TÉCNICO LISBOA

# Sentiment Analysis for Financial Data Prediction

**Frederico Pascoal Gaião Martins Monteiro**

Thesis to obtain the Master of Science Degree in

## Information Systems and Computer Engineering

Supervisor: Prof. Diogo Manuel Ribeiro Ferreira

### Examination Committee

Chairperson: Prof. Alberto Manuel Rodrigues da Silva
Supervisor: Prof. Diogo Manuel Ribeiro Ferreira
Member of the Committee: Prof. Bruno Emanuel Da Graça Martins

**November 2020**

Dedicated to both my grandfathers.

Fernando and José, you will be forever missed.

# Acknowledgments

I would like to start by thanking my M.Sc. Thesis supervisor. I want express my gratitude for the constant availability to share knowledge and clarify questions about my research or writing. Professor Diogo R. Ferreira allowed this dissertation to be my own work, but guided me in a direction he thought I needed. Without Prof. Diogo's help this dissertation would have never reached good terms.

I would also like to credit my parents and sister. Their patience when returning home from long days of work was immeasurable. They encourage me to pursue my dreams and support my decisions.

In addition, I would like to thank my friend Luís by always giving me the right advice at the right time. To my friend Diogo, a person ready to give his input whenever I was facing a mental block. Likewise, a big thank you to José and Henrique. Two colleagues that became friends throughout this process.

Last but not least, Amalia. You picked me up when I was down. You forced me to rest and stop when needed. You were my rock throughout this project. Sincerely, thank you.

To all the aforementioned and, to other friends and colleagues who were there for me, either for work or for distractions, my sincere gratitude. My journey at Instituto Superior Técnico would not have been the same without you.

# Resumo

Os mercados financeiros, como a bolsa de valores, são extremamente voláteis e sensíveis às notícias publicadas nos meios de comunicação. Através da análise do sentimento dessas notícias, para além da análise de séries temporais de cotações, deverá ser possível fazer uma melhor previsão do comportamento futuro de um determinado ativo financeiro. O principal objetivo deste trabalho é quantificar o benefício que pode ser obtido através da análise de sentimento, quando comparada com a análise de séries temporais apenas, para prever o movimento futuro de ações em bolsa. Para atingir esse objetivo, a abordagem proposta utiliza vários modelos diferentes de aprendizagem profunda. Começamos por utilizar modelos simples que dependem apenas de indicadores dos mercados financeiros, passando em seguida para a utilização de redes recorrentes que incorporam o sentimento de notícias relacionadas com ativos financeiros. Surpreendentemente, os resultados obtidos sugerem que o benefício da análise de sentimento é diminuto. No entanto, é possível obter melhores resultados com modelos de aprendizagem mais sofisticados, nomeadamente utilizando mecanismos de atenção.

**Palavras-chave:** Previsão de Ações, Análise de Sentimento, Aprendizagem Profunda, Atenção

# Abstract

Financial markets, such as the stock exchange, are known to be extremely volatile and sensitive to news published in the media. Using sentiment analysis, as opposed to using time series alone, should provide a better indication for the prospects of a given financial asset. In this work, the main goal is to quantify the benefit that can be obtained by adding sentiment analysis to predict the up or down movement of stock returns. The approach makes use of several different deep learning models, from vanilla models that rely on market indicators only, to recurrent networks that incorporate news sentiment as well. Surprisingly, the results suggest that the added benefit of sentiment analysis is diminute, and a more significant improvement can be obtained by using sophisticated models with advanced learning mechanisms such as attention.



