter space of table 3, with batch size equal to 16, we obtain the best configuration of 1 hidden layer, 150 neurons, adam optimiser, and a learning rate of 0.01. The validation MSE is $1.5777 \times 10^{-4} \pm 0.0216 \times 10^{-4}$ and it requires only 4.95 ± 0.36 seconds.

4.3.2. LSTM

Table 8 shows the batch size tests and once more we opt for size 16 because of the more consistently low training times. The best validation MSE, from the hyper-parameters search space of table 5, is $1.5159 \times 10^{-4} \pm 0.0217 \times 10^{-4}$ with 14.90 ± 2.53 seconds for training. The configuration is 2 timesteps, 1 hidden layer, 100 neurons, adam optimiser, and 0.01 learning rate.

Batch size	Validation MSE (10^{-4})	Training time (s)
8	1.5315 ± 0.0165	43.67 ± 8.30
16	1.5364 ± 0.0195	46.05 ± 2.24
32	1.5923 ± 0.0319	30.36 ± 4.26

Table 8. LSTM scores ± SEM for different batch size

4.4. Test results and discussion

In this section we compare the test set results of the best configurations for each of our four models and provide possible explanations for our observations. After we built table 9, without the columns with the double asterisk, we realised that the performance of the models with the new feature was catastrophic. This could have occurred for two reasons. Either sentiment volume impeded our networks' performance, or the best hyper-parameters we found for our new latest models had three lucky runs. Maybe even just one run with an exceptionally low MSE which dragged the average MSE down, lower than the MSE of possibly better configurations. The second one is the most likely scenario, since all the validation MSE with the added feature are in par with the baselines, something which couldn't happen if the new feature was problematic. In addition, if the problem is sentiment volume, then any configuration would produce catastrophic results. Therefore, only for the purpose of checking the hypothesis that neither the sentiment nor the search data are solely responsible for the bad performance, we go against best practices and use our test set a second time, testing with the new feature and using the best configurations from the baselines. The results are the columns with the double asterisk and indeed, the disastrous performance was not due to the added feature. A closer investigation at the best three configurations for the BP network with the new feature, reveals that the first two, which have learning rate of 0.01, have validation MSE $1.5777 \times 10^{-4} \pm 0.0216 \times 10^{-4}$ and $1.5795 \times 10^{-4} \pm 0.0213 \times 10^{-4}$. The third one, with learning rate of 0.001, has $1.5804 \times 10^{-4} \pm 0.0086 \times 10^{-4}$. Due to the SEM difference, we conclude that the higher learning rates produce more unstable results.

Disregarding the unlucky selection of hyper-parameters, our new feature is unable to assist in predicting stock values. In the best case scenario we produce similar results to our baselines. A possible reason for this is that the sentiment approach was good enough for the conference but it may still produce enough incorrect sentiments to corrupt our dataset. Another reason may be that either the news headlines, or the search volume, or both, are not accurate indicators of stock movements. Finally, it is possible that their combination through multiplication was inappropriate and another approach could produce a more meaningful feature.

With regards to the days n we use in the technical indicators, and the amount of days ahead we are predicting, it is evident from table 9 that the further in the future we are predicting, the worse our results. The most surprising fact of table 9 is the improvement in predicting 10 days ahead when we use $n = 10$ instead of $n = 5$. It seems that a 5 day window is unable to encompass the information required to predict 10 days ahead.

The true stock value and the predictions for our test set are displayed in figure 2. It is clear the the high learning rate of 0.01 for both models which utilised sentiment volume, in combination with early stopping, prevent the optimiser from further updating the weights and as a result both predictions undershoot the true price.

5. Related Work

Our results are inconsistent with the related literature. Behavioural finance (Li et al., 2014) has emphasized the critical role of behavioural and emotional factors in financial decision making. As a consequence, measuring investor and social mood has become a key research issue in financial prediction. This has caused a growing amount of research interests in exploring different methods to compute indicators of public sentiment state from large-scale, readily available textual data. Tetlock (2007) found negative words from a Harvard psychosocial dictionary, which classifies words by the moods they express, thus a bag of words related to signs of weakness of a company is generated and matched with the text to produce scores. As a result, the study found that negative words in the Wall Street Journal foreshadows market returns to have an average fall of 0.081% points trading volume the following day. Furthermore, (Tetlock et al., 2008) has narrowed down Tetlock's work to firms' level, which has reinforced the point that media content can be employed as a proxy for new insight about the fundamental asset values of companies.

