# Dow Jones Trading with Deep Learning: The Unreasonable Effectiveness of Recurrent Neural Networks

Mirco Fabbri and Gianluca Moro
*Department of Computer Science and Engineering (DISI), University of Bologna,*
*Via Cesare Pavese, I-47522, Cesena, Italy*

Keywords: Stock Market Prediction, Trading, Dow Jones, Quantitative Finance, Deep Learning, Recurrent Neural Network, LSTM.

Abstract: Though *recurrent neural networks* (RNN) outperform traditional machine learning algorithms in the detection of long-term dependencies among the training instances, such as in term sequences in sentences or among values in time series, surprisingly few studies so far have deployed concrete solutions with RNNs for the stock market trading. Presumably the current difficulties of training RNNs have contributed to discourage their wide adoption.This work presents a simple but effective solution, based on a deep RNN, whose gains in trading with Dow Jones Industrial Average (DJIA) outperform the state-of-the-art, moreover the gain is 50% higher than that produced by similar feed forward deep neural networks. The trading actions are driven by the predictions of the price movements of DJIA, using simply its publicly available historical series. To improve the reliability of results with respect to the literature, we have experimented the approach on a long consecutive period of 18 years of historical DJIA series, from 2000 to 2017. In 8 years of trading in the test set period from 2009 to 2017, the solution has quintupled the initial capital, moreover since DJIA has on average an increasing trend, we also tested the approach with a decreasing averagely trend by simply inverting the same historical series of DJIA. In this extreme case, in which hardly any investor would risk money, the approach has more than doubled the initial capital.



## 1 INTRODUCTION

Past stock market prediction approaches, which were based on analysis of macroeconomic variables, led to the formulation of Efficient Market Hypothesis (EMH) (Fama, 1970) according to which the financial market is efficient and reflects not only the information publicly available (e.g. news, stock prices etc.) but also the future expectations of traders. This result, together with the study in (Malkiel, 1973) that showed accordance between the stock values and the random walk theory, corroborate the idea of the stock market unpredictability.

Despite these outcomes, the literature in stock market predictions is among the most long-lived researches, evidently in the attempt of rejecting the hypothesis of the market unpredictability, such as in (Lo and MacKinlay, 1988) and in (Malkiel, 2003) where, being random walk theory confirmed only within a short time window, the market trend should generally be predictable. The last two mentioned studies do not contradict the EMH because the possibility of predicting the stock prices trend is not necessarily linked to the market inefficiency but to the predictability of the financial macro variables.

As we have more extensively reported in Section 2, the stock prediction methods have ranged from auto-regressive ARIMA models (Box and Jenkins, 1970) to more advanced approaches also based on machine learning, including in the last decade the neural networks, using both structured and unstructured data, namely time series and free text such as news, posts, forums etc. (Mitra, 2009; Atsalakis and Valavanis, 2009a; Mostafa, 2010).

The novel proposals, which have improved the accuracy predictions of preceding approaches, are based on the combination of time series values with news or social network posts, such as in (Bollen et al., 2011; Domeniconi et al., 2017a; Akita et al., 2016). The drawback of combining financial time series and news or posts, is that such approaches require daily huge amount of fresh text which are almost impossible to gather in real time, even because the sources of news and social networks do not permit like in the



Fabbri, M. and Moro, G.
Dow Jones Trading with Deep Learning: The Unreasonable Effectiveness of Recurrent Neural Networks.
DOI: 10.5220/0006922101420153
In *Proceedings of the 7th International Conference on Data Science, Technology and Applications (DATA 2018)*, pages 142-153
ISBN: 978-989-758-318-6

past unconditional massive download of their data. In other words their application appear unfeasible in real time scenarios typical of the stock market exchange.

Recently the deep learning has become a wide and prolific area of research with successful breakthroughs in several scientific and business areas, such as speech and image recognition (Krizhevsky et al., 2012), autonomous driving, diagnosis of diseases or genomic bioinformatics (Esteva et al., 2017; Domeniconi et al., 2016; di Lena et al., 2015), astronomical discoveries and so on. The impressive effectiveness of deep learning solutions rely on novel neural networks with memory capabilities. In particular *recurrent neural networks* (RNN) are the first concrete learning technology capable of detecting long-term dependencies among the training instances, such as among consecutive words in sentences (Domeniconi et al., 2017b) or among consecutive values in time series.

Despite this capability of detecting temporal correlations or patterns, surprisingly few studies so far have deployed concrete solutions with for the stock market trading based on RNNs. One of the main reasons that have limited the wide employment of RNNs for predicting the stock market, is that defining and training a successful RNN is almost always a challenge. In fact the number of choices to discover an effective RNN, are much more large and mutually dependent with respect to the preceding approaches: concretely the choices start from the architecture design, such as the number of neuronal layers, the neuronal units in each layer, the number of connections among layers, the kind of activation functions in each neuron unit and so on, up to the discovery at training time from a wide number of hyper-parameters, those that maximise the test set accuracy.

The contribution of this work is the set of the successful choices that have led to the definition, implementation and training of a deep RNN for trading with the Dow Jones Industrial Average (DJIA), whose profits outperform the state-of-the-art. The solution [1] is also a trade-off among efficacy and efficiency in order to be applied in real time trading and therefore using only publicly available historical series of DJIA, without any further external data. The solution performs trading actions, such as buy/sell/none, from the ability of predicting accurately the closing price movements of DJIA, which appears unreasonable being in contradiction with the stock market unpredictability.

Our approach has quintupled the initial capital in the test phase of the last 8 years of trading, between 2010 and 2017, after a training phase between 2000 and 2009. The profit is about 50% higher than achievable via a Feed-Forward (FF) deep neural network, which is without explicit memory capacity. The DJIA has on average a positive trend and to evaluate the robustness of our solution, we have applied it also to the historically inverted DJIA, which consequently is on average a negative time series. In this extreme case, where investors would not risk money, the gain produced is 2.33 times the initial capital.

The study is organised as follows. Section 2 analyses the recent literature in the stock market prediction, from classical methods to those based on news and on social network posts. Section 3 describes the data set employed and the proposed solution with a brief background on recurrent neural networks. Section 4 reports the experiment results and how they outperform preceding results with DJIA based on neural networks. Section 5 summarises the work and outlines the future developments.