**Keywords:** Stock Prediction, Sentiment Analysis, Deep Learning, Attention

# Contents

# List of Tables

# List of Figures

# List of Acronyms

| | |
|---|---|
| **AF** | Activation Function |
| **AI** | Artificial Intelligence |
| **AR** | Autoregressive |
| **ARCH** | Autoregressive Conditional Heteroscedasticity |
| **ARIMA** | Autoregressive Integrated Moving-Average |
| **ARMA** | Autoregressive Moving-Average |
| **AUC** | Area Under the ROC Curve |
| **BOW** | Bag-Of-Words |
| **BP** | Back-Propagation |
| **BPTT** | Back-Propagation Through Time |
| **CNN** | Convolutional Neural Network |
| **DL** | Deep Learning |
| **EMH** | Efficient-Market Hypothesis |
| **EWT** | Elliott Wave Theory |
| **FA** | Fundamental Analysis |
| **FNN** | Feed-Forward Neural Network |
| **GARCH** | Generalized Autoregressive Conditional Heteroscedasticity |
| **GRU** | Gated Recurrent Unit |
| **HL** | Hidden Layer |
| **LSTM** | Long Short-Term Memory |
| **MA** | Moving-Average |
| **MCC** | Matthews Correlation Coefficient |
| **ML** | Machine Learning |
| **NLP** | Natural Language Processing |
| **NN** | Neural Network |
| **NPV** | Negative Predictive Value |
| **PPV** | Positive Predictive Value |
| **RL** | Reinforcement Learning |
| **RNN** | Recurrent Neural Network |
| **RWT** | Random Walk Theory |

| | |
|---|---|
| **SA** | Sentiment Analysis |
| **SL** | Supervised Learning |
| **SP** | Stock Prediction |
| **SVM** | Support Vector Machine |
| **TA** | Technical Analysis |
| **TS** | Time Series |
| **TSP** | Time Series Prediction |
| **UL** | Unsupervised Learning |

# Chapter 1

# Introduction

Financial markets are part of the core structure of modern society, since the stability and prosperity of these markets have a great impact on economic development. However, financial markets are known to be extremely volatile and sensitive to its surrounding environments. The ability to forecast stock price movement allows to avoid losses and grow gains, consequently preventing crisis like the ones occurred in 2009 or 2018 and increasing general prosperity. Therefore, apart from obvious benefits to investors, to accurately predict future stock behavior is a crucial activity with clear interest to protect interests of public companies and governments.

Stock Prediction (SP), i.e. the prediction of the future value of a company stock, is an extremely difficult task due to intense amount of data produced, its high degree of uncertainty and noise. As a result, the extent to which the future price of a stock can be predicted from its past history has always been the source of much debate [1].

In this work, besides predicting stock price movement based in its past history, we will be interested in checking whether such prediction can be improved based on the perceived sentiment from market news. For this purpose, besides Time Series (TS) we will also use Sentiment Analysis (SA). Both components being described in greater detail in the following Chapter 2. In the remaining of this chapter, we will concentrate on introducing Stock Prediction main theories and explain media influence on the stock market.

## 1.1 Predictability of the Stock Market

In essence, there are three main theories about the predictability of the stock market: Random Walk Theory (RWT) [1], Elliott Wave Theory (EWT) [2] and Efficient-Market Hypothesis (EMH) [3].

RWT endorses that markets are completely arbitrary and stock prices are not predictable, having random and independent movements. On the contrary, EWT is based on stock trends following a repeating pattern that can be forecast. Thus, through careful analysis of technical indicators charts, an understanding of these patterns can be developed to predict stocks future movements.

EMH advocates that stock prices reflect all information. There are three forms of EMH: weak, where only past historical stock data is integrated; semi-strong, where in addition all public information is incorporated; and strong, where private information is also considered. The latter supports that any price change not based on available information is inherently unpredictable [4].

Financial data plays a key role in all of the above theories, and its analysis is divided in two categories:

1. Technical Analysis (TA), supporting EWT, is a short-term approach used for trading. It examines peaks, bottoms and trends based on financial data to find the correct timing to enter and exit the market. It makes use of statistical techniques and visual pattern recognition methods further explored in Section 2.1. However, it is somewhat subjective in nature, since it depends on each trader interpretation of technical charts and data.

2. Fundamental Analysis (FA), supporting EMH strong form, is a long-term approach used for stable and long investments. FA tries to predict changes before they appear on charts by analyzing intrinsic value, hence harder to formalize into rules. It integrates multiple sources of distinct data such economic indicators, financial reports from banks and textual information from the media. The challenge in this form of analysis is to deal with unstructured data in a systematic manner.

It is interesting to observe how both approaches above can be mapped to two fundamentally different data sources. On one hand, there are market indicators that reflect the day-to-day evolution of stock prices and can be used for TA. On the other hand, there are other complementary sources, such as media news and sentiments, that ultimately contribute to shape the evolution of those stocks, and are important for FA. While TA can be addressed, for example, with Time Series Prediction (TSP), FA may require the use of additional data sources and techniques, e.g. Sentiment Analysis, to fully understand how the current stock price synthesizes all the information available about a company.