The effectiveness of different textual source for sentiment

n	Days ahead	BP	BP*	BP**	LSTM	LSTM*	LSTM**
5	1	2.46 ± 0.04	9.92 ± 1.24	2.36 ± 0.01	5.48 ± 2.31	8.45 ± 4.14	7.00 ± 1.24
5	5	5.20 ± 0.05	19.64 ± 10.16	5.27 ± 0.15	9.05 ± 0.96	22.51 ± 7.15	7.50 ± 0.39
5	10	13.36 ± 0.09	12.93 ± 0.59	14.08 ± 0.58	17.60 ± 3.71	51.86 ± 16.78	74.90 ± 12.93
10	1	3.07 ± 0.02	4.55 ± 0.38	2.94 ± 0.09	8.84 ± 2.10	6.90 ± 1.56	5.63 ± 0.82
10	5	6.08 ± 0.09	9.20 ± 0.43	6.23 ± 0.03	17.99 ± 2.13	32.13 ± 14.44	6.19 ± 0.25
10	10	7.30 ± 0.23	19.00 ± 6.15	7.41 ± 0.34	7.56 ± 0.91	35.40 ± 13.98	9.56 ± 2.15

Table 9. Test set MSE ± SEM for our models.
*Model using the new feature and hyper-parameters based on the best validation MSE.
**Model using the new feature and hyper-parameters equal to the baseline.
n is the number of past days used in the technical indicators. Days ahead is how many days forward we are predicting

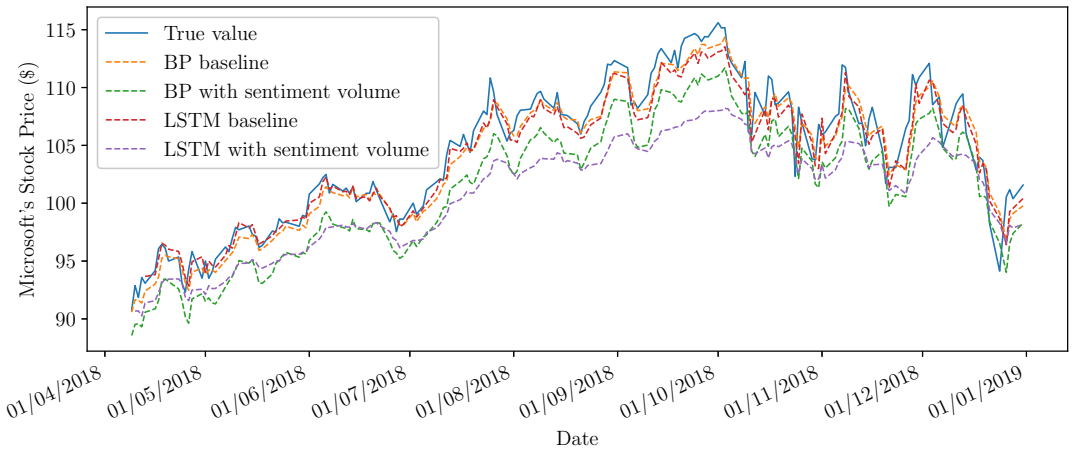

Figure 2. Test set true stock value and predictions of our models

indicators for stock market prediction and insignificant improvement has been reported based on Granger causality and compared by Mao et al. (2011). Granger causality is a technique that is widely used to analyse the relations between economic time series based on t-test. in terms of direction accuracy and MAPE of the forecasting accuracy for DJIA, VIX and volume prediction. The study blamed the extremely high volatility in the financial markets and the limitation of their simple linear model as factors for the insignificant improvements. In addition, they emphasized the research gap of a lack of aggregated textual data source to computer a particular financial index, since most existing work in sentiment analysis for stock prediction adopts only a single source, for example: Tweets, Facebook posts or new articles. The correlation between historical stock prices and market sentiment at the time has been studied by (Bharathi & Geetha, 2017). Their study used 2100 news articles and expert opinions over a 5 years from 2011 to 2016, sentimental parsing of word glossary was used to create a bag of words to match with input to produce scores. They concluded that market sentiment both drives and is driven by actual stock prices without any quantitative results being shown.

6. Conclusions

Our novel feature failed to provide compelling results, on the contrary we found that the outcome of using sentiment volume can be much worse than not using it. In addition, we identified that in order to predict d days in the future we have to calculate technical indicators with $n >= d$.

Although we calculate our deviation from the true stock price, an interesting future direction would be to implement a trading strategy, in order to get a better understanding of our networks' performance. In addition, hyper-parameter tuning in the Kaggle LSTM network was omitted, and since its score is similar to the one tuned by HHU, it may be prudent to use it for further similar research.