## 2 RELATED WORKS

The literature of stock market predictions is large being a long time research thread that goes from classic historical series forecasting to social media analysis.

A well-known research in (LeBaron et al., 1999), in contrast to the unpredictability of the stock market, explains the existence of a temporal lag between the issuance of new public information and the decisions taken by the traders. In this short time frame the new information can be used to anticipate the market.

In 2009 (Schumaker and Chen, 2009), following this idea that new information may quickly influence the trading, experimented the prediction of stock prices by combining financial news with stock prices through Support Vector Machine (SVM). They tried to estimate the stock price 20 minutes after the release of the new information achieving actually the best results with the combination of text news and stock values.

Successively other works proposed methods that combine the text mining techniques with data mining techniques. (Lin et al., 2011) combines K-means and hierarchical clustering to analyse financial report (formal record of the financial activities and position of a business, person, or other entity (Wikipedia contributors, 2018)) through quantitative and qualitative features. This approach allows to detect clusters of features and improves the quality of prototypes of the reports.

The most prominent approach used in time-series forecasting has been introduced in 1970 by Box and Jenkins and it is called ARIMA (Autoregressive inte-

[1] http://tiny.cc/dow_jones_trading

grated moving average (Box and Jenkins, 1970)); it shows an effective capability to generate short-term forecasts. ARIMA considers future values as linear combination of past values and past errors, expressed as follows:

$$\begin{aligned} Y_t = \phi_0 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \dots \phi_p Y_{t-p} \dots \\ + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \theta_q \varepsilon_{t-q} \end{aligned} \tag{1}$$

where $Y_t$ is the actual stock value, $\varepsilon_t$ is the random error at time $t$, $\phi_i$ and $\theta_i$ are the coefficients and $p, q$ are integers called autoregressive and moving average. The main advantage of this approach, is the ability of observing it from two perspectives: statistical and artificial intelligence techniques. In particular ARIMA methodology is known to be more efficient in financial time series forecasting than ANNs (Lee et al., 2007; Merh et al., 2010; Sterba and Hilovska, 2010). Some works that used ARIMA are (Khashei et al., 2009; Lee and Ko, 2011; Khashei et al., 2012).

Recent studies as (Lo and Repin, 2002) in behavioural finance and behavioural economy have applied cognitive psychology in order to understand the reasons that led investors to make certain choices and how these affect the market trend. This, together with ability of computers to process massive amounts of data, has given a strong boost to a research thread which tries to correlate the sentiment acquired from repositories of social opinion (as social media or journalistic news) to the market trend. Some of these studies use mining technique to predict stock prices (e.g. the DJIA) (Gidofalvi and Elkan, 2001), (Schumaker and Chen, 2006), or to make more accurate predictions detecting relevant tweets such as in (Domeniconi et al., 2017a). Furthermore, significant steps taken in natural language understanding (NLU) and natural language processing (NLP) achieved a new state-of-the-art in the field of market forecasting by exploiting deep neural networks (Akita et al., 2016; Wang et al., 2017; Fisher et al., 2016). The idea behind the use of NLP lies in the ability of extracting sentiments and opinions directly from unstructured data and exploits them to predict market trend. (Tetlock, 2007) has analysed the correlation between sentiment, extracted from news articles, and the market price trend with the conclusion that media pessimism may affect both market prices and trading volume.

Similarly (Bollen et al., 2011; Domeniconi et al., 2017a) have taken advantage of text mining techniques to extract mood from social media, like Twitter, and improve the DJIA index prediction; in both cases, the result obtained are very interesting because they achieved an accuracy of 86% and 88% respectively. Another consideration can be raised from Bollen et al. work relative to the *time-lag*. Their study showed that the best performance can be obtained grouping information from several past days to predict the next one.

Also in the deep learning thread, NLP methods have been used to improve language understanding and tackle financial market trading. (Akita et al., 2016) experiments the possibility of predicting stock values by exploiting existing information correlations between companies. For every day predictions, the set of news related to companies is converted in a distributed representation of features by using Paragraph Vector which is then used to feed a memory network along with stock prices. Then the model is trained to predict the output class, *buy* if the predicted close price is greater than real opening prices, otherwise *sell*.

(Ding et al., 2015) introduces a new technique, which proposes to automatically extract a dense representation of *event embeddings* from financial news. To do that, a word embedding approach is used to create a distributed representation of words, taken from Reuters and Bloomberg news, and used to feed a Neural Tensor Network (NTN) subsequently producing a distributed representation of event embeddings. These events are used to feed common models like FF or convolutional neural networks. The prediction has been done on S&P 500 stock and it showed an accuracy of 65% and an economical profit of 1.68 times the initial capital.

However, the works that use NLP and NLU to predict market movements are de facto inapplicable in real time scenarios because they should continuously collect and process large amounts of fresh, noise-free unstructured textual data from sources, such as Web news sites and social networks, which now, furthermore, prevent unconditional and massive data gathering.

There are, however, several studies that have employed neural network to forecast stock market using only structured data such as time series or financial data (Atsalakis and Valavanis, 2009b; Soni, 2011). (Olson and Mossman, 2003) attempted to predict annual stock returns for 2352 Canadian companies by feeding a neural network with 61 accounting reports obtaining better results than those obtained by regression-based methods. Similarly (Cao et al., 2005) has shown that neural networks outperformed the accuracy predictions of generalised linear models in Chinese stock market.

Although neural networks have provided good results in stock market forecasting, they do not achieve the same performance achievable through Deep Learning, as reported in (Abe and Nakayama, 2018). In this work different models of both shallow and FF deep neural networks are compared; they are fed by the combination of market and financial data to predict one-month-ahead stock returns of MSCI Japan Index.

In (Fischer and Krauss, 2017) LSTM networks have been compared with classical methods, among which *Random Forest*, for the prediction of the S&P 500 (Thomson Reuters) between the years 1993 and 2015. They have obtained an unstable effectiveness with significant differences in the following three phases:

- 1993-2000: The cumulative profit at the end of this period was about 11 times the initial capital, also overcoming the random forest based approach.
- 2001-2009: The cumulative profit was about 4 times the initial capital under-performing the *Random Forest*, which obtained a gain of about 5.5 times the initial capital. This result was attributed to a greater robustness of decision tree-based methods, like *Random Forest*, against noise and outliers, which plays out during such volatile times.
- 2010-2015: this was considered the most interesting period since it shows how LSTM networks reduce the prediction variability while keeping the initial capital unchanged and instead Random Forest leads to a loss of about 1.2 times the initial capital.