## 1.2 Media Influence on the Stock Market

From traders' traditional use of technical methods, to an easier accessibility and processing of financial data when comparing to textual information, the great majority of approaches always revolved around Technical Analysis. However, people use media to be informed about what is happening and what might happen. Moreover, the stock market is driven by humans, their judgment and conflicting thoughts. Therefore, analyzing media articles as the ones in Fig. 1.1 can be useful to gain a deeper insight when predicting stock price future movement.


Figure 1.1: Examples of the impact of media articles on stock prices.

As sentiment can be detected from an article [5], it is rational to expect that media articles containing negative words such as “hacked” or “cuts” precede a downward trend in the stock price. In contrast, positive terms like “merger” or “acquisition” should be followed by an upward trend [6]. In addition, a specific number of relevant entities and related events directly affect companies and their assets [7]. Based on this fact, it is logical to expect that cases as Amazon acquiring Whole Foods would positively impact Amazon.com share prices. Or negative events such Steve Jobs death and Elon Musk mocking the SEC on Twitter to impact Apple and Tesla shares in a negative way, correspondingly. In addition, analyzing Google Trends or searches on Wikipedia pages related with stock assets can capture public interests and help to better predict stocks' prices [8].

## 1.3 Main Goals of this Work

Stock market data and news articles are constantly being produced, released and in permanent change. However, human perception is limited and incapable of effectively handling all information available. As the two types of data have a unique time date, both can be considered Time Series and analyzed as such. Numerous attempts to predict stock price movement have been proposed using this approach.

Although is general belief among researchers that news sentiment will never replace financial data as primary source when doing SP [9–12], this type of data can be used as a complement. But how much can one gain by incorporating Sentiment Analysis over a pure Technical Analysis such as Time Series Prediction? In principle, SA should be able to provide some explanation for the seemingly random evolution of a Time Series, so it should be possible to achieve an increase in the accuracy of any Stock Prediction model. The question that we address in this work is how much of an improvement can one actually obtain by incorporating Sentiment Analysis.

A panoply of techniques can be used to analyze financial data and do SA. Since recently there has been an increasing focus on addressing both TSP and SA with Deep Learning (DL) models, we also use DL to facilitate the development of prediction models that combine both components.

Lastly, a state-of-the-art approach for Natural Language Processing tasks, the Attention mechanism, was developed recently. This has achieved promising results in machine translation and SA. As a result, we employ this mechanism in our models to measure its performance on financial data as well. A comparison using DL techniques with, without and solely relying on Attention will also be included. To measure the impact of Sentiment Analysis and Attention when doing Stock Prediction, the directional accuracy will be used as evaluation metric.

## 1.4 Document Structure

Following this introductory section, Chapter 2 reviews the main techniques used when doing SP, namely: Section 2.1 only with historical financial data; Section 2.2 introduces the field of Sentiment Analysis; and Section 2.3 combines financial data and media Sentiment Analysis for Stock Prediction.

Chapter 3 details the application of Deep Learning to Stock Prediction, namely: Section 3.1 introduces the field of Deep Learning, its origins and components for a better comprehension of this work; Section 3.2 analyses most known Neural Network models; Section 3.3 details Attention mechanisms and how they can be incorporated into Stock Prediction.

Chapter 4 describes the experimental approach to SP with and without SA, namely: Section 4.1 where we detail the origin of the data; Section 4.2 where the features used are described; Section 4.3 details the preprocessing done; Section 4.4 explains the input layers for each model; Section 4.5 characterizes the different models employed; and Section 4.6 specifies how the results are going to be evaluated and which metrics are used.

Chapter 5 describes the results obtained by NN models introduced throughout this document, having a section for each NN with its best architecture. In the end of the chapter, Section 5.9 summarizes all results and draw more broader conclusions.

Chapter 6 concludes the document with our major contributions, providing some final remarks about potential uses of our approach and some of its limitations.