Finally, a mechanism that has been found to be successful in various tasks such as machine translation (Bahdanau et al., 2014), speech recognition (Chorowski et al., 2015), question answering (Hermann et al., 2015) and image captioning (Xu et al., 2015) is the attention mechanism. The idea behind this mechanism is to be able to capture and model the dependencies in the inputs or the outputs without regard to their distance in the sequence. In most cases (Parikh et al., 2016), the attention mechanism is used on top of a recurrent neural network such as an LSTM and the procedure of combining these two is described below and visualised at figure 3.

Let a vector H = $[h_1, h_2, \ldots, h_T]$ to be the output of the LSTM layer, where T is the length of the sequence. Then in order to include the attention mechanism in our model and calculate the representation r of the sequence , we have to do the following steps (Zhou et al., 2016):

1. M = $tanh(H)$
2. The numbers starts at 1 with every call to the enumerate environment.
3. $softmax(w^T * M)$, where w is a trained vector
4. The output of our combined model is calculated by passing the representation r through a sigmoid or tanh activation function and thus:
   $r = H * a^T$
   $h* = tanh(r)$ or $h = *sigmoid(r)$

The benefit of adding the attention mechanism is that each decoder output does not depend just on the last decoder state, but on a weighted combination of all the input states, something that is crucial for long sequences. The lower MSE we achieved with some initial experiments was 3.6.

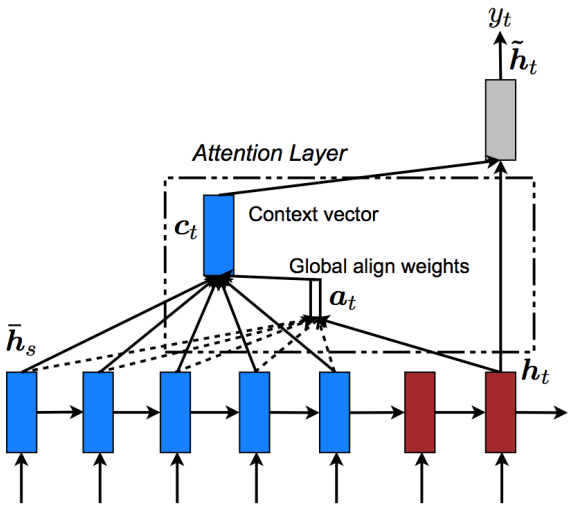

Figure 3. Attention layer architecture

References

Common financial technical indicators implemented in Pandas.: Peerchemist/finta, March 2019a.

Pseudo API for Google Trends. General Mills, February 2019b.

Akita, Ryo, Yoshihara, Akira, Matsubara, Takashi, and Uehara, Kuniaki. Deep learning for stock prediction using numerical and textual information. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pp. 1–6. IEEE, 2016.

Alpha Vantage. Alpha Vantage - Free APIs for Realtime and Historical Stock, Forex (FX), Cryptocurrency Data, Technical Analysis, Charting, and More! https://www.alphavantage.co/, 2017.

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Bank, Matthias, Larch, Martin, and Peter, Georg. Google search volume and its influence on liquidity and returns of German stocks. *Financial Markets and Portfolio Management*, 25(3):239, September 2011. ISSN 1555-4961, 1555-497X. doi: 10.1007/s11408-011-0165-y.

Bharathi, Shri and Geetha, Angelina. Sentiment Analysis for Effective Stock Market Prediction. *International Journal of Intelligent Engineering and Systems*, 10(3): 146–153, 2017.

Cabanski, Tobias, Romberg, Julia, and Conrad, Stefan. HHU at SemEval-2017 Task 5: Fine-Grained Sentiment Analysis on Financial Data using Machine Learning Methods. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pp. 832–836. Association for Computational Linguistics, 2017. doi: 10.18653/v1/S17-2141.

Cavalcante, Rodolfo C., Brasileiro, Rodrigo C., Souza, Victor L. F., Nobrega, Jarley P., and Oliveira, Adriano L. I. Computational Intelligence and Financial Markets: A Survey and Future Directions. *Expert Systems with Applications*, 55:194–211, August 2016. ISSN 0957-4174. doi: 10.1016/j.eswa.2016.02.006.

Chen, K., Zhou, Y., and Dai, F. A LSTM-based method for stock returns prediction: A case study of China stock market. In *2015 IEEE International Conference on Big Data (Big Data)*, pp. 2823–2824, October 2015a. doi: 10.1109/BigData.2015.7364089.