# 3 METHODOLOGY

## 3.1 The DJIA Time Series Dataset

In order to perform a comparative analysis with other results, we decided to use the Dow Jones Industrial Average (DJIA) as the target market. This index approximately reflects the US financial market status being a combination of the 30 most important stock market titles. There are several studies that use both classic mining and artificial intelligence techniques in forecasting the results using the DJIA as reference. The purpose of using deep learning approach is to abstract from one single instance and permit the model to catch generic recurrent patterns.

In order to guarantee a higher reliability during the testing phase, we have downloaded from Yahoo finance a long time series of DJIA between 01/01/2000 (1st January) and 31/12/2017 (31st December) (Yahoo! FinanceYahoo! Finance, 2017). This data set contains the open, high, low, close and adjusted close prices for every working day throughout these eighteen years.


Figure 1: Close price trend of the DJIA between 01/01/2000 and 31/12/2017.

In Figure 1 we can clearly observe that the DJIA closing values trend is positive on average. To properly evaluate the models ability in the prediction of the DJIA prices, we have divided the time series in the middle, generating a training set between 01/01/2000 and 30/06/2009 (3285 stock prices) and a test set between 01/07/2009 and 31/12/2017 (3285 values). The dimensions of the test set allow to have a high reliability with respect to the real predictive capabilities of the model.

## 3.2 Data Preparation

In order to compensate for the lack of prices on the closing days of exchange (holidays, week-ends), we have executed a first step of preprocessing by applying a convex function , obtaining the new prices as follow:

$$\frac{x_{prev} + x_{last}}{2} \tag{2}$$

where:

- $x_{prev}$: last price available
- $x_{last}$: first price available after the interval of missing values.

Several works have shown that there is a temporal lag between the issuance of new public information and the decisions taken by the traders. As shown by the experiments in (Bollen et al., 2011; Domeniconi et al., 2017a) the higher correlation between social mood and the DJIA is obtained by grouping unstructured data of several days and shifting the prediction for a certain time lag. Similarly, (LeBaron et al., 1999) illustrated how the new information, publicly available, can be used within a short time window to anticipate the market.

According with these approaches, we have aggregated several days of information in order to predict the closing price of the last day in the list. To do that we introduced an *aggregation* (*agg*) parameter that identifies the number of days to aggregate, so $agg = 3$

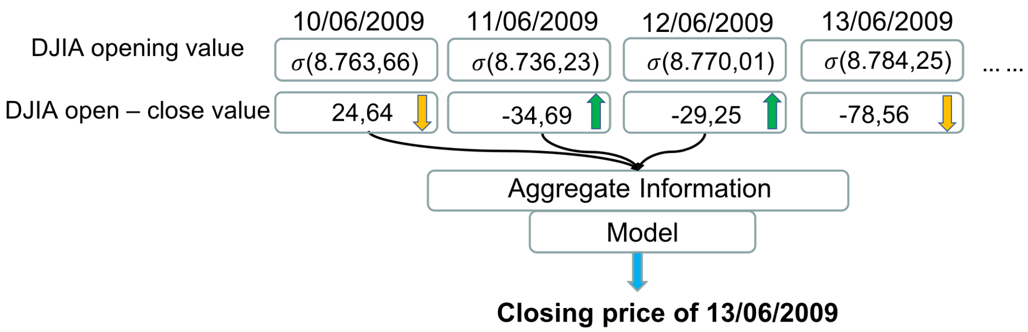


Figure 2: Example of the DJIA prediction process.

means the data aggregation from the three days, prior to the prediction day.

The diagram in Figure 2 summarises the DJIA prediction process. A normalisation process was applied, both at the opening and the closing prices, through z-score considering only the information available at time $t$.

$$Z = \frac{X - \mu}{\sigma} \tag{3}$$

For example, the z-score (both for opening and closing price) on 12/06/2009 was calculated considering only the prices available until 11/06/2009, leaving out those starting from 13/06/2009, respecting the conceptual integrity.

### 3.3 Recurrent Neural Networks Overview

*Recurrent neural networks* (RNN) have shown better learning capabilities than the traditional FF ones whenever the data exhibit complex temporal dependencies, which are typical of sequence learning, such as speech recognition, language modelling and time series.

As depicted in Figure 3 this is achieved by providing a retroactive feedback which feed the network activations from a previous time step as inputs to the network in order to influence predictions at the current time step. These activations contribute to create the internal state of the network which can hold long-term temporal information. This mechanism allows RNNs to dinamically and autonomously exploit different time windows over the input sequence history unlike FF networks that are bound to a pre-defined fixed size window.

While typical FF networks operate as a *combinatorial system*, associating an output patterns to an input patterns, RNN networks act as a *sequential system* associating an output pattern to an input pattern with a dependence on an internal state that evolves over time. The two most popular RNNs are Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997; Gers et al., 1999; Graves, 2013) and Gated Recurrent Unit (GRU) (Cho et al., 2014).

Both solutions are based on the RNNs but improve them by overcoming some architectural limitations such as the vanishing and exploding gradient problems which are caused by the *weight matrix* and the *activation function* (or rather its derivative).