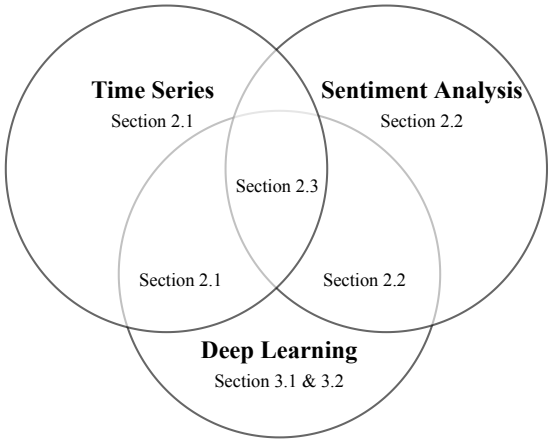



Figure 1.2: Outline of work areas of study.

In summary, the main areas of study involved in this work are represented schematically in Fig. 1.2. It is important to notice that Deep Learning has brought the possibility of uniting financial data as Time Series and Sentiment Analysis to develop better models for Stock Prediction as we will see in the next chapter.

# Chapter 2

# Components of Stock Prediction

Based on the Efficient-Market Hypothesis (EMH), a possible definition of successful Stock Prediction (SP) is to achieve the highest accuracy while using the smallest amount of information possible. The goal being to reduce investors' risk while maximizing profits [13].

Numerous attempts have been made to predict the movement of stock prices. Since stocks are in a continuous shaping process due to new information [14], their associated data is usually complex, uncertain, incomplete, and vague [15]. SP is acknowledged to be one of the most challenging Time Series (TS) tasks [16], with the hardest part being to select relevant features and learning mechanisms.

In this chapter, we have a look at how other authors have approached Stock Prediction from two different perspectives, namely from the perspective of Time Series Prediction (TSP) and from the perspective of Sentiment Analysis (SA), which are the most relevant components for our purpose. Rather than an in-depth study of each technique, the intention is to reflect about their purpose and gain insights regarding their performance. In Section 2.1, models only using financial data and their challenges are detailed. Section 2.2 introduces SA, a field of Natural Language Processing (NLP) that analyses investors sentiment and published media. Finally, Section 2.3 highlights how both historical data and SA are incorporated to achieve state-of-the-art accuracy. Later on Chapter 3, Deep Learning (DL) techniques and their application to Stock Prediction will be described in detail.

## 2.1 Time Series Prediction

Any problem where order of data captured is registered and taken into account can be considered a TSP problem [17]. Therefore, Stock Prediction is a TSP problem composed by several indicators.

Consequence of the stock market being affected by a myriad of factors, financial data is comprised by a great set of unstructured nature features causing complex behavior with hidden relationships between different stocks. These circumstances cause a constant evolution, leading to highly noisy and chaotic data that has neither stationary (mean, variance, and frequency change over time) or linear properties [18]. As a result, Stock Prediction is acknowledged as one of the most challenging TSP tasks [16], numerous models based on Statistical, Machine Learning and Reinforcement Learning techniques have

been proposed for financial data prediction [19]. Below each technique will be briefly described.

- **Statistical Techniques** such as AR [20], MA, ARMA and ARIMA models make the assumption that stock data has linear properties. Additionally, all except ARIMA, assume that variables also have stationary properties [21]. By employing moving averages, these models are sufficient to describe a trend, hence recognizing patterns [22, 23]. As a drawback, when in market reversions, significant losses are suffered [24]. In addition, only a stock is predicted.

  Moreover, aforementioned models assume homoscedasticity (constant variance of the error term over time). Although, as a consequence of assets volatility from their dynamic variables relationship, financial data is heteroscedasticity [25, 26]. Advanced models such as ARCH [27] and GARCH [28] already consider that error variance.

  To mitigate the uni-variable constraint above mentioned, alternative models like VAR were proposed. These models are able to cope with multi-variable data however, making presumptions about data distribution as well. Therefore, likewise resulting in an inability to perceive underlying non-linear behavior or existent dynamic between stocks [29]. Consequently, these models adoption on complex real-world problems produce unsatisfactory predictions [20].

  From economic decisions to political events, every aspect influences stocks fluctuation and thereby its corresponding valuation or devaluation [30]. Based on this information, no model aforementioned can yield significant forecasting accuracy with such disparate data in such noisy environments. All models discussed within statistical techniques, derived from a poor generalization ability, underperform when used for live forecasting.

- **Machine Learning (ML) Techniques** adapt well to noisy environments having great tolerance to imprecision. These techniques are able to forecast multiple stocks at the same time, consequently allowing them to capture non-linear relations and to make statistical assumptions about training data with partially unknown parameters [31–33].