Chen, Kai, Zhou, Yi, and Dai, Fangyan. A LSTM-based method for stock returns prediction: A case study of China stock market. In *2015 IEEE International Conference on Big Data (Big Data)*, pp. 2823–2824. IEEE, 2015b.

Chollet, François et al. Keras. 2015.

Chorowski, Jan K, Bahdanau, Dzmitry, Serdyuk, Dmitriy, Cho, Kyunghyun, and Bengio, Yoshua. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems*, pp. 577–585, 2015.

Cortis, Keith, Freitas, André, Daudert, Tobias, Huerlimann, Manuela, Zarrouk, Manel, Handschuh, Siegfried, and Davis, Brian. SemEval-2017 Task 5: Fine-Grained Sentiment Analysis on Financial Microblogs and News. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pp. 519–535. Association for Computational Linguistics, 2017. doi: 10.18653/v1/S17-2089.

Da, Zhi, Engelberg, Joseph, and Gao, Pengjie. In Search of Attention. *The Journal of Finance*, 66(5):1461–1499, October 2011. ISSN 1540-6261. doi: 10.1111/j.1540-6261.2011.01679.x.

Dzielinski, Michal. Measuring economic uncertainty and its impact on the stock market. *Finance Research Letters*, 9(3):167–175, September 2012. ISSN 1544-6123. doi: 10.1016/j.frl.2011.10.003.

Elliot, Aaron and Hsu, Cheng Hua. Time Series Prediction : Predicting Stock Price. *arXiv:1710.05751 [stat]*, October 2017.

Fischer, Thomas and Krauss, Christopher. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, October 2018. ISSN 0377-2217. doi: 10.1016/j.ejor.2017.11.054.

Gal, Yarin and Ghahramani, Zoubin. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. *arXiv:1512.05287 [stat]*, December 2015.

Gers, Felix A., Eck, Douglas, and Schmidhuber, Jürgen. Applying LSTM to Time Series Predictable through Time-Window Approaches. In Dorffner, Georg, Bischof, Horst, and Hornik, Kurt (eds.), *Artificial Neural Networks — ICANN 2001*, Lecture Notes in Computer Science, pp. 669–676. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-44668-2.

Greff, Klaus, Klein, Aaron, Chovanec, Martin, Hutter, Frank, and Schmidhuber, Jürgen. The Sacred Infrastructure for Computational Research. *Proceedings of the 16th Python in Science Conference*, pp. 49–56, 2017. doi: 10.25080/shinma-7f4c6e7-008.

Hermann, Karl Moritz, Kocisky, Tomas, Grefenstette, Edward, Espeholt, Lasse, Kay, Will, Suleyman, Mustafa, and Blunsom, Phil. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pp. 1693–1701, 2015.

Hotten, Russell. Volkswagen: The scandal explained. December 2015.

Hu, Minqing and Liu, Bing. Mining and Summarizing Customer Reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pp. 168–177, New York, NY, USA, 2004. ACM. ISBN 978-1-58113-888-7. doi: 10.1145/1014052.1014073.

Hutto, C. J. and Gilbert, Eric. VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. In *Eighth International AAAI Conference on Weblogs and Social Media*, May 2014.

Kara, Yakup, Acar Boyacioglu, Melek, and Baykan, Ömer Kaan. Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. *Expert Systems with Applications*, 38(5):5311–5319, May 2011. ISSN 0957-4174. doi: 10.1016/j.eswa.2010.10.027.

Karpathy, Andrej. The Unreasonable Effectiveness of Recurrent Neural Networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/, May 2015.

Kim, Kyoung-jae. Financial time series forecasting using support vector machines. *Neurocomputing*, 1-2(55): 307–319, 2003. ISSN 0925-2312. doi: 10.1016/S0925-2312(03)00372-2.

Kingma, Diederik P. and Ba, Jimmy. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, December 2014.

Kiritchenko, S., Zhu, X., and Mohammad, S. M. Sentiment Analysis of Short Informal Texts. *Journal of Artificial Intelligence Research*, 50:723–762, August 2014. ISSN 1076-9757. doi: 10.1613/jair.4272.

Kristoufek, Ladislav. Can Google Trends search queries contribute to risk diversification? *Scientific Reports*, 3: 2713, September 2013. ISSN 2045-2322. doi: 10.1038/srep02713.

Levy, Omer and Goldberg, Yoav. Dependencybased word embeddings. In *In ACL*, 2014.

Li, Xiaodong, Xie, Haoran, Chen, Li, Wang, Jianping, and Deng, Xiaotie. News impact on stock price return via sentiment analysis. *Knowledge-Based Systems*, 69:14–23, 2014.