In the recurrent networks, the backpropagation of the error occurs not only for all the levels as well as in the vanilla backropropagation, typically used in FF networks, but also for all the time instants provided as input to the network. This process is called *backpropagation through time* (BPTT) and can be summarized as:

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial h_t} \frac{\partial h_t}{\partial c_t} \frac{\partial c_t}{\partial c_k} \frac{\partial c_k}{\partial W} \tag{4}$$

The vanishing/explosion gradient is generated when the sizes of the time window considered by the network are high. The equation 4 summarizes the process by which the error is transported from the current time $t$ up to the first. The crucial point of BPTT is:

$$\frac{\partial c_t}{\partial c_k}$$

which allows to transport the error from a temporal instant to the previous one. The vanishing/exploding gradient problem is raised when the time windows is wide:

$$\frac{\partial c_t}{\partial c_k} = \prod_{k<i<=t} \frac{\partial c_i}{\partial c_{i-1}}$$

where the right side of the equation is the product of the terms of jacobi, therefore:

$$\prod_{k<i<=t} \frac{\partial c_i}{\partial c_{i-1}} = \prod_{k<i<=t} w^t diag(f^{'}(c_{i-1}))$$

where $f^{'}$ is the derivative of the activation function.

if $k$ is a small value and $t$ is a big value the result of the product will be a very large or very small

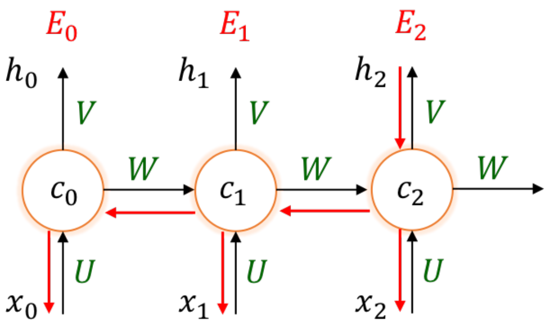


Figure 3: Back-propagation through time.

1. $U, V, W$: are the matrices of the weights shared among all the temporal instants.
2. $E_0, E_1, \ldots E_t$: are the errors committed in the various temporal instants.
3. $h_0, h_1, \ldots h_t$: are the predictions of the network in the various temporal instants.
4. $c_0, c_1, \ldots c_t$: are the hidden states of the network in the various temporal instants. They represent the 'memory' of the network.

value depending on whether the gradient is $> 1$ (the gradient will tend to 0) or $< 1$ (the gradient will tend to $\infty$).

To address this problem, the LSTM networks have special blocks called *memory units* in the recurrent hidden layer and they use the *identity function* (whose derivative is 1) as activation function so as to keep the error constant during the back propagation (CEC=Control error carousel).

In order to allow the network to learn, a *gate* concept has been introduced, which is a unit able to open or close access to the CEC:

- **Input**: it allows to add / update information to the memory cell state $C_t$ with respect to the past state.
  1. $\tilde{C}_t$ represent a proposal for the information to be written within the cell state.
  2. $i_t$ modulates the proposal ($\tilde{C}_t$) of the information to be written, or how much "part of information of the proposal to accept".

  The input gate is therefore determined by 'how much information to forget' from the cell state at the previous time ($C_{t-1}$) and 'how much information to add' at the current time ($x_t$).
- **Output:** it controls the output flow of cell activations into the rest of the network and it is determined by the cell state $C_t$.
- **Forget:** In the original proposal (Hochreiter and Schmidhuber, 1997) the forget gate was not present but was added later (Gers et al., 1999) to prevent LSTM from processing continuous input streams that are not segmented into subsequences.

  The forget gate's task determines *how much information* of the internal memory unit state should forget ($h_{t-1}$) and *how much information* should add as input at, current time, via the self-recurrent connection ($x_t$).

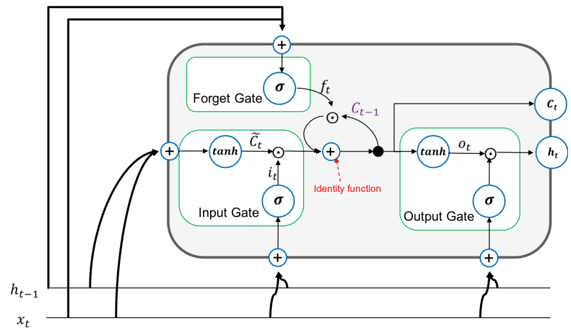


Figure 4: Architecture of a Long-Short Term Memory Unit.

The main difference between LSTM and GRU is the lack of the memory gates in the GRU networks. This simplification makes GRU faster to train and generally more effective than LSTM on smaller training dataset. On the other hand, the complexity of LSTM architecture makes them able to correlate longer training sequences than the GRU networks.

### 3.4 Design & Settings of Our Recurrent Neural Network

We have defined a recurrent neural network architecture based on LSTM with two hidden layers. The peculiarity of this choice is to have in first hidden layer twice the number of neurons contained in than the second one. This choice derives from the desire to allow the network to abstract from the specific information of the instances, enabling it to identify generic patterns.

The first two recurrent hidden layers were trained using the hyperbolic tangent (tanh) as activation function while in the output level a linear function was used. The network was fed using the opening prices of $n$ days, including the opening price of the target day, in order to predict the closing price of the target day by regression. This predicted value has been used to determine the type of action to be performed as:

$$action = \begin{cases} buy & \text{if } open - close_{predicted} > 0 \\ sell & \text{otherwise} \end{cases} \quad (5)$$

As first step, the closing values predicted by the network using the regression, was used to determine which action to take. Then the accuracy was calculated assuming a *+1* for each action / class expected correctly and *-1* otherwise. The resulting accuracy was calculated as:

$$Accuracy = \frac{correct\ predictions}{total\ predictions} \tag{6}$$

The tests were conducted analysing three types of networks:

1. *small network*:
   - 1 hidden layer: 32 neurons
   - 2 hidden layer: 16 neurons
2. *medium network*:
   - 1 hidden layer: 128 neurons
   - 2 hidden layer: 64 neurons
3. *large network*:
   - 1 hidden layer: 512 neurons
   - 2 hidden layer: 256 neurons

Then three types of gradient optimisation algorithms have been experimented:

- **Momentum**: it's a gradient optimization algorithm whose basic idea is the same as the kinetic energy accumulated by a ball pushed down a hill. If the ball follows the steeper direction, it will quickly accumulate kinetic energy and when it meets resistance, it will lose it. Since the gradient of a function indicates the direction along which it is growing faster, the momentum grows according to the directions of the past gradients. If the gradients calculated in the previous steps all have the same direction, the momentum tends to grow. If the gradients instead have different directions, the momentum tends to decrease. As a consequence we will have a faster convergence than classic *Stochastic Gradient Descent* limiting its variability. Weights update happens as

$$w = w - v_t$$
$$v_t = \gamma v_{t-1} + \lambda \frac{\partial C}{\partial W}$$

  where:
  - $\gamma$ (0.9): is the momentum's factor
  - $\lambda$ (0.01): is the learning rate
  - Decay: 0.0
- **Adagrad:** address the slow convergence problem of Stochastic Gradient Descent using an adaptive learning rate which allows a fine update for frequent parameters and a large update for rare ones.

$$w_{t+1}(i) = w_t(i) - \frac{\lambda}{G_t(i,i) + \varepsilon} \frac{\partial C}{\partial W}$$

  where:
  - $G_t \in R^{dxd}$: is a diagonal matrix where every component on the diagonal $(i,i)$ is the sum of squared of the gradients from instant *0* to *t*
  - $\varepsilon$: $1e^{-7}$
  - $\lambda$ (0.01): is the learning rate
  - Decay: 0.0
- **Rmsprop:** it is an extension of *Adagrad* that tries to solve the problem of the learning rate that tends to 0 by accumulating the squares of the past restricted gradients within a window $w$. However, instead of storing all squared gradients of the last $w$ instants (which would be inefficient), it uses the average of the squared gradients at the instant t. where:
  - Learning rate: 0.001
  - $\rho$: 0.9
  - $\varepsilon$: $1^{-7}$
  - Decay: 0.0

Finally, in order to conduct an analysis of the real performance of the network architecture a test was conducted so as to identify the economic gain in terms of dow points. Each test was conducted by aggregating the opening prices of several days before the target date, as described in 3.2, with the goal to predict the closing price of last day in the aggregate list. The results were then compared to those obtained by randomly extracting each action between *buy* and *sell*.

Testing was performed according to the following scheme starting from equation 5:

- Only one action can be possessed at a given time. If the network have chosen to buy and does not have any previous stock then *buy*
  - The purchase cost is deducted from the capital
  - The difference between the opening and closing price is added to the *capital*
- If the network has chosen to buy but has already bought a stock, then *keep* the stock
  - The difference between the opening and closing price is added to the *capital*
- If the network has chosen to sell and owns a stock then *sell*.
  - The opening price is added to the *capital*
- Otherwise no action is taken

Where *capital* is equal to *50000 dow points* and represents the amount of money initially available to the recurrent neural network. The resulting economic gain is then calculated as:

$$Gain = \frac{final\ capital}{initial\ capital} \tag{7}$$

The transaction costs are considered later.

# 4 EXPERIMENT SETTINGS AND RESULTS

## 4.1 Results with a Positive Trend Time Series

The table 1 illustrates the average accuracies, achieved by the architecture depicted in Figure 4, obtained on 6 different runs for each test. These accuracies correspond to 81 different learning models trained according to the combination of the number of input days, of the types of mentioned above network, of the number of epochs and the kind of the algorithm optimisation. It is possible to infer from the table that *small* and *medium* networks do not perform as well as the *big* ones and that the algorithms *adagrad* and *momentum* are more suitable for the type of task considered. Furthermore 2 input days on average perform better than a 5 and 10 days.

The hardware configuration on which the solution have been implemented and experimented consists of 1 GPU NVIDIA Titan Xp 12 GB, 16 cores Xeon E5-2609 CPU and 32 GB RAM. The workstation is equipped with Ubuntu 16.04.4 LTS, Python 3.5.2, keras 2.1.5, tensorflow 1.6.0 and NVIDIA CUDA 9.0.176. With this configuration, each learning model took on average about 25 minutes to complete the training phase and about 1 minute to complete the testing phase. The training phase has regarded the period between 01/01/2000 and 30/06/2009 and the test phase has occurred in the period between 01/07/2009 and 31/12/2017, where each phase contains 3285 time instant values.

The results of the economic gain achieved by the learning model with the accuracy of 83%* - which corresponds in Table 1 to the bold value in the row 2 days and column Momentum 200 epochs - are shown in Figure 5, where the $x$ axis represents the gain with respect the initial capital and the $y$ axis the number of days of input data supplied to the neural network before the current day in which predicting the closing price. To profitably increase the gain, the neural network was also lawfully trained during the testing phase using only the opening and closing prices prior to the prediction day.

The boxplot in Figure 5 reports the normalized gain:

$$\begin{cases} profit, & gain > 1. \\ loss, & gain < 1. \end{cases}$$

Unit gain is the discriminating value that determines whether the model is able to create profit.

From Figure 5 it is possible to notice how, in the case of predictions based on random choice, the median is below the unit value most of the time. This indicates that random prediction always leads to a loss. On the contrary, basing the choice using model 4 leads to a gain that is *5.09 times* the initial capital.

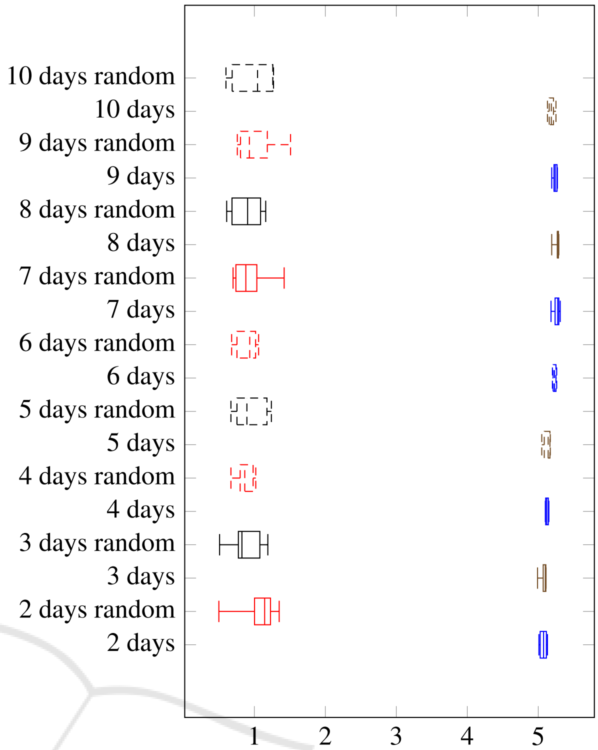


Figure 5: Economic gains, in terms of dow points, calculated according to the equation 7 according to the test between 01/01/2000 and 31/12/2017 with 200 training epochs, with optimisation algorithm Momentum and with the neural network composed by 512 and 256 neurons in the level 1 and 2 respectively, see Table 1.

In this last case, it is evident how the variability is considerably lower than the random based approach. Obviously the difference between percentile 25 and percentile 75 is much inferior than the same difference when the actions to be performed are randomly chosen. This leads us to conclude that the forecast is much more stable and reliable if carried out with the illustrated model.

The gain obtained by our LSTM learning model with the accuracy of 83% (highlighted with an asterisk in Table 1), has outperformed the gains achieved with neural networks applied to DJIA (O'Connor and Madden, 2006). The authors in (O'Connor and Madden, 2006) have applied a FF model to a test set interval between the years 1987 and 2002, obtaining a maximum ROI (*Return of Investment*) of 23.42% per year, excluding transaction fees. The data inputs their provided to the model are the opening price of the dow jones and other external indicators, including the

Table 1: Accuracies of the implemented neural network architecture, whose unit is depicted in Figure 4, in response to changes in the number of input days, training epochs, network sizes and types of optimisation algorithms. The learning model we used in the trading experiments is the one corresponding to **83%***.

| Test set accuracies between 01/07/2009 and 31/12/2017 | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Adagrad | | | Momentum | | | Rmsprop | | |
| Epochs | 50 | 100 | 200 | 50 | 100 | 200 | 50 | 100 | 200 |
| 2 days | s: 39%<br>m: 41%<br>**l: 84%** | s: 40%<br>m: 81%<br>l: 80% | s:64%<br>**m: 79%**<br>l: 47% | s: 69%<br>m: 78%<br>l: 70% | s: 71%<br>m: 74%<br>l: 70% | s: 72%<br>m: 80%<br>***l: 83%**** | s: 72%<br>m: 71%<br>l: 68% | s: 68%<br>m: 61%<br>**l: 79%** | **s: 70%**<br>m: 69%<br>**l: 70%** |
| 5 days | s: 52%<br>m: 74%<br>l: 69% | s: 79%<br>m: 59%<br>l: 80% | s: 66%<br>m: 71%<br>**l: 79%** | s: 68%<br>m: 68%<br>l: 71% | s: 68%<br>m: 74%<br>l: 74% | s: 71%<br>m: 78%<br>**l: 83%** | s: 55%<br>**m: 73%**<br>l: 67% | s: 51%<br>m: 70%<br>l: 68% | s: 68%<br>m: 68%<br>l: 66% |
| 10 days | s: 68%<br>m: 70%<br>l: 80% | s: 68%<br>m: 78%<br>**l: 86%** | s: 65%<br>m: 54%<br>l: 45% | s: 69%<br>m: 71%<br>**l: 80%** | s: 73%<br>m: 73%<br>l: **79%** | s: 71%<br>m: 74%<br>l: 75% | s: 68%<br>m: 68%<br>l: 68% | s: 64%<br>m: 68%<br>l: **79%** | s: 66%<br>m: 68%<br>l: 64% |
| s: 32 neurons in level 1, 16 in level 2<br>m: 128 neurons in level 1, 64 in level 2<br>l: 512 neurons in level 1, 256 in level 2 | | | | | | | | | |

opening price of the previous 5 days, the 5-day gradient and the USD/YEN, USD/GBP, USD/CAN conversion rates. Our LSTM-based model has achieved on average a ROI of 56.5% per year, which produces a cumulative profit of 254,000 dow points from an initial capital of 50000 dow points, that is more than 5 times the initial capital, which is also larger than the 18.48 ROI achieved with the same LSTM technology in (Bao et al., 2017).

Table 2 compares the performances obtained with our selected LSTM-based model and the one based on FF networks considering the same test procedure used in 1. The results show that LSTM networks achieve a higher accuracy which is then translated into a greater cumulative profit.

Table 2: Performance, obtained on 6 runs, comparison between the LSTM based model and the one based on FF networks with $agg = 2$, with 200 epochs of training and Momentum as gradient optimisation algorithm, with (S)mall, (M)edium and (L)arge networks' sizes.

| Test set accuracies comparison | | | | | |
|---|---|---|---|---|---|
| Feed-Forward | | | LSTM | | |
| S | M | L | S | M | L |
| 63% | 65% | 68% | 72% | 80% | 83% |
| Cumulative Profits (times initial capital) | | | | | |
| Feed-Forward | | | LSTM | | |
| S | M | L | S | M | L |
| 3.47 | 3.59 | 3.56 | 4.03 | 4.33 | 5.09 |

The graph (C) of Figure 6 compares the closing prices predicted by the model, trained for 200 epochs with $agg = 2$ and Momentum as gradient optimisation algorithm, with real prices. The average error committed, by regression, by the model, *in the entire period*, is 113.09 dow points and the relative $\sigma$ is 157.92 dow point where $\sigma$ of close prices in test set is 4140.74. This denotes the high capacity of deep networks (in particular LSTM) to extract meaningful information from a noisy context.

The great ability to forecast closing prices translates into a good ability to determine the action to be taken, according with equation 5, in order to maximize the economic return. The graph (A) of figure 6 illustrates the cumulative profit trend during the test period between 01/07/2009 and 31/12/2017 both gross and net of fees considering 2 days of aggregation, 200 epochs of training and Momentum as optimization algorithm. In the latter case we have assumed a fixed commission of 15‰ obtaining 251336.21 dow points or 1.35% less than the case without commissions which is *254786.55*.

The graph (B) of Figure shows the trend of daily profit with the inverted DJIA time series, which, as reported in the graph (D), has a negative trend on average. In the latter case, the profit derives from the daily volatility of the stock prices. It is also interesting to highlight, as in correspondence with the greater drop in the value of the stock prices, the network has decided to 'keep' the stocks (not buy and not sell). Here the network has extracted some form of pattern that led it to decide to minimize losses by assuming a neutral attitude.

Labelling as *False positive* and *False negative* respectively the actions in which the network bought and sold erroneously, the calculated *f1 measure* is 0.82 supporting the hypothesis that the model is able to catch patters in the data and with which performing reliable predictions.