  - *Support Vector Machine (SVM)* is a common approach when dealing with TSP [19, 34, 35]. Through its risk minimization principle, better generalization than previous models is reached even outperforming some Neural Networks [36–38] .

    Nonetheless, a disadvantage is the large amount of computation time required when solving large problems. A dimensionality reduction, through preprocessing and clustering, is required in order to expedite the process and diminish the overfitting risk [39, 40].

  - *Neural Network (NN)* learns from examples, understanding underlying patterns in them [41]. In addition, it can generalize from past experience and predict on data that might never been previously observed, without *a priori* assumptions about its properties [42].

    NN adapts non-linear functions to training data [43]. This contributes for NN to have the highest accuracy, outperforming statistical techniques by at least 10% [42, 44, 45] and other traditional ML models such as SVM and Random Forests [46, 47].

Noise tolerance and the ability to handle incomplete or corrupted data are also advantages when comparing to previous models. Aforementioned factors, the ability to generalize its prediction to any asset and the relative ease when dealing with complex data jointly justify why NN are chosen for financial data prediction [48]. Deeper NN outperform more simpler ones [49–51].

- *Convolutional Neural Network (CNN)*, through convolutions in the form of a sliding window over its input, are capable of capturing variable correlations, hence discovering hidden features and understanding non-linear relations. CNNs are very popular in image processing where the input is 2D, but they are also very successful at 1D input processing such as text sentences [52] and Time Series [53]. Additionally, they can also be used in 3D inputs.

  Recently, several models based on the WaveNet [54] have been proposed. These often perform better than DL models with memory [55]. This is due to a CNN analysis be only based on the current window. Therefore, when applied to financial data, this model better captures sudden changes, hence obtaining its best results on high volatility markets [56–58]. CNN models are able to follow trends, however, cannot predict stock price [59].

  When dealing within the Time Series tasks scope, ref. [53] argues that CNN is the most widely applied model due to its robustness and relatively small amount of training time when compared to Recurrent Neural Network.

- *Recurrent Neural Network (RNN)* are very appropriate for sequence processing [60], since they keep an internal state and have a feedback loop to use that internal state as an additional input at each time step. Based on this assumption, it is possible to reuse historical stock information from the past to predict the future. [21, 56, 58, 61, 62]. However, simple RNNs are unable to perform well when a long-term context is required [63].

- *Long Short-Term Memory (LSTM)* is a type of RNN with the ability to capture past data influence over the course of time. The LSTM architecture [64] improves on long-term dependencies by keeping a cell state with the possibility of carrying past information across multiple time steps. LSTM, due to their longer memory, improved results and allow to outperform memory-free models [65–67]. From all discussed models, LSTM is presenting the most promising results for Stock Prediction.

  Recently, slightly different recurrent versions as the GRU model have been proposed [68]. However, LSTM models can still provide state-of-the-art results [69]. Nonetheless, in ref. [10] it is hypothesized that the inclusion of Attention mechanisms can capture longer-term dependencies and even outperform stand-alone LSTM.

- **Reinforcement Learning (RL) Techniques** train an agent through trial and error in order to maximize a reward [62]. With this strategy, Technical Analysis (TA) noisy and uncertain environments can be overcome [70]. The agent automates trades and diminishes costs by classifying and systematizing TA as a set of rules to anticipate future price shifts [71].

Most studies use the Sharpe Ratio [72, 73] or the Calmar Ratio [74] as objective function and a measure of performance. Trading frequency can also be increased to lower the agent's performance degradation [73]. Quality-Learning algorithm (QLa), by not having a limited number of states [72], can also be a solution to better cope with confused markets where stocks' value may assume an infinite number of values [75].

However, using RL techniques for Stock Prediction has challenges. Due to high volatility and noise, common in financial data, patterns may be unclear in training data and significant differences may be observed between training and test dataset. Both factors cause the agent to underperform. Another difficulty, in order to maximize profit, is to determine an optimal number of shares to buy/sell instead of trading a fixed number, which is a tremendous task for the agent to execute.

The paragraphs above summarize the main techniques used for Stock Prediction. In order to have a broader overview of these works, Table 2.1 and Table 2.2 present the literature on TSP in an alternative format.