Mao, Huina, Counts, Scott, and Bollen, Johan. Predicting financial markets: Comparing survey, news, twitter and search engine data. *arXiv preprint arXiv:1112.1051*, 2011.

Mikolov, Tomas, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*, January 2013.

Mittal, Anshul and Goel, Arpit. *Stock Prediction Using Twitter Sentiment Analysis*.

Nagy, Peter. LSTM Sentiment Analysis | Keras | Kaggle. https://www.kaggle.com/ngyptr/lstm-sentiment-analysis-keras, February 2017.

Nelson, D. M. Q., Pereira, A. C. M., and de Oliveira, R. A. Stock market's price movement prediction with LSTM neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 1419–1426, May 2017. doi: 10.1109/IJCNN.2017.7966019.

Nielsen, Michael A. Neural Networks and Deep Learning. 2015.

Pagolu, V. S., Reddy, K. N., Panda, G., and Majhi, B. Sentiment analysis of Twitter data for predicting stock market movements. In *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, pp. 1345–1350, October 2016. doi: 10.1109/SCOPES.2016.7955659.

Parikh, Ankur P, Täckström, Oscar, Das, Dipanjan, and Uszkoreit, Jakob. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*, 2016.

Patel, Jigar, Shah, Sahil, Thakkar, Priyank, and Kotecha, K. Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications*, 42(4):2162–2172, March 2015. ISSN 0957-4174. doi: 10.1016/j.eswa.2014.10.031.

Preis, Tobias, Moat, Helen Susannah, and Stanley, H. Eugene. Quantifying Trading Behavior in Financial Markets Using Google Trends. *Scientific Reports*, 3:1684, April 2013. ISSN 2045-2322. doi: 10.1038/srep01684.

Qin, Yao, Song, Dongjin, Chen, Haifeng, Cheng, Wei, Jiang, Guofei, and Cottrell, Garrison. A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. *arXiv:1704.02971 [cs, stat]*, April 2017.

spaCy. spaCy · Industrial-strength Natural Language Processing in Python. https://spacy.io/.

Steven Loria. TextBlob: Simplified Text Processing — TextBlob 0.15.2 documentation. https://textblob.readthedocs.io/en/dev/, 2018.

Tetlock, Paul C. Giving content to investor sentiment: The role of media in the stock market. *The Journal of finance*, 62(3):1139–1168, 2007.

Tetlock, Paul C, Saar-Tsechansky, Maytal, and Macskassy, Sofus. More than words: Quantifying language to measure firms' fundamentals. *The Journal of Finance*, 63(3): 1437–1467, 2008.

The Flu Trends Team. The Next Chapter for Flu Trends, August 2015.

The New York Times Developer Network. Home | Dev Portal. https://developer.nytimes.com/.

Vargas, Manuel R, De Lima, Beatriz SLP, and Evsukoff, Alexandre G. Deep learning for stock market prediction from financial news articles. In *2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, pp. 60–65. IEEE, 2017.

Vlastakis, Nikolaos and Markellos, Raphael N. Information demand and stock market volatility. *Journal of Banking & Finance*, 36(6):1808–1821, June 2012. ISSN 0378-4266. doi: 10.1016/j.jbankfin.2012.02.007.

Wawre, Suchita V and Deshmukh, Sachin N. Sentiment classification using machine learning techniques. *Int. J. Sci. Res*, 5(4):1–3, 2016.

Wu, Lynn and Brynjolfsson, Erik. The Future of Prediction: How Google Searches Foreshadow Housing Prices and Sales. *NBER*, pp. 89–118, April 2015.

Xu, Kelvin, Ba, Jimmy, Kiros, Ryan, Cho, Kyunghyun, Courville, Aaron, Salakhudinov, Ruslan, Zemel, Rich, and Bengio, Yoshua. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pp. 2048–2057, 2015.

Zhai, Yuzheng, Hsu, Arthur, and Halgamuge, Saman K. Combining News and Technical Indicators in Daily Stock Price Trends Prediction. In Liu, Derong, Fei, Shumin, Hou, Zengguang, Zhang, Huaguang, and Sun, Changyin (eds.), *Advances in Neural Networks – ISNN 2007*, Lecture Notes in Computer Science, pp. 1087–1096. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-72395-0.

Zhou, Peng, Shi, Wei, Tian, Jun, Qi, Zhenyu, Li, Bingchen, Hao, Hongwei, and Xu, Bo. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pp. 207–212, 2016.