Finally, the minimum capital needed was calculated so that the capital would never be negative du-

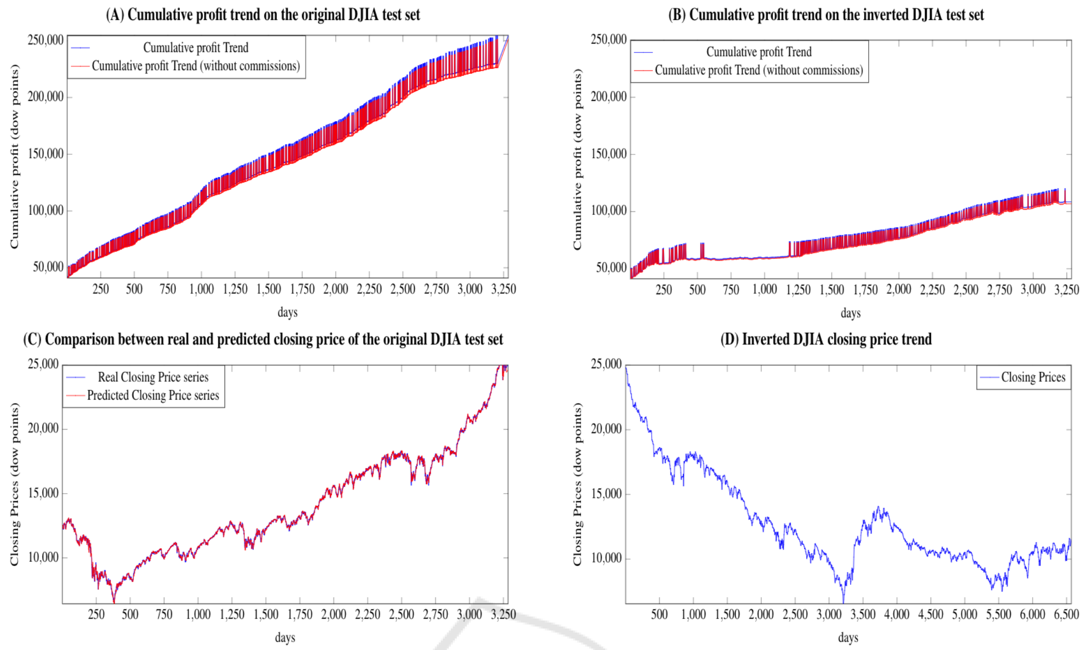


Figure 6: **(A)** Daily profit trend with and without commissions (fixed at 15 ‰) for the DJIA between the 01/07/2009 and 31/12/2017 with an initial capital of 50000 dow points
**(B)** Daily profit trend with and without commissions (fixed at 15 ‰) for the inverted DJIA time series between the 30/06/2009 and 01/01/2000 with an initial capital of 50000 dow points; the training phase is between 31/12/2017 and 01/07/2009
**(C)** Comparison between real and predicted closing price as reported in Table 1 for the DJIA with $agg = 2$, with 200 epochs of training and Momentum as gradient optimisation algorithm.
**(D)** Close price trend of the inverted DJIA time series between the 31/12/2017 and 01/01/2000.

ring the entire forecast period. Of all the experiments conducted, the minimum capital required in the worst case was 11878.28 dow points.

## 4.2 Results with a Negative Trend Time Series

The previous section illustrated the results of our RNN achieved on the DJIA time series between the year 2000 and the year 2017. Observing the time series it is possible to notice that its trend is on average positive, which might concealing the real predictive capabilities of the model.

To obtain a counter-test of the results obtained in 4.1, we have tested the same learning model used in the previous subsection, on the same but inverted time series, that is a negative trend on average, where the first day of the series is the 31/12/2017 and the last day is the 01/01/2000.

Then we have reran the most significant tests considering also the random counterparts. In particular, the tests were conducted on the large network with 512 neurons in the first hidden layer and 256 in the second hidden layer, moreover it has been trained with 200 epochs from the aggregation of the opening prices of 7 and 8 days prior to the target day. The test policies are the same defined in 4.1 and the final gain is computed according to the equation 7. The gains achieved, prior to commission fees, are the following: with 7 days is 2.23 versus 0.74 of the random approach, which corresponds to a loss of 26%, and with 8 days is 2.12 against 0.75 of the random approach, which amount to a loss of 25%.

This experiment shows that the gain is halved compared to that obtained with the original DJIA historical series, but it still leads to a profit of more than twice the initial capital with both cases of 7 and 8 days. The gain obtained through a model operating random choices is about 25% lower than that obtained on the original DJIA time series, however, leading to losing part of the initial capital.

This last test is particularly relevant as it clearly shows how the proposed solution obtains significant gains even in the presence of a time series with a negative averagely trend. Moreover from the graph (B) of Figure 6, we can see that the selected learning model almost never performs trading actions in test interval between the day number 600 and 1200, which corresponds in the graph (D) to the interval between the day number 3885 (i.e. 3285+600) and 4485

(3285+1200), that is a long consecutive test period of decreasing of the inverted DJIA. This highlights that the solution has also learned when it is better to refrain from trading.

## 5 CONCLUSION

In this work, we have analysed the predictive capabilities on the DJIA index of a simple solution based on novel deep recurrent neural networks, which in several research areas have shown superior capabilities of detecting long term dependencies in sequences of data, such as in speech recognition and in text understanding. The aim of our work was to move away from the latest complex trends, in terms of stock market prediction based on the use of non-structured data (tweets, financial news, etc.), in order to focus more simply on the stock time series. From this viewpoint the work follows the philosophy of the ARIMA approach proposed in 1970 by Box and Jenkins but experimenting advanced approaches.

Our tests have shown that the proposed solution is able to obtain a prediction accuracy of about 83% and a profit of more than 5 times the initial capital, outperforming the state-of-the-art. The tests have also shown how the predictions benefit from a lower variability compared to that obtained with an approach operating random choices.

The work can be extended to a scenario of parallel trading with multiple stocks, also investigating and exploiting possible correlations among different market indexes. Another possible improvement is to predict the best trading action according to forecasts of stock price movements referred to two or more days in the future. This strategy may lead to the identification of additional recurring patterns able to exploit the time lag in the reactivity of stock market as supposed in (LeBaron et al., 1999).

## REFERENCES

Abe, M. and Nakayama, H. (2018). Deep learning for forecasting stock returns in the cross-section. *arXiv preprint arXiv:1801.01777*.

Akita, R., Yoshihara, A., Matsubara, T., and Uehara, K. (2016). Deep learning for stock prediction using numerical and textual information. In *Computer and Information Science (ICIS), 2016 IEEE/ACIS 15th International Conference on*, pages 1–6. IEEE.

Atsalakis, G. S. and Valavanis, K. P. (2009a). Forecasting stock market short-term trends using a neuro-fuzzy based methodology. *Expert systems with Applications*, 36(7):10696–10707.

Atsalakis, G. S. and Valavanis, K. P. (2009b). Surveying stock market forecasting techniques–part ii: Soft computing methods. *Expert Systems with Applications*, 36(3):5932–5941.

Bao, W., Yue, J., and Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLOS ONE*, 12(7):1–24.

Bollen, J., Mao, H., and Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of computational science*, 2(1):1–8.

Box, G. E. P. and Jenkins, G. (1970). *Time Series Analysis, Forecasting and Control*. Holden-Day, Inc., San Francisco, CA, USA.

Cao, Q., Leggio, K. B., and Schniederjans, M. J. (2005). A comparison between fama and french's model and artificial neural networks in predicting the chinese stock market. *Computers & Operations Research*, 32(10):2499–2512.

Cho, K., van Merriënboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

di Lena, P., Domeniconi, G., Margara, L., and Moro, G. (2015). GOTA: GO term annotation of biomedical literature. *BMC Bioinformatics*, 16:346:1–346:13.

Ding, X., Zhang, Y., Liu, T., and Duan, J. (2015). Deep learning for event-driven stock prediction. In *Ijcai*, pages 2327–2333.

Domeniconi, G., Masseroli, M., Moro, G., and Pinoli, P. (2016). Cross-organism learning method to discover new gene functionalities. *Computer Methods and Programs in Biomedicine*, 126:20–34.

Domeniconi, G., Moro, G., Pagliarani, A., and Pasolini, R. (2017a). Learning to predict the stock market dow jones index detecting and mining relevant tweets.

Domeniconi, G., Moro, G., Pagliarani, A., and Pasolini, R. (2017b). On deep learning in cross-domain sentiment classification. In *Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - (Volume 1), Funchal, Madeira, Portugal, November 1-3, 2017.*, pages 50–60.

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., and Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118.

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, 25(2):383–417.

Fischer, T. and Krauss, C. (2017). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*.

Fisher, I. E., Garnsey, M. R., and Hughes, M. E. (2016). Natural language processing in accounting, auditing and finance: A synthesis of the literature with a roadmap for future research. *Intelligent Systems in Accounting, Finance and Management*, 23(3):157–214.

Gers, F. A., Schmidhuber, J., and Cummins, F. (1999). Learning to forget: Continual prediction with lstm. *Neural Computation*, 12:2451–2471.

Gidofalvi, G. and Elkan, C. (2001). Using news articles to predict stock price movements. *Department of Computer Science and Engineering, University of California, San Diego*.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Khashei, M., Bijari, M., and Ardali, G. A. R. (2009). Improvement of auto-regressive integrated moving average models using fuzzy logic and artificial neural networks (anns). *Neurocomputing*, 72(4-6):956–967.

Khashei, M., Bijari, M., and Ardali, G. A. R. (2012). Hybridization of autoregressive integrated moving average (arima) with probabilistic neural networks (pnns). *Computers & Industrial Engineering*, 63(1):37–45.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA. Curran Associates Inc.

LeBaron, B., Arthur, W. B., and Palmer, R. (1999). Time series properties of an artificial stock market. *Journal of Economic Dynamics and control*, 23(9-10):1487–1516.

Lee, C.-M. and Ko, C.-N. (2011). Short-term load forecasting using lifting scheme and arima models. *Expert Systems with Applications*, 38(5):5902–5911.

Lee, K., Yoo, S., and Jongdae Jin, J. (2007). Neural network model vs. sarima model in forecasting korean stock price index (kospi). 8.

Lin, M.-C., Lee, A. J., Kao, R.-T., and Chen, K.-T. (2011). Stock price movement prediction using representative prototypes of financial reports. *ACM Transactions on Management Information Systems (TMIS)*, 2(3):19.

Lo, A. W. and MacKinlay, A. C. (1988). Stock market prices do not follow random walks: Evidence from a simple specification test. *The Review of Financial Studies*, 1(1):41–66.

Lo, A. W. and Repin, D. V. (2002). The psychophysiology of real-time financial risk processing. *Journal of cognitive neuroscience*, 14(3):323–339.

Malkiel, B. G. (1973). *A Random Walk Down Wall Street*. Norton, New York.

Malkiel, B. G. (2003). The efficient market hypothesis and its critics. *Journal of economic perspectives*, 17(1):59–82.

Merh, N., Saxena, V. P., and Pardasani, K. R. (2010). A comparison between hybrid approaches of ann and arima for indian stock trend forecasting. *Business Intelligence Journal*, 3(2):23–43.

Mitra, S. K. (2009). Optimal combination of trading rules using neural networks. *International business research*, 2(1):86.

Mostafa, M. M. (2010). Forecasting stock exchange movements using neural networks: Empirical evidence from kuwait. *Expert Systems with Applications*, 37(9):6302–6309.

O'Connor, N. and Madden, M. G. (2006). A neural network approach to predicting stock exchange movements using external factors. *Knowl.-Based Syst.*, 19(5):371–378.

Olson, D. and Mossman, C. (2003). Neural network forecasts of canadian stock returns using accounting ratios. *International Journal of Forecasting*, 19(3):453–465.

Schumaker, R. and Chen, H. (2006). Textual analysis of stock market prediction using financial news articles. *AMCIS 2006 Proceedings*, page 185.

Schumaker, R. P. and Chen, H. (2009). Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Transactions on Information Systems (TOIS)*, 27(2):12.

Soni, S. (2011). Applications of anns in stock market prediction: a survey. *International Journal of Computer Science & Engineering Technology*, 2(3):71–83.

Sterba, J. and Hilovska, K. (2010). The implementation of hybrid arima neural network prediction model for aggregate water consumption prediction. *AplimatJournal of Applied Mathematics*, 3(3):123–131.

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

Wang, W., Li, Y., Huang, Y., Liu, H., and Zhang, T. (2017). A method for identifying the mood states of social network users based on cyber psychometrics. *Future Internet*, 9(2):22.

Wikipedia contributors (2018). Financial statement — Wikipedia, the free encyclopedia. [Online; accessed 20-May-2018].

Yahoo! FinanceYahoo! Finance (2000-2017). Dow jones industrial average (djia). http://bit.ly/2rozaW8.