Table 2.1 presents a summary of the methods employed, the stock indexes and the data sources used by different authors. This survey leads us to conclude that *Yahoo Finance* and *Thomson Reuters* have been the most popular data sources; other sources used are *Bloomberg*, *Google Finance* and public datasets. 25% of the articles did not reveal its origin. It is our assumption, due to the time range periods analyzed, that their origin must be similar to aforementioned sources. On the other hand, the most common indexes and markets are the *American* (*S&P*, *NASDAQ*, *DJIA* and *NYSE*) and the *Asian* (*NSE*, *Nikkei*, *KOSPI* and *CSI*); others include *EuroStoxx50* and *Ibovespa*.

Table 2.1: Techniques and data sources for stock prediction based on time series.

| Technique | Index | Yahoo Finance | Thomson Reuters | Other | Unidentified |
|---|---|---|---|---|---|
| **Stats** | American | [76–79] | [80, 81] | | [82] |
| | Asian | | [81, 83] | | [21, 42, 58, 66] |
| **ML** | American | [4, 7, 8, 61, 79, 84, 85] | [47, 80, 81] | [46, 86, 87] | [56, 59, 82] |
| | Asian | [85, 88] | [81, 83] | [89] | [21, 56, 69, 90, 91] |
| | Other | [85] | | [34] | |
| **NNs** | American | [7, 8, 51, 61, 79, 92, 93] | [43, 47, 67, 80] | [10, 46, 94] | [56, 59, 82] |
| | Asian | [51] | [67] | [89] | [21, 42, 56, 58, 66, 69, 90, 95] |
| **RL** | American | [62, 74, 75] | [75] | | |
| | Asian | [75] | [75] | | |
| | Other | [75] | [75] | | |

From Table 2.2 it is possible to conclude that regarding the time frame for training and evaluation, the shortest period found in these works was one month [4] and the longest was 66 years [59]. It is worth noting that statistical techniques tend to use shorter time frames to avoid noise and dimensionality issues [76–79], while ML techniques tend to use longer time frames to properly train the models [47, 59, 67, 80]. Also, the directional movement of stocks (up or down) is the most common prediction goal, and arguably the most important one [86]. Only two articles tried to predict the actual stock price [87, 93].

In conclusion, models employed vary frequently, corroborating that SP is a very well explored field. It is important to notice that almost every article often used ML techniques, in particular at least one NN model. The latter have been reaching remarkable results when predicting financial Time Series. Especially LSTM, the state-of-the-art approach, appearing as one of the most promising models [69].

Table 2.2: Comparison of financial data sources.

| Ref. | Algorithm | Source | Index | Time-Frame | Period | Forecast Type |
|---|---|---|---|---|---|---|
| [59] | MLP, CNN, LSTM | – | S&P 500 | Daily | 01/1950 – 12/2016 | Direction |
| [82] | GARCH, MLP, NN | – | NASDAQ | Daily | 10/2008 – 06/2009 | Direction |
| [42] | AR, NN | – | KOSPI | Daily | 01/2010 – 12/2014 | Direction |
| [69] | MLP, CNN, LSTM<br>LSTM + Attention | – | KOSPI 200 | Daily | 01/2000 – 07/2018 | Direction |
| [66] | LSTM + GARCH | – | KOSPI 200 | Daily | 01/2001 – 09/2011 | Direction |
| [21] | AR, RF, SVM, CNN<br>RNN, LSTM, MFNN | – | CSI 300 | Minutely | 12/2013 – 12/2016 | Direction |
| [90] | AR, MLP, RNN, LSTM | – | Nikkei 225 | Daily | 01/2001 – 12/2008 | Direction |
| [95] | NN | – | TWSE [1] | Daily | 01/2013 – 12/2014 | Direction |
| [91] | Random Forests | – | SGEM [2] | Daily | 01/2010 – 10/2016 | Direction |
| [58] | ARIMA<br>CNN, RNN, LSTM | – | NSE | Minutely | 07/2014 – 06/2015 | Direction |
| [56] | MLP, CNN<br>RNN, LSTM | – | NSE<br>NYSE | Daily | 01/1996 – 06/2017 | Direction |
| [6] | – | Yahoo! Finance | S&P 500 | Hourly | 10/1999 – 02/2000 | Direction |
| [76] | VAR[3] | Yahoo! Finance | S&P 100 | Daily | 11/2012 – 02/2013 | Direction |
| [77] | VAR[3] | Yahoo! Finance | S&P 100 | Daily | 11/2012 – 04/2013 | Direction |
| [4] | SVM | Yahoo! Finance | S&P 500 | Daily | 10/2005 – 11/2005 | Direction |
| [7] | SVM<br>NN | Yahoo! Finance | S&P 500 | Daily | 10/2006 – 11/2013 | Direction |
| [92] | CNN | Yahoo! Finance | S&P 500 | Daily | 10/2006 – 11/2013 | Direction |
| [61] | SVM, CNN, RNN | Yahoo! Finance | S&P 500 | Daily | 10/2006 – 11/2013 | Direction |
| [62] | RL | Yahoo! Finance | S&P 500 | Daily | 01/1990 – 09/2015 | Direction |
| [51] | EMD2FNN | Yahoo! Finance | S&P 500<br>SSEC[7]<br>NASDAQ | Daily | 01/2012 – 12/2016 | Direction |
| [84] | SVM | Yahoo! Finance | NYSE<br>NASDAQ | Daily | 07/2012 – 07/2013 | Direction |
| [8] | SVM, NN<br>Random Forest | Yahoo! Finance | NASDAQ<br>DJIA | Daily | 01/2013 – 12/2016 | Direction |
| [93] | SOFNN[5] | Yahoo! Finance | DJIA | Daily | 02/2008 – 12/2008 | Price |
| [78] | Pearson Correlation | Yahoo! Finance | DJIA | Minutely | 11/2014 – 03/2015 | Direction |
| [79] | AR, SVM, SOFNN[5] | Yahoo! Finance | DJIA | Daily | 06/2009 – 12/2009 | Direction |
| [88] | SVM | Yahoo! Finance | HKEX[6] | Daily | 01/2003 – 03/2008 | Direction |
| [74] | RL | Yahoo! Finance | IWD, IWC<br>SPY, DEM<br>CLY | Weekly | 01/2011 – 12/2015 | Direction |
| [85] | Bayesian Network | Yahoo! Finance | iBOVESPA<br>DJIA, NYSE<br>Nikkei 225 | Daily | 06/2005 – 04/2012 | Direction |
| [75] | RL | Yahoo! Finance<br>Thomson Reuters | S&P 500<br>KOSPI, HSI<br>EuroStoxx50 | Daily | 01/2001 – 12/2017 | Direction<br>#Shares |
| [43] | FNN | Thomson Reuters | S&P 500 | Hourly | 04/2011 – 04/2016 | Direction |
| [80] | AR, Random Forest<br>NN, LSTM | Thomson Reuters | S&P 500 | Daily | 12/1989 – 09/2015 | Direction |
| [67] | LSTM | Thomson Reuters | S&P 500<br>KOSPI 200 | Daily | 01/2000 – 07/2017 | Direction |
| [81] | AR<br>SVM<br>Random Forest | Thomson Reuters | S&P 500<br>FTSE 100<br>NIKKEI 225<br>TWSE, KSE | Daily | 01/1995 – 12/20016 | Direction |
| [83] | AR, SVM | Thomson Reuters | SSEC[7] | Daily | 12/2014 – 04/2016 | Direction |
| [47] | NN<br>Random Forest | Thomson Reuters | S&P 500 | Daily | 12/1992 – 10/2015 | Direction |
| [86] | Decision Tree | Bloomberg | DJIA | Daily | 04/2012 – 04/2013 | Direction |
| [34] | SVM | Bloomberg | Ibovespa | Daily | 06/2001 – 12/2016 | Direction |
| [94] | CNN<br>LSTM | Google Finance | S&P 500<br>DJIA | Daily | 01/2006 – 11/2017 | Direction |
| [87] | SVM | Google Finance | Panasonic<br>Sharp, Sony | Daily | 01/2006 – 08/2008 | Price |
| [89] | SVM, LSTM | Wind Database | CSI 300 | Daily | 01/2009 – 10/2014 | Direction |
| [46] | SVM, NN<br>Random Forests | FRED database<br>SNL | S&P 500 | Daily | 01/1996 – 12/2017 | Direction |
| [10] | LSTM | Kaggle's Two Sigma | S&P 500 | Daily | