Evaluating the Performance of a Stacked Long Short-Term Memory Deep Learning Model on AEX Stock Price Prediction

Jeff Maes, Daan Cuppen, Isaac Seminck, Benjamin CD De Bosscher, Kipras Paliuis
Delft University of Technology
Faculty of Computer Science
Group 55
https://github.com/benjamindebosscher/DL-LSTM-Stock-Prediction

Abstract

Recent trends have shown the successful application of deep learning architectures on stock data. In most cases, recurrent neural network models are used for this form of time series data. In this paper, an existing long short-term memory (LSTM) model will be further developed by adding the volume as an extra input dimension. The improved model will be analysed by means of several key performance indicators and graphs. From these results, this proves to be improving the prediction accuracy of the model.

1. Introduction

In stock prediction, the goal is predicting future trends of a company stock as doing this successfully could yield a significant profit or a reduction in losses. An algorithm capable of doing this could help a trader during the decision-making process of buying, keeping or selling stocks. Stock prices have been predicted in the past using techniques such as moving averages, however, this technique has several drawbacks, as will be elaborated upon in section 3.

In this paper, modifications are made to an existing stock price prediction model based on a long short-term memory deep learning model [1] to see if its performance can be increased. Currently, this model only considers the effect of stock price changes, but research suggests that also the volume of a stock can be of interest to include in the deep model [5]. In short, the volume of a stock is how many stocks have been traded on a single trading day. A significant change in volume can reflect stock volatility, which refers to a drastic decrease or increase in stock value, causing an imbalance in trade orders. The second alteration that is made to the algorithm is to vary the prediction sequence lengths. In this way, the correlation between prediction performance and prediction length sequence. Furthermore, the LSTM model only considers previous stock data and thus no other influences on the stock price such as news events, quarterly company results, macro economic variables, etc. will be considered. In this way, we hope to achieve better results which can be used for longer periods of time as we do not want to retrain our model every trading day. The research question can be stated as follows: How can the prediction performance of an LSTM model on the stock prices of companies in the Amsterdam Exchange Index (AEX) be improved?

2. Problem Statement

Accurately predicting stock data is a difficult task. This paper aims to improve performance of an already existing deep algorithm. The data used for this project are all the underlying stocks of the Amsterdam Exchange Index. Obtaining the relevant stock data is not a problem as it is widely available. It can be downloaded but it can also be imported using an API (eg. Quandl), etc. For this project, downloading the data is preferred as this makes it possible to retrieve older stock and thus longer stock data. For every trading day the following values are obtained: open, close, low, high and volume.

The approach taken in this paper is to train an LSTM-model on all stocks within the AEX index, which takes into account both mid-price (average of open and close stock prices) and volume. The performance is checked after every alteration performed on the algorithm. Therefore, the effect of each modification can be analysed and the optimal configuration for the model can be determined. To specify in more detail which 25 stocks are analysed, a summary is given in Table 1.

Table 1. Ticker symbols of the investigated AEX stocks.

AALB	ASML	GTO	MT	REN
ABN	ASRNL	HEIA	NN	UNA
AD	ATC	INGA	PHIA	URW
AGN	DSM	KPN	RAND	VPK
AKZA	GPLG	LIGHT	RDSA	WKL

3. Method

To predict a future trend of a stock, there are several methods. One could for example opt for a well-known moving average technique. But there are also other techniques available such as a momentum-based algorithm, an LSTM model, etc. In this section, there will be elaborated on the method that is used for the stock prediction model.

3.1. Moving Average

Moving average techniques are widely applied for several prediction purposes; they give good results [1]. However, one should look further than just the optical illusion they actually might be. These techniques come with subtle but significant risks to investors [4].

First of all, they cannot predict more than one step in the future. The same prediction comes back for further predictions. This is a huge drawback for investors; instead of knowing the stock price for the next step, they would like to know the future trend (whether the stock will rise or fall) for more than one prediction step. Furthermore as their name says, moving average techniques use an average of their past values. This can be simple, exponential, weighted, cumulative, etc. Because of this average, they have difficulties to cope with high volatile stocks. Another drawback is time period. Choosing a certain time period influences the general trend a stock typifies. This means that a moving average prediction might include a certain undesired bias. Lastly, stocks show often an oscillatory pattern. This effect is hidden when using moving average methods.

All this together makes moving average techniques ideal to analyse the past behaviour of a stock. However in order to do future predictions, it lacks some effectiveness. There are better methods, such as a long short-term memory (LSTM) deep learning model.

3.2. LSTM Model

LSTM models are very powerful methods to model sequence dependent inputs. They are classified under the bigger group of Recurrent Neural Networks, RNNs in short. RNNs are able to persist information as they consist of networks with loops within them [2]. The general principle of an LSTM is visualised on the left side of Figure 1. The variable x stands for the input of the LSTM-cell A, which outputs h. The right side of Figure 1 displays the structure of the stacked LSTM model that is taken as the base model in this paper. This base model has three LSTM-cells per time step and is able to predict multiple time steps in the future. In Figure 1, the number of hidden nodes between the LSTM-cells per time step are indicated and equal 200 for the two hidden layers, and 150 for the regression layer.

As can be seen from the visualisation; the loops pass information throughout the steps. This means that this architecture has the ability to remember information. This remembering property is exactly the reason why LSTMs (RNNs in general) are much more powerful than other deep learning architectures (e.g. a convolutional neural network, CNN in short) to model time series data.

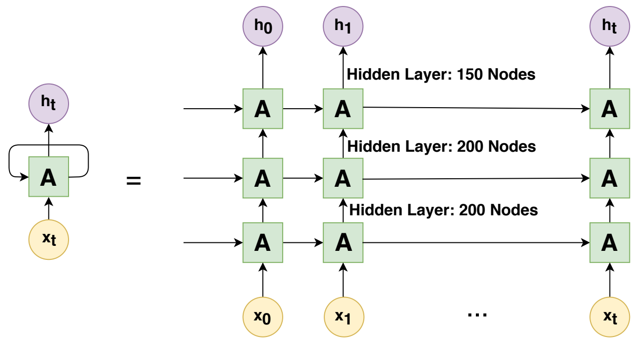

Figure 1. Visualisation of a stacked LSTM [3].

Next to remembering, LSTM models also have the ability to forget information with the help of a sigmoid layer. This property made them eliminate the long-term dependencies problem conventional RNNs traditionally struggle with [3]. The problem with these long-term dependencies is that the gradients might either vanish or explode; a consequence which should be avoided. The general outline of an LSTM network is visualised in Figure 2.

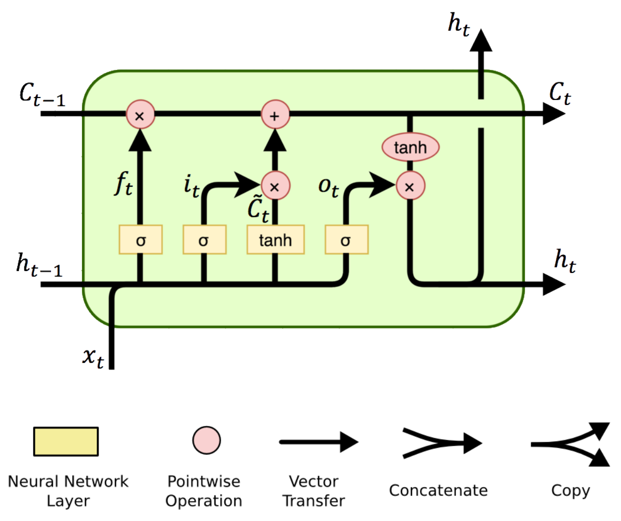

Figure 2. Visualisation of an LSTM cell [3].

Readily, five fundamental components can be identified [1]. First, there is the cell state c_t which represents the memory of the cell. Secondly, an LSTM consist of a hidden state h_t. Essentially, the hidden state contains the output which is merely a filtered version of the cell state. Furthermore, the input gate i_i can be identified. It decides how much of the input x_t flows to the cell state c_t. Fourthly, there is a forget gate f_t which determines the amount of information that flows from the previous hidden state h_{t-1} and the input x_t to the current cell state c_t. Lastly, the output gate o_t decides the amount of information that flows from the cell state c_t to the hidden state h_t.

To summarise, an LSTM model is a powerful tool in modelling time series, The basis LSTM model in this paper has three LSTM-cells per time step, is able to predict multiple steps into the future and is a SISO-system.

3.3. Data Preprocessing

For illustrative purposes, only one stock (PHIA) will be considered. For this stock, the total number of data inputs is slightly over 6000 trading days. This input data was first split into a training set of 5000 data points and a test set of over 1000 data points. Then both the test and training data sets were normalised with respect to the training set, because the algorithm should have no access to the test data in advance. Due to the fact that a particular stock increases substantially over the years, different time periods have different value ranges. Therefore the data is normalised by splitting the full series of data into four normalising windows. This introduces a break at the end of every window, thus four data points will be lost due to this approach of normalisation. For the PHIA stock this normalising window was chosen to be 1000 datapoints. Subsequently, the training data was smoothed out by using the exponential moving average in order to erase the sharpness of the prices. This helps the LSTM to train effectively and obtain improved performance as the LSTM is trained only to predict the general trend of the stock price.

3.4. Data Augmentation

Normally the prediction output $\hat{y}_i$ is based upon one input x_i. In order to augment the data, the input is sampled from the full input sequence $[x_{t+1}, x_{t+2}, \cdots, x_{t+N}]$ where N is the number of data points, resulting in input and output batches. Here, the assumption is made that the price between two subsequent input data points does not change much. This is a reasonable assumption for stock prices. In Figure 3, this data augmentation is explained visually. In a batch each entry is taken to be the first data point after a cursor. Thus the first sampled input batch is $[x_0, x_{N/b}, \cdots, x_{N-b}]$. Each cursors starts at the beginning of a new segment in the input sequence, which is N/b data points long. For the second batch the cursors are shifted to the right by one array element, resulting in $[x_1, x_{N/b+1}, \cdots, x_{N-b+1}]$. This means that the data augmentation results in b batches, each N/b long. The batch size is chosen to be 500 with a input training data sequence of 5000.

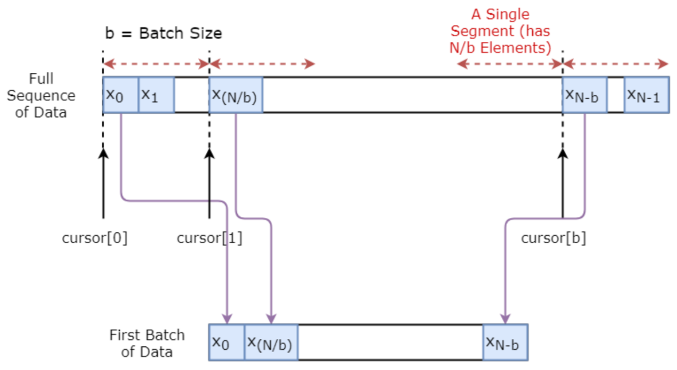

Figure 3. Visualisation of data augmentation [1].

3.5. Adding Volume

The original LSTM model [1] is a stacked SISO (single input single output) LSTM model with three cells per time step and is able to predict multiple time steps in the future. The original model only takes the mid-price as input and output. The volume is added as an additional input to this model to see if stock price prediction performance increases and the input thus becomes two-dimensional. In the prediction phase, the input data for the model is the output of the previous time step. Therefore, the model also needs to output volume when predicting multiple time steps into the future. If the model would only predict one time step into the future, the volume output would be unnecessary.

3.6. Varying Prediction Length

Another way of improving the current LSTM model is by looking into the effect of varying the length of time for which predictions are made. Ideally, the model predicts for a longer time window, making it usable for longer periods without the need for retraining and allowing for more long-term predictions. However, using a longer prediction window might negatively impact the prediction accuracy. It is thus worth looking into the effects on the performance by varying the prediction window length and determining the optimal window length, finding the optimal accuracy and prediction length ratio. The model will thus be trained for different prediction length windows and its performance evaluated.

3.7. Performance Indicators

In order to evaluate the performance of the LSTM more accurately, the following performance indicators are used. The mean absolute error (MAE) measures the average magnitude of the errors in a set of predictions, without taking their direction into consideration. The MAE is is defined as follows:

$$\text{MAE} = \frac{1}{n}\sum_{j=1}^{n} |y_j - \hat{y}_j| \tag{1}$$

Where n is the number of prediction steps (number of unrollings of the LSTM), $\hat{y_j}$ are the pricepredictions and y_j are the mid-prices. The root mean squared error (RMSE) is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n} (y_j - \hat{y}_j)^2} \tag{2}$$

The RMSE gives a higher weight to large errors compared to the MAE (same weight to all errors) as the errors are squared before averaged. The linear correlation coefficient (LCC) measures the strength of the linear relationship between two variables. The LCC for a sample is defined as:

$$r_{xy} = \frac{\sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n} (y_i - \overline{y})^2}} \tag{3}$$

Where $\overline{x}$ and $\overline{y}$ are the means of the predictions and the mid-prices over the number of prediction step. Essentially, this equation is equal to: $\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y}$. Lastly the maximum absolute error (MAE) measures the maximum error and indicates how the predictions follow extreme trends in the stock price.

3.8. Hyperparameter Tuning

It was also worth to look into how sensitive the model is to the choice of the parameters and whether they could be optimally tuned to maximise the model's predictive capabilities. A grid search will be performed, training the model several for a variation of the following parameters:

- Batch size
- Number of nodes of each layer
- Number of layers
- Dropout rate

While the results of the model will be evaluated using the up/down correctness factor.

3.9. Random Initialisation

Furthermore, the effect of randomness on the results should be examined. Machine learning algorithms and in this case, deep learning algorithms, rely on the consequence of randomness. For the LSTM, the weights are randomly initialised. To check the consequence of this, the algorithm will be run 50 times (with each 10 epochs) and the prediction accuracy parameter will be compared.

3.10. Training on All Stocks

In order to improve the performance of the algorithm further, the LSTM model was trained on all stocks of the AEX. This means that the training set consists of all the first 80% of each stock. Thus the test set consist of the last 20% of every stock placed after each other.

In this paper the ADAM optimiser used was used for training as this optimiser combines momentum based optimiser and RMSprop.

4. Results

Having already achieved adding the stock volume next to the stock price as an input variables for the network to train on, now the results of these extra inputs are compared with tables 2 and 3. It can be easily seen that the performance indicators do not change significantly after the fifth epoch. Comparing tables 2 and 3, the LSTM excluding volume performs marginally better across all performance indicators.

Table 2. Results for different performance indicators for LSTM excluding volume.

	MAE	MRE	RMSE	LC
epoch 1	0.5849	0.5728	0.4234	0.5476
epoch 5	0.1098	0.1135	0.0854	2.4416
epoch 10	0.1140	0.1199	0.08912	2.2471
epoch 15	0.1125	0.1185	0.0879	4.887
epoch 20	0.1107	0.1174	0.0863	2.8042
epoch 25	0.1101	0.1171	0.0858	2.3500
epoch 30	0.1113	0.1178	0.0868	3.2977

Table 3. Results for different performance indicators for LSTM including volume.

	MAE	MRE	RMSE	LC
epoch 1	0.8304	0.7937	0.5952	0.5857
epoch 5	0.1058	0.1096	0.0824	1.2686
epoch 10	0.1415	0.1319	0.1102	0.6968
epoch 15	0.1230	0.1185	0.0959	1.3257
epoch 20	0.1221	0.1180	0.0952	1.2834
epoch 25	0.1228	0.1186	0.0958	1.2172
epoch 30	0.1234	0.1189	0.0963	1.1571

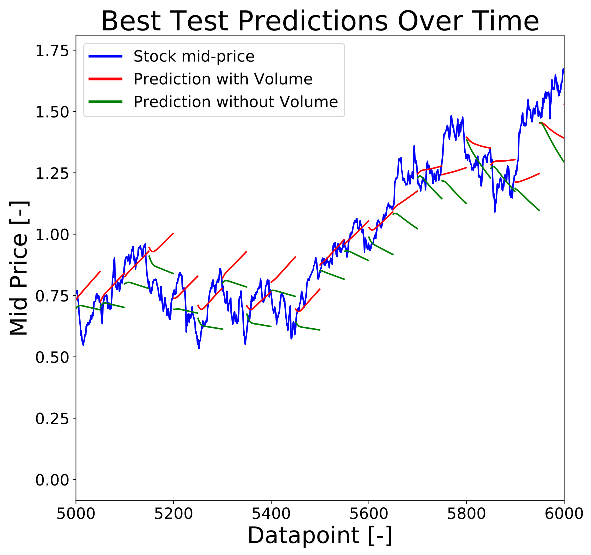

Figure 4. Prediction of mid-price for the model with and without volume for the Philips (PHIA) stock.

In Figure 4, the prediction of the normalised mid-price is plotted for the 1,000 data points considered in the testing phase. Predictions are plotted for both models, thus with and without taking volume into account. It can be seen that visually, the predictions with volume perform better than the model without volume. However, the results from Table 3 indicate that the opposite is true.

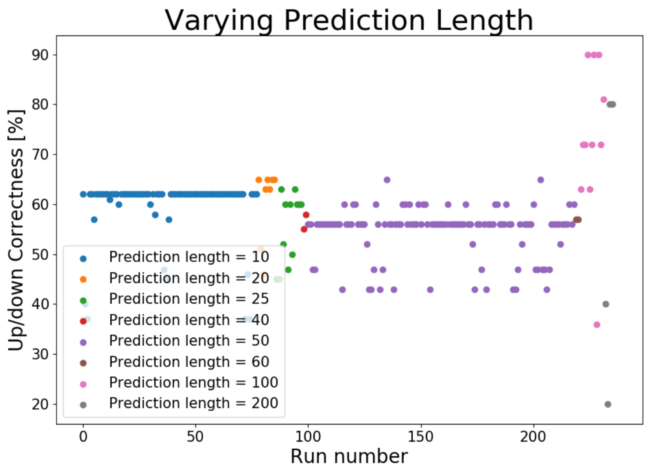

Figure 5. Up/down correctness factors for varied prediction lengths.

The up/down correctness factor for different prediction length windows can be seen in Figure 5. A drop in performance can be seen when comparing the predictions for 10 days, the average correctness being 59%, whereas at 50 days, the average correctness goes down to 54.5%, which does indicate that it is more difficult to predict the stock movement for longer periods of time.

A significant increase can be seen when predictions are made for 100 days in advance, with the average correctness reaching 72.8%. However, this is does not seem like a reliable result, as the test dataset is only around 1000 data points, meaning that only 10 predictions are made. It is thus not extremely statistically unlikely that a high number of them turn out to be correct. This is further backed up by the visibly increased variance in the data, as the prediction length increases, as the correctness factor becomes much more spread out.

The optimal results were derived with the following hyper-parameters:

- Batch size of 250
- 3 layers with 200, 200 and 150 nodes
- Dropout of 0

Which was the default setting for the number of layers and nodes. However, the most accurate results came from a batch size which was half of the default 250 instead of 500, with no dropout being used.

As mentioned earlier, the effect of randomness is examined. In Table 4, the frequency of the results for the up/down prediction accuracy parameter are given for the 50 runs for a prediction up to 25 days.

Table 4. Results for the randomness effect test.

Up/down correctness [%]	Frequency [%]
47	2
50	2
52	2
54	6
56	26
58	12
60	26
63	20
65	4
Total	100

In the left column of Table 4, the different results of the prediction accuracy parameter are listed. At the right side, the frequency of a certain value of the prediction accuracy is given. Logically, the right column adds up to 100%. For the 50 runs, the mean of the up/down correctness parameter is equal to 58.54% and the median is equal to 59%. The standard deviation is equal to 3.76%. From the results, one can see that the randomness can introduce quite a big difference in the outcome of the KPIs. However, looking at the frequency of the different outcomes and the rather small standard deviation, running the algorithm multiple times and averaging the results solves the effect of this randomness factor.

Lastly, the affect of training on the first 80% and training on the last 20% of all stocks was investigated. This yields no significant increase in performance with a correctness rate of 59% and a RMSE of 0.9394. Normally one should expect the performance to increase when using more data, however this is not the case. This is due to the fact that the training set is normalized, but the test set is not. Thus there exist a great difference in value ranges as the different stock prices differ a lot from each other.

5. Discussion

Reflecting on the obtained results, in section 4, it can be concluded that adding the volume data to the LSTM model improves prediction accuracy with maximum 9%, with tuned hyperparameters.

When the prediction length was increased, the model became significantly less reliable, with the standard deviation increasing from 6% when predicting for 10 days to 15% for 100 days. It is thus optimal to keep the prediction length minimal and 10 days would be the recommended setting.

A limitation on this approach might be the fact that the performance of the model is evaluated on the correctness factor, whereas it is trained using the mean squared error as a loss function. The correctness factor and the mean squared error are not necessarily negatively correlated, as was seen when comparing Figure 4 and Table 3 and Table 2, where the correctness increased, while RMSE also increased after implementing volume. This is further backed up by the fact that about a third of all runs had the highest correctness factor at the first epoch, when the weights were randomly selected. Indeed, an implementation of a better fitting loss function could provide better results and is worth looking into.

Looking into the technical analysis of stock data was a valuable suggestion given by the teaching assistants during the poster presentations of the course. After having a closer look at technical analysis, it became apparent that this does not fit within the goal of deep learning. Since the 1940's, technical analysis has been the antipole of fundamental analysis. Technical analysis makes it possible to detect trends and patterns in past stock price fluctuations, which can be linked to certain future events of a stock's price. Several indicators have been developed and include the Relevant Strength Index (RSI), Moving Average Convergence Divergence (MACD), Bollinger bands, etc. The problem with these indicators is that they are manually developed features. However, this is paradoxical with deep learning as deep learning does not require the input of man-made features, what distinguishes it from machine learning, which does use features.

Lastly, some variants of the standard LSTM model could be investigated in order to improve performance. One of these variants is am LTSM with peepholes, where gate layers can 'look' at the cell state to obtain more information about the state. Another variant is Gated Recurrent Unit (GRU) model, where the forget and input gates are combined into a single update gate. This model also merges the cell state and the hidden state. More information can be found in [3].

6. Conclusion

The goal of this paper was to evaluate the performance of a stacked Long Short-Term Memory (LSTM) deep learning model on stock price prediction. For prediction, all underlying stocks of the Amsterdam Exchange Index were considered. The problem of stock price prediction using an LSTM has been done before in [1], which was used as a baseline model for this paper. This baseline model is in this paper improved in several ways in order to obtain better prediction results.

First, as the baseline only considered the input of stock prices, additional effort has been done in adding the volume data which increases prediction performance with maximum **9%**, with tuned hyper parameters. Second improvement is the addition of hyperparameter tuning, which was not yet performed in [1], but was necessary to minimise the predefined loss function and maximise performance. Running the model with tuned hyperparameters and the addition of the volume data leads to an increased prediction accuracy of **65%** compared to the model without volume data and no hyperparameter tuning.

Several extra improvements are made, such as making it possible to save the model once it is trained, avoiding the need to training the model over and over again. As the model is initialised randomly every run, there has been looked at the effect of this on its performance.

References

[1] Thushan Ganegedara. Stock market predictions with lstm in python. `https://www.datacamp.com/community/tutorials/lstm-python-stock-market`, May 3rd, 2018. Accessed on: 2019-20-05.

[2] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks. `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`, May 21, 2015. Accessed on: 2019-26-05.

[3] Christopher Olah. Understanding lstm networks. `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, August 27, 2015. Accessed on: 2019-26-05.

[4] Kalen Smith. The 7 pitfalls of moving averages. `https://www.investopedia.com/articles/trading/11/pitfalls-moving-averages.asp`, June 25th, 2018. Accessed on: 2019-25-05.

[5] Tianyi Wang and Zhuo Huang. *The Relationship between Volatility and Trading Volume in the Chinese Stock Market: A Volatility Decomposition Perspective*. 2012.