university of groningen / faculty of science and engineering

Stock prediction using a Hidden Markov Model versus a Long Short-Term Memory

Bachelor's Project Thesis

Bram de Wit, b.de.wit.2@student.rug.nl,
Supervisor: Prof. Dr. L.R.B. Schomaker

Abstract: This study will compare the performance of a Hidden Markov Model (HMM) and a Long Short-Term Memory neural network (LSTM) in their ability to predict historical AAPL stock prices. Approximately one hundred other stocks will be used as context vectors in order to predict the following price. This problem is a typical time-series problem, where the models will try to predict the next time step. The performance of the two models will be compared using root mean squared error (RMSE), correlation, mean absolute percentage error (MAPE) and a comparison of the fractal dimension of the predicted stock price sequence. Using k-fold cross validation, for both models the best parameters were chosen. The performance of the best performing models is compared. The results showed that the HMM had a higher performance compared to the LSTM with a RMSE of 2.49 and a MAPE of 4.72, as well as a better fitting fractal dimension when compared to the fractal dimension of the actual data.

1 Introduction

The stock market has always been an interesting field due to its complexity and unpredictable behavior. This was already argued early in the nineteen hundreds in Cowles (1933) and Cowles (1944), where it is argued that stock prices take random walks. In Fama (1965) the efficient-market hypothesis is introduced, in which it is argued that stock prices always represented their true values. Although this hypothesis never has been invalidated, there is still research done in order to find a way to 'beat the market'. In earlier days certain calculations resulting in 'technical indicators' were used, that were supposed to indicate what the future direction of the stock price will be. Most technical indicators that are used today are described in Welles Wilder (1978). Knowing what the price will do in the future can be very beneficial, which is one of the reasons why it is such a popular topic. Essentially this problem can be classified as a time-series analysis, where the goal is to predict what will happen in the next time step or period. The rise of neural networks methods for time series, under which recurrent architectures such as the Long-Short Term Memory (LSTM), has lead to new research dedicated to stock price forecasting.

1.1 Problem description

The stock price is influenced by many factors under which the current season, consumer trust, how well a company is doing, what the crowd thinks, the current political state and the economical state of a country. The information needed to predict a stock price can be thought of as a multidimensional vector, moving in time.

To explain this using a metaphor: imagine a comet flying in 3-dimensional space. It has a direction, given by a vector of three elements (x, y, z). A historical set of these vectors describe the direction of the comet, which can be used to predict where the comet will be in a next time step. There can be gravitational attractors in this space, for example a planet with an gravitational force. This influences the direction of the comet. If all the attractors in this space were known as well as how they influence the comet, the state of the comet can be derived. This makes the assumption that there is no noise in the space.

The stock price can be thought of in the same way: it is determined not by a three-dimensional but N-dimensional state vector. In theory, if it is known how a space with all its attractors looks like, and in which way they will influence the stock price,

one could make predictions about the state of the stock price in the future, using a time-series of state vectors. However, the movement of stock prices is partly deterministic but chaotic, and partly a stochastic process. It includes randomness which make it hard to make accurate predictions. Furthermore, this space is extremely high dimensional, and the stock price is influenced by such a high number of factors, that it is practically impossible to fit these all in a model. The process of defining the n-dimensional space of the stock price together with the attractors is an extremely complex task. This study will try to capture this task using two different models, and will then compare the performance of these models. The first model that will be used is a Hidden Markov Model, which is based on Markov chains. Rabiner (1989) gives a detailed description on the workings of a Hidden Markov Model, which will not be discussed in depth here. The second model that will be used is a LSTM neural network as introduced by Hochreiter (1994).

1.2 Background

Over the years, researchers have extensively searched for a model which captures the trend of a market. Several studies that use different models to predict the stock price are given below.

Hidden Markov Model

A well known method that is used for time-series analysis is the Hidden Markov Model. This model learns the probabilities of (hidden) state transitions and the observation probabilities (in this case some distribution) given a certain hidden state. This allows for calculating the probability of observing a sequence and determining which state is most likely at the current time. Hidden Markov Models are often used for classification, since they can easily calculate probabilities of observation sequences. There is however no clear or straight forward method how to use a Hidden Markov Model for predicting time-series. Several methods can be used to make predictions with a Hidden Markov Model. One method introduced by Hassan (2005) is calculating the probability of an observation, searching for a similar probability with this observation in the past, and use the historical data at that time in the past to make a prediction for the current time. Nguyen (2018) experimented with a range for the number of observations and hidden states using Hassan's approach. Hassan (2009) also uses a hybrid model with fuzzy logic and a HMM to predict stock prices. However, one disadvantage of using a Hidden Markov Model is that it is not able to learn long term dependencies, since a transition between states takes into consideration only the current time step. However, there could be an indicator of what will happen next, which was present multiple time steps in earlier.

Long Short-Term Memory

Long-term dependencies were in theory solved by recurrent neural networks, but research has shown that it still raised a problem for basic recurrent neural networks Bengio (1994). It was shown that although information is passed to the next time step, over a longer period of time, this information vanishes, which is called the vanishing gradient problem. To solve this problem, Hochreiter and Schmidhuber introduced a new type of recurrent neural network called the Long Short-Term Memory (LSTM) network (Hochreiter (1994)). A LSTM consists of memory cells that allow these type of networks to learn long term dependencies. A LSTM cell contains a cell-state, which is updated every time-step. Due to the cell-state, there is a constant error flow, preventing vanishing gradients. Several variations were made to this basic architecture, such as forgetting and adding peephole connections, for details see Gers (2000b) Gers (2000a) Greff (2017). Due to its state of art performance in the field of predicting time-series, the LSTM was applied to stock markets, which is done in Pang (2018), Gao (2018).

1.3 Research question

The aim of this study is to compare a Hidden Markov Model and a LSTM neural network in their ability to capture stock movements by training on historical data, and as a result of this their ability to predict stock prices. The reason these two models are chosen is because of the fundamental differences between these two models. The Hidden Markov Model relies on statistics and distributions, and therefore probability maximization, whereas a LSTM searches for relations in the data set. The

aim of this study is to determine which of these two models works best on stock data. This gives us the following research question:
Is there a significant difference in the ability of predicting stock prices between a Hidden Markov Model and a LSTM neural network?
The following section explains the Hidden Markov Model and LSTM network in more detail as well as how the two models were used for stock market prediction. Furthermore it explains how the performance of the models was measured. In the result section the results of both models are presented separately and in the discussion section these results are discussed.

2 Methods

This section will describe in detail how this study is constructed, which data is used, how the models are implemented and which measures were used.

2.1 Data

As described in the previous section, the stock price is determined by context vectors. This study will use 110 stocks taken from the SP500, including the AAPL (Apple) stock, which is the stock that the two models have tried to predict. All the stocks in the data set have data available from 1 January 1990 until 1 June 2019. These stocks will be used as the context vectors for the AAPL stock that the models try to predict. Other additions to this context vector are omitted in this study, due to unavailability of this data in a structured form. However, this study does not claim that these other additions to the context vector do not have a significant influence on the AAPL stock movements. The data of a single stock contains four daily values: open price, close price, highest price of that day and the lowest price of that day. In general there are five trading days in a week, except for national holidays in America. In this study, only the close price of a stock is used when training and predicting stock prices. All data was retrieved from Yahoo. Historical stock prices from 1 January 1990 until 1 June 2019 results in approximately 7000 data points (trading days) per stock.

Data preparation

In order to create enough data for the two models to test and train on, linear interpolation was applied to every stock. Then, in order to prevent overfitting, a random noise of 0.01% was added to the data set. After this, the data was split into two distinct groups using odd-even splitting, where one group consisted of all the data points that had an even index and the other group consisted of all the data points that had an odd index. In this study data was always trained on one of these groups and then tested on the other group.

2.2 Hidden Markov Model

A Hidden Markov Model is a statistical model which consists of the following the parts:

- N = the number of hidden states
- T = the number of time steps/observations in an observation sequence
- π = initial state probabilities for each N
- A = the transition matrix, with size $N \times N$
- B = the emission matrix, with size $N \times z$, where z is the number of markets used in the model

A Hidden Markov Model works with observation sequences. An observation sequence has T time steps. Every time step t has a corresponding state for this time step. Transitions between states are defined by the transition matrix, which determines the chance of moving from one state to another state in the next time step. These states all have a probability of emitting the possible observations. Observations can be discrete or continuous. This study will use continuous observations, given the nature of the stock data. To model probabilities for continuous observations, distributions are needed. An example of how a Hidden Markov Model would generate or describe an observation sequence is shown in 2.1. A certain state at time t yields a distribution, from which a sample can be generated. After this the model moves to the next state and repeats this process. Furthermore, four algorithms are needed when working with a Hidden Markov Model:

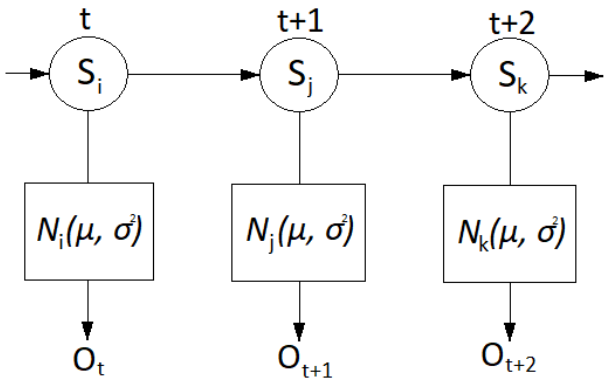

Figure 2.1: This figure shows the process of generating a sequence with a Hidden Markov Model with continuous observations, hence distributions.

1. The forward algorithm. This algorithm determines the likelihood of an observation sequence. This algorithm starts at $t = 1$.

2. The backward algorithm. This algorithm has the same functionality as the forward algorithm, but starts at $t = T$, where T is the latest time step of the observation sequence, and then moves backwards in time.

3. The Viterbi algorithm. This algorithm finds the state sequence that would most likely have emitted the given observation sequence.

4. The Baum-Welch algorithm. This algorithm maximizes the likelihood of the model, given a set of observation sequences.

These algorithms make it possible to train a Hidden Markov Model given a set of observations, allow the model to determine the probability of an observation sequence and determine the state sequence that is most likely to have emitted the current observation sequence.

HMM for stock price forecasting

An observation sequence in this study looks like z parallel vectors, each containing T historical daily close prices for every z markets. Instead of a single normal distribution (as in figure 2.1) this study will use Gaussian Mixtures. This is done as there is no a priori reason to assume that stock prices per state are distributed unimodally like a Gaussian. The GMM approach allows to model distributions with multiple peaks. Each state in the Hidden Markov Model of this study will have z Gaussian Mixture Models to describe the emission probabilities, where z is the number of markets that were used.

Training The Hidden Markov Model is trained using a sliding window. This sliding window of length 200 moves over the data with steps of $t = 1$, generating observations. The label of each observation is the value of the stock at $T+1$. Thus, the first observation will yield the values for $1 <= t <= 200$ with as label the value at $t = 201$, the second observation will yield the values for $2 <= t <= 201$ with as label the value at $t = 202$, and this is repeated for the complete data set. After the model is trained on the first observation using the Baum-Welch algorithm for 200 epochs, a prediction is made for $T+1$. After this, the next observation is presented to the model and is further trained on. Then a prediction is made for that observation, continuing this for all observations.

Predicting There are multiple methods how one can use a Hidden Markov Model for stock prediction. In the literature presented in the introduction, one method that was used is given one observation, search an observation in the past that yields the same likelihood given the trained Hidden Markov Model. In the past observation, calculate the difference between T and $T+1$ and add it to the current value at t, yielding the prediction.

A different method that could be used is given an observation sequence, first compute the most likely state sequence and determine the most likely next state that will be reached using the transition matrix of the trained HMM. Then, the Gaussian Mixture of this state can be used to generate the mean, and use this mean as the prediction. However, when having a Gaussian Mixture, this mean does not always describe the most likely value of this distribution.

When having a mixture of Gaussians, it would be more reasonable to select one of the Gaussians in the mixture by chance, and output the mean of this Gaussian as the prediction.

In this study, in order to make a prediction, the

observation sequence will be given to the Viterbi algorithm. This algorithm will yield the state sequence that would most likely have omitted the given observation sequence. Only the last state s_t of this sequence is remembered, the state at the current time t, and using the transition matrix A of the trained model, the state s_{t+1} for $t+1$ will be chosen such that $A[s_t, s_{t+1}]$ is the highest probability in that row. This state contains a Gaussian Mixture Model. In order to make a prediction, the probability density function (PDF) is converted to a cumulative distribution function (CDF). Then, a random value i along the y-axis is generated. The x-value of our the intersection of the line $y = i$ and the CDF is taken as the predicted value.

2.3 Long Short-Term Memory

A basic neural network only has connections that go forward. This architecture is not ideal for time-series problems, as this does not allow for long-term dependencies (where certain events depend on events in the past, with some time in between). When trying to solve time-series problems with neural networks, a recurrent neural network is often used. This neural network has recurrent connections, thus neurons passing on information to the next layer, but also to themselves for the next time step. In theory, this allows for long-term dependencies. However, as time goes by, this information vanishes for long-term dependencies. Therefore, a LSTM, which is a modified version of a recurrent neural network, is used for this study. A LSTM is able to handle long-term dependencies and therefore is currently one of the most often used models for time-series problems.

The following formulas are used when presenting input to a LSTM:

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \tag{2.1}$$
$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \tag{2.2}$$
$$\hat{C}_t = \tanh(W_C * [h_{t-1}, x_t] + b_C) \tag{2.3}$$
$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{2.4}$$
$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \tag{2.5}$$
$$h_t = o_t * \tanh(C_t) \tag{2.6}$$

The input of a LSTM model consists of a vector with all features at the current time step t. In this study the number of input nodes equals z, the number of markets used for predicting the AAPL stock. This input is then fed to a LSTM-cell, which has a cell-state with h nodes. A vanilla LSTM-cell first concatenates the output of the LSTM-cell at $t-1$ with the current input. Then it determines using a forget weight matrix using a sigmoid activation function which part of the current cell-state should be forgotten 2.1. After updating the cell-state it determines which information of the current input should be added to the cell-state 2.2, and a *tanh* layer that determines what of this can be added to the cell-state 2.3. This yields the cell-state C_t at time t 2.4. The output of the cell state is then determined by 2.5. This yields the input for the next layer of the network, as well as the input for the next time step, with which the input is then concatenated. This architecture, using a cell-state, allows for long-term dependencies, since the formulas for calculating the new cell-state allow for a recursive constant derivative. The LSTM has a constant error flow through time. Therefore, over a long period of time, the derivative does not go to zero.

LSTM for stock price forecasting

The LSTM can be used for stock price forecasting by presenting a vector containing all daily close prices for all z markets. This is then passed to the first LSTM-cell. The output of this cell will go through a dimension reduction layer. A common problem with LSTM networks when predicting time-series is that they learn to output the current value as the prediction for $t+1$. Although this often reduces error quite efficiently, it is often not the preferred solution, as the model does not learn the underlying relationships in the data. In order to prevent this, a dimension reduction layer is added so that the LSTM network is forced to compress the information it has. When the information is compressed, the LSTM will not output the same value. Then, the dimension-reduction layer passes its output to the last layer of size z. The output then yields a vector which is a prediction at $t+1$ for all z markets. An example of such an architecture is shown in figure 2.2.

Training When training the LSTM, batches of size 200 are used. The batches are created with a 100 point/day interval. In order to prevent overfit-

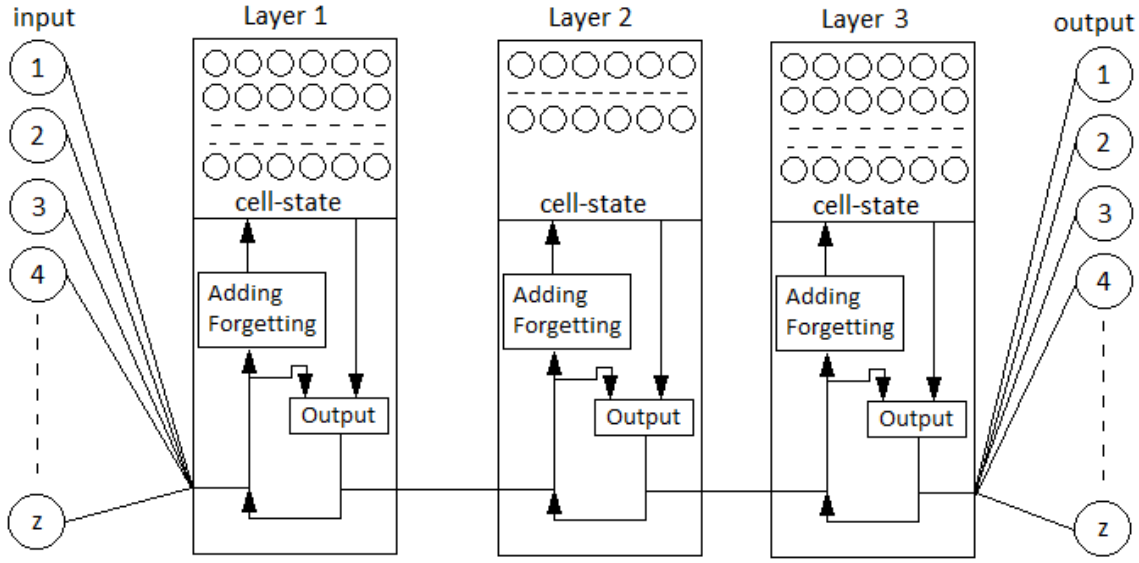

Figure 2.2: This figure shows a LSTM architecture with a dimension reduction layer (layer 2). z represents the number of markets used for prediction.

ting in the network, a dropout of 20% per layer is used. As error measurement to train the network, the mean squared error is used, and ADAM optimizer is used for the weight updates. When training, the network was trained for 200 epochs, where one epoch looped over all batches once.

Predicting For every input vector that is presented, an vector of the same length will be the output. This output vector will be the prediction for the next time step. The relevant AAPL prediction can be extracted from this vector as well.

2.4 Performance measures

In order to evaluate the performance of the model, this study will look at multiple measures.

Mean Absolute Percentage Error

The Mean Absolute Percentage Error (MAPE) is a measure which expresses the prediction accuracy of a model. The measure is represents the (averaged) expected percentage error when making a prediction. The formula used when calculating the MAPE is:

$$MAPE = \frac{100}{n} \sum_{t=1}^{n} |\frac{R_t - P_t}{P_t}| \tag{2.7}$$

In 2.7, R_t is the real value at time t, P_t is the predicted value at time t and n is the number of predictions.

Root Mean Squared Error

The Root Mean Squared Error (RMSE) is a similar measure for comparing the difference between predicted values by a model and the actual values of the data that is frequently used in machine learning. The RMSE is calculated using the following formula:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(R_t - P_t)^2}{n}} \tag{2.8}$$

In 2.8, R_t is the real value at time t, P_t is the predicted value at time t and n is the number of predictions.

Correlation

A different measure that can be used to measure the quality of the prediction is to determine the

correlation between the predicted time-series and the actual time-series. The correlation in this study is calculated with the following formula:

$$cov_x = \sqrt{\sum_{t=1}^{n}(x_t - \mu_x)^2} \tag{2.9}$$

$$CORR = \frac{\sum_{t=1}^{n}((P_t - \mu_P) * (R_T - \mu_R))}{(n-1) * cov_P * cov_R} \tag{2.10}$$

In 2.10, n is the number of predictions, cov_x is the covariance of time-series x, P is the predicted time-series and R is the real time-series. The correlation gives insight about how well the model is able to capture the trend of the data, even if the predicted values have a certain error.

Fractal dimension

The fractal dimension of a signal can be calculated in order to determine its complexity. In order to see if a model is able to capture the process of the AAPL stock, the fractal dimension of the predicted time-series should be as close as possible to the real time-series. Therefore, the fractal dimension is calculated for both time-series, and compared with each other. There are multiple methods how one could calculate the fractal dimension. This study will use the fractal dimension that is developed for stock prices specifically and is calculated as follows:

$$HL_1 = \frac{\max(X[0...n/2]) - \min(X[0...n/2])}{n/2} \tag{2.11}$$

$$HL_2 = \frac{\max(X[n/2...n]) - \min(X[n/2...n])}{n/2} \tag{2.12}$$

$$HL = \frac{\max(X) - \min(X)}{n} \tag{2.13}$$

$$D = \frac{\log(HL_1 + HL_2) - \log(HL)}{\log(2)} \tag{2.14}$$

This formula can be used to create a moving fractal dimension of a sequence. In 2.14, X is a certain time-series and n is the length of the sliding window used to create a moving fractal dimension. This function yields a line describing the fractal dimension over time. The quality of the predicted time-series can be described by the similarity between the fractal dimension line of the predicted time-series and the fractal dimension line of the real data.

2.5 Hyperparameter determination

Both models have several hyperparameters that influence the performance of the model. This study will look at a range for the number of hidden states used in the Hidden Markov Model. For the LSTM, this study will look at a range for the number of dimensions the cell reduction layer. Before comparing the models, the optimal hyper parameters are determined for both models. This is done by applying k-fold cross validation. The mean across all runs is calculated, as well as the standard error of the mean (SEM). The standard error of the mean gives a measure if a certain value of a parameter yields a significant better performance compared to other parameters. Using the results of the k-fold cross validation, the best parameters for each model are selected using the 'elbow method', which is the point in the graph where a further change in parameters does not yield a significant better performance. Then both of the best performing models are trained on the same data set and tested on the same data set, after which the performance is compared.

3 Results

In the following section, first the results of the k-fold cross validation are presented for both models separately. After this, the results of the best performing models trained on the same data set are presented.

3.1 K-fold cross validation

HMM

In figures 3.1, 3.2 and 3.3 the results of the k-fold cross validation for the Hidden Markov Model are presented. From the figures it can be observed that the performance increases as the number of hidden states increases. However, this increase stagnates when the number of hidden states is greater or equal to six. In all three figures it can be seen that the increase in performance decreases as more hidden states are added to the model. Figure 3.2

shows that seven hidden states results in an increase of the MAPE on average, which suggests an 'elbow' at six hidden states. Therefore, this study will use six hidden states in the final HMM model.

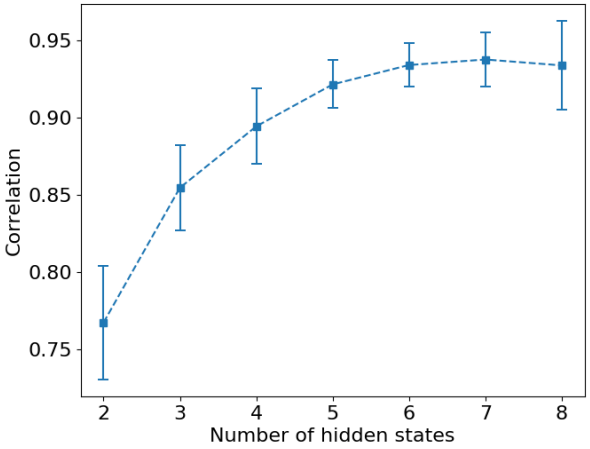

Figure 3.1: Graph showing the error bars (standard error of the mean) for the correlation of the prediction of the Hidden Markov Model and the actual data, where the squares represent the means. The y-axis represents the correlation (a value between -1 and 1) and the x-axis represents the number of hidden states that were used in the model.

LSTM

The results for k-fold cross validation in order to determine which number of dimensions to use in the reduction layer of the LSTM network showed that there was no difference in correlation, MAPE and RMSE for the different number of dimensions. K-fold cross validation was tested for a range of five dimensions up until ninety dimensions. In order to reduce visual noise, the figures are not presented here. Due to the fact that there was no best performing number of dimensions in the reduction layer, this study will continue using the number of dimensions that yielded the lowest average MAPE, RMSE and the highest correlation, which means that 45 dimensions will be used in the reduction layer.

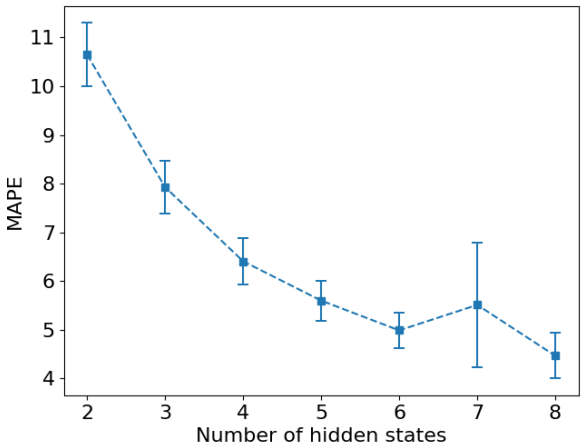

Figure 3.2: Graphs showing the error bars (standard error of the mean) for the Mean Absolute Percentage Error of the prediction of the Hidden Markov Model, where the squares represent the means. The y-axis represents the value of the Mean Absolute Percentage Error and the x-axis represents the number of hidden states that were used in the model.

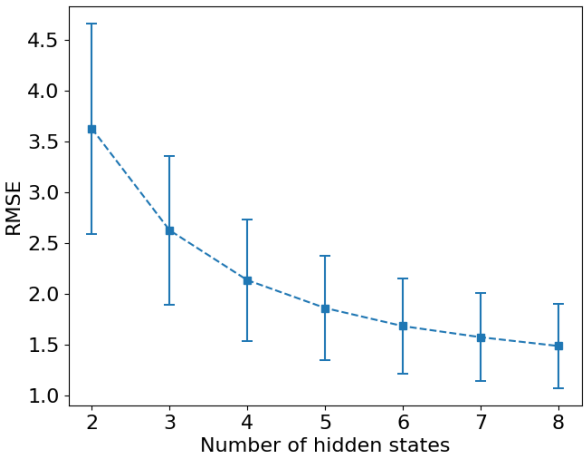

Figure 3.3: Graphs showing the error bars (standard error of the mean) for the Root Mean Squared Error of the prediction of the Hidden Markov Model, where the squares represent the means. The y-axis represents the value of the Root Mean Squared Error and the x-axis represents the number of hidden states that were used in the model.

3.2 Prediction results

The results presented in the following subsection will show measures for the predictions of both models using the hyperparameters that were determined above. In table 3.1 the values of the measures of both models are presented. The predictions of the HMM are presented in 3.4. Figure 3.5 shows the same predictions, but only for the most recent time steps. In figure 3.6 the fractal dimension over time of the predictions of the Hidden Markov Model are shown for the same period as in 3.5. The predictions themselves are presented in 3.7. Figure 3.8 shows the same predictions, but only for the most recent time steps. In figure 3.9 the fractal dimension over time of the predictions of the LSTM are shown, for the same period as in 3.8.

Table 3.1: This table shows the values for all the measures for the prediction of both models.

	MAPE	RMSE	Corr	% Direction
HMM	**4.72**	**2.49**	0.99	0.50
LSTM	12.72	5.35	0.99	**0.51**

4 Discussion

When looking at the results that were found in this study, it can be observed that the HMM had a better performance compared to the LSTM. The MAPE and the RMSE for the prediction of the HMM was lower than the MAPE and RMSE of the prediction of the LSTM. However, it only predicted the correct direction of the price 50% of the time, which is not better than random guess. This also goes for the LSTM, which predicts the price direction correct 51% of the time. The correlation of both predictions were the same.

When looking more closely at the predictions of the HMM and its fractal dimension, it can be concluded that the HMM is able to capture the chaotic characteristics of the time-series. It can be seen that it follows the movements of the stock price closely and that it is able to follow different trends in the stock price. However, it also raises the question whether the HMM simply learned to predict a value close to previous value. If this is the case, it would also explain the high similarity between the fractal dimension of the actual data and the fractal dimension of the prediction. If the HMM used this 'copy trick', this means that the HMM did not actually capture the underlying processes that determine the stock prices. The fact that the HMM did only predict the right direction of the price movement 51% of the time supports this. This 'copy trick' can occur when using a HMM when the observation used is too short.

When looking at the performance of the LSTM model, previous studies have shown better performance. It can be observed from the prediction plot that although the prediction does seem to follow the general trend of the actual data, it is unable to predict the high volatile movements of the stock. This can be explained by the number of dimensions in the reduction layer, since this number of dimensions is small, a smoothing takes place. This clearly can be seen in the prediction. Furthermore, the model is trained to predict the stock price when it was worth 1$, as well as when it was worth 170$ at the end. This means the model tries to predict the small stock movements as well as the big stock movements. What can be observed is that when the stock is worth the least, it predicts relatively too volatile movements, whereas if the stock is worth the most, it predicts relatively too small movements. It can be expected that it trains in this way, since on average this will yield the lowest error. It can be concluded that this model did not capture the underlying model that determines stock movements. This can also can be seen the plot showing the comparison between the fractal dimension of the prediction of the LSTM model and the fractal dimension of the actual data (figure 3.9). Although it has periods in which it follows the actual fractal dimension, it also shows periods in which the fractal dimension of the prediction is far off.

The conclusion of this study is therefore that given the results, the HMM is the better performing model when predicting stock prices when comparing it with the LSTM. The MAPE and RMSE of the HMM prediction were more than twice as low as the MAPE and RMSE of the LSTM prediction. Furthermore, the fractal dimension of the prediction of the HMM model has a better fit when comparing it to the fractal dimension of the actual data when comparing it with the fit of the prediction of the LSTM. However, as stated above, it might be the case that the HMM used the 'copy trick' in order to make predictions. If this would

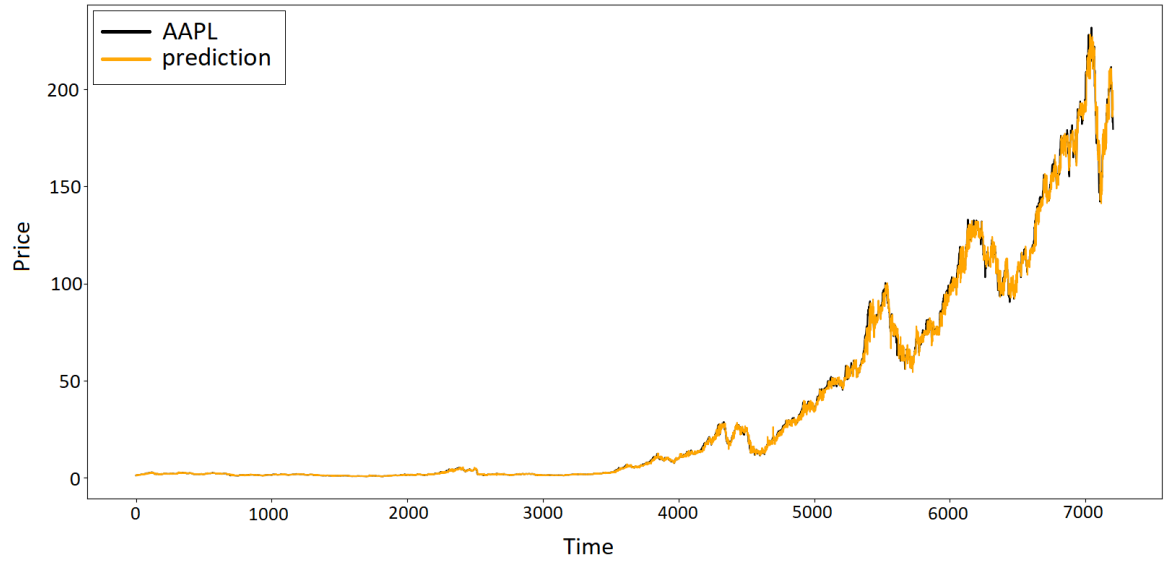

Figure 3.4: This figure shows the predictions made by the HMM versus the actual data. The reader is referred to the discussion section for further details.

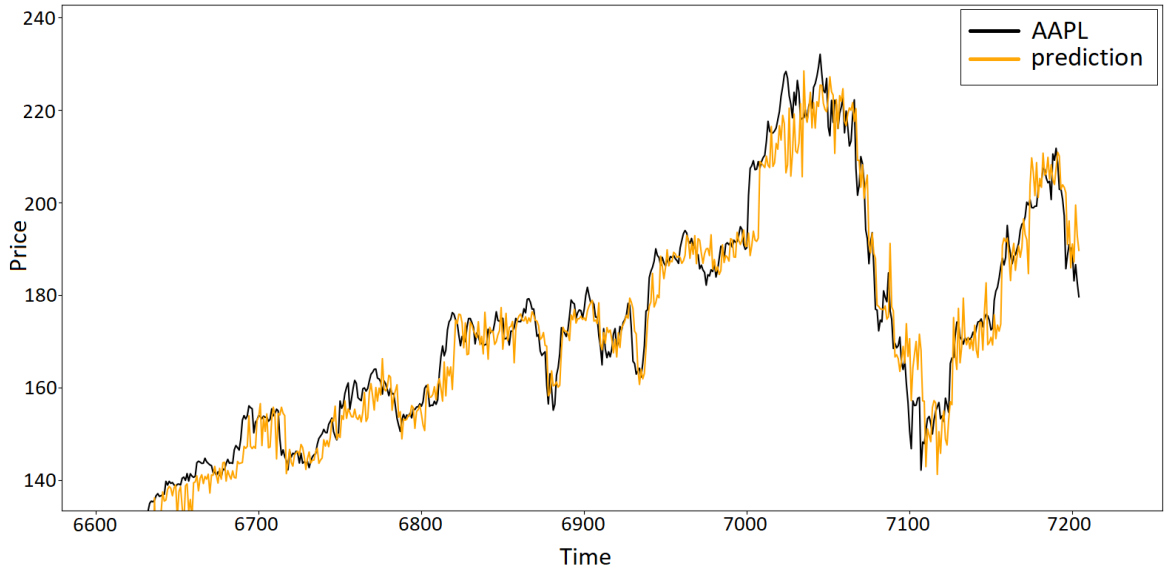

Figure 3.5: This figure shows the predictions made by the HMM versus the actual data, zoomed in on the latest predictions. The reader is referred to the discussion section for further details.

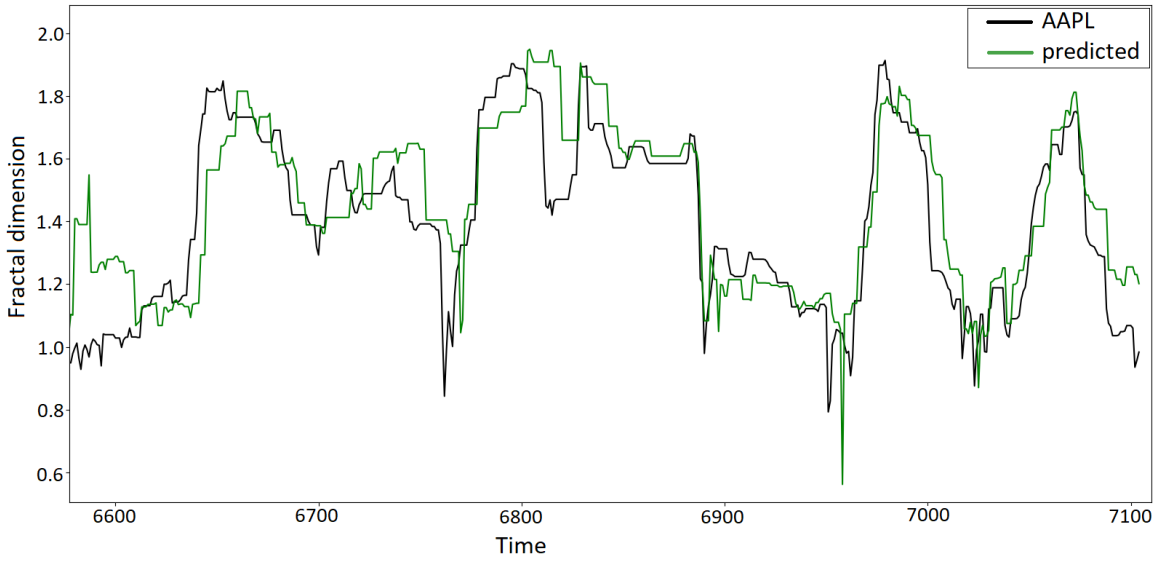

Figure 3.6: This figure shows the fractal dimension over time of the prediction made by the HMM versus the fractal dimension over time of the actual data. It shows only the fractal dimension of the latest time steps in order to reduce visual noise. The reader is referred to the discussion section for further details.

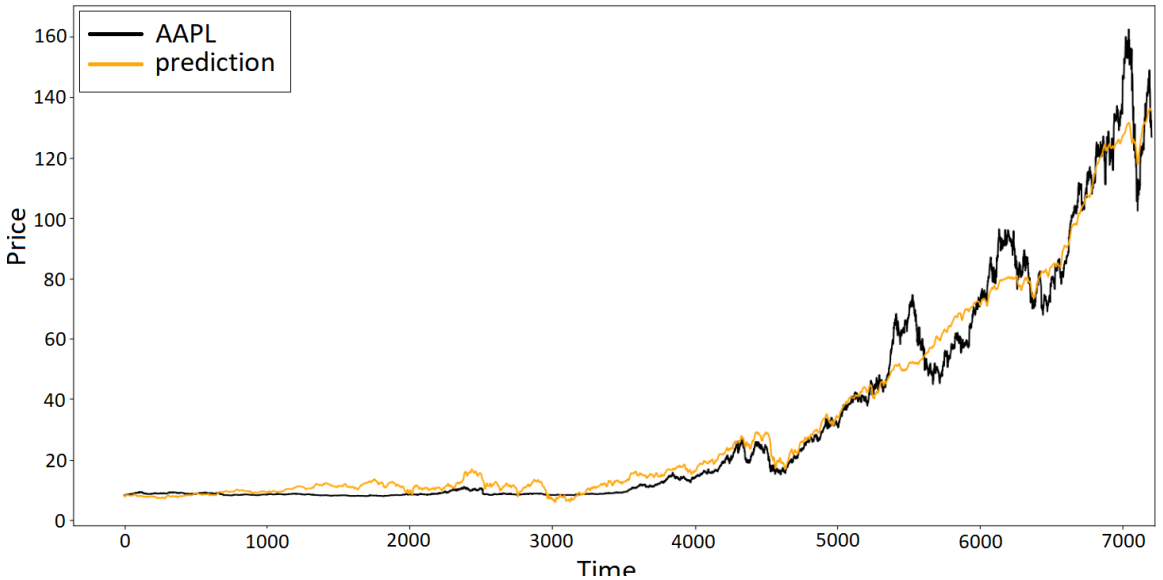

Figure 3.7: This figure shows the predictions made by the LSTM versus the actual data. The reader is referred to the discussion section for further details.

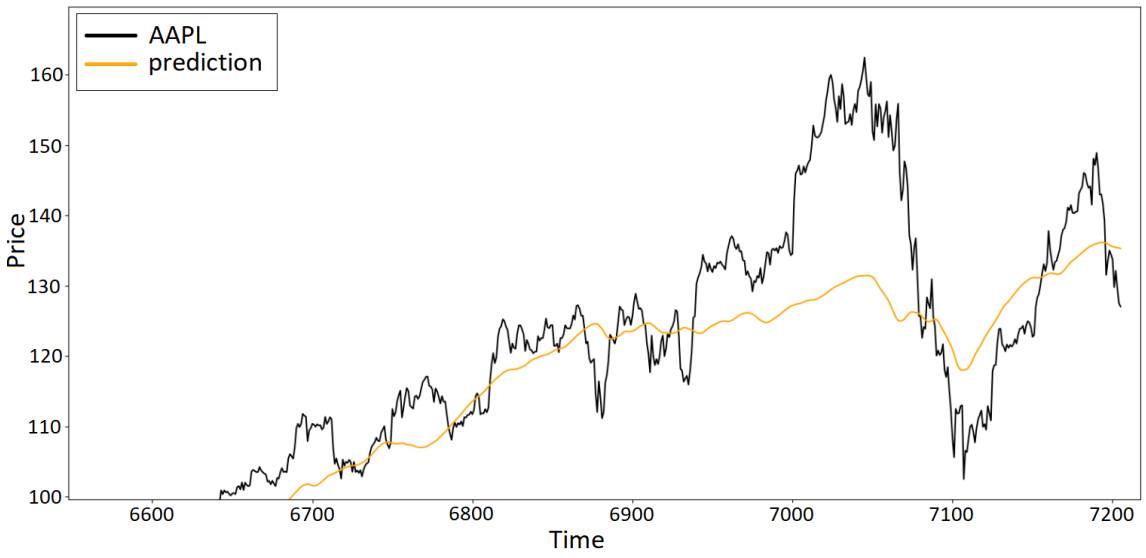

Figure 3.8: This figure shows the predictions made by the LSTM versus the actual data zoomed in on the latest predictions. The reader is referred to the discussion section for further details.

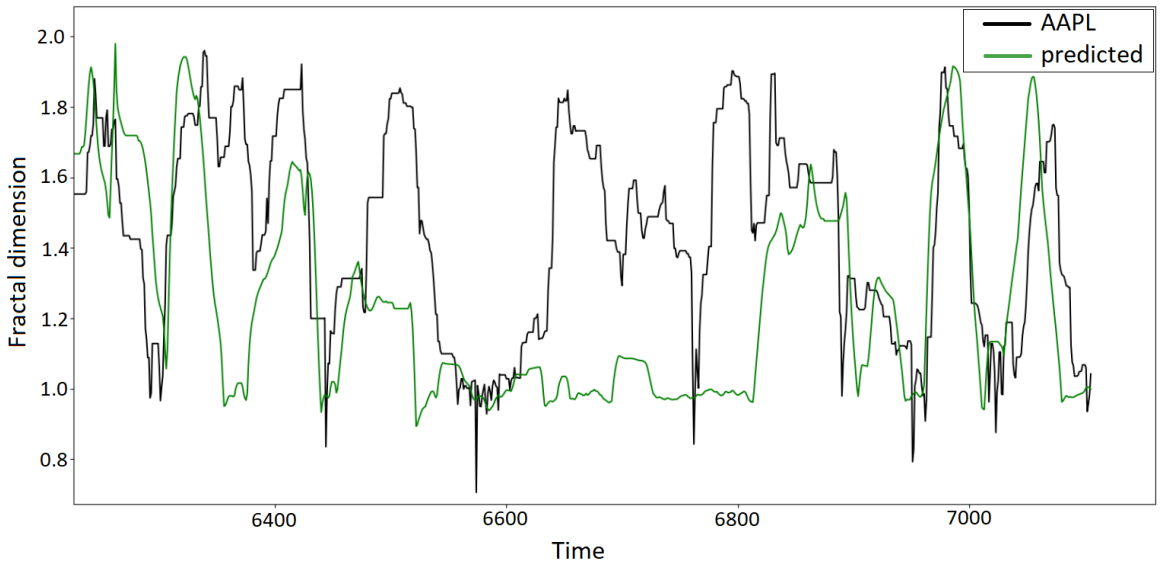

Figure 3.9: This figure shows the fractal dimension over time of the prediction made by the LSTM versus the fractal dimension over time of the actual data. It shows only the fractal dimension of the latest time steps in order to reduce visual noise. The reader is referred to the discussion section for further details.

be the case, it would mean that the HMM did not actually capture the underlying chaotic processes present in the data. Furthermore, it might be that the LSTM performance presented in this study is not the best performance that can be achieved using such a model.

4.1 Further research

It might be necessary to take a closer look at the HMM and how it makes the predictions in order to make sure that the HMM will not be able to apply the copy trick. One method could be to compress the input before it is used as input of the HMM. This would be similar to the approach used in the LSTM in this study, where the number of dimensions is reduced in the middle layer. The two models might be compared to accomplish this .One example of a different architecture where this can be achieved could be where the LSTM reduces the number of dimensions by processing the input, after which it feeds the compressed information to a HMM, which then works on the compressed information, gives an output, which is then translated back by the LSTM.

This study has omitted several factors that could be included in the context vectors used to make a prediction. Further research might be dedicated to finding which factors have the biggest influences, as well as how they interact with each other. Furthermore, one could try to exploit the models used in this study in order to improve performance. This could be done by adding more data, trying different architectures or a wider variety of hyper parameters.

References

Simard P. Frasconi P. Bengio, Y. Learning long-term dependencies with gradient descent is difficult. *Transactions on Neural Networks*, 5(2): 157–166, 1994.

A. Cowles. Stock market forecasting. *Econometrica: Journal of the Econometric Society*, pages 206–214, 1944.

A. 3rd Cowles. Can stock market forecasters forecast? *Econometrica: Journal of the Econometric Society*, pages 309–324, 1933.

E. F. Fama. The behavior of stock-market prices. *The journal of Business*, 38(1):34–105, 1965.

Chai Y. Gao, T. Improving stock closing price prediction using recurrent neural network and technical indicators. *Neural Computation*, 30:2833–2854, 2018.

Schmidhuber J. Gers, F. Recurrent nets that time and count. *Neural Computation*, 3:189–194, 2000a.

Schmidhuber J. Cummins F. Gers, F. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12:2451–2471, 2000b.

Srivastava R.K. Koutnik J. Steunebrink B. R. Schmidhuber J. Greff, K. Lstm: A search space odyssey. *Neural and Evolutionary Computing*, 27(10):2222–2232, 2017.

M.R. Hassan. A combination of hidden markov model and fuzzy model for stock market forecasting. *Neurocomputing*, 72:3439–3446, 2009.

Nath B. Hassan, M.R. Stock market forecasting using hidden markov model: a new approach. *5th International Conference on Intelligent Systems Design and Applications (ISDA'05), Warsaw*, pages 192–196, 2005.

Schmidhuber Hochreiter, S. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1994.

N. Nguyen. Hidden markov model for stock trading. *International Journal of Financial Studies*, 6(36), 2018.

Zhou Y. Wang P. Weiwei L. Chang V. Pang, X. Stock market prediction based on deep long short term memory neural network. *Proceedings of the 3rd International Conference on Complexity, Future Information Systems and Risk (COMPLEXIS 2018)*, pages 102–108, 2018.

L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 1989.

J. Welles Wilder. *New Concepts In Technical Trading Systems*. Trend Research, 1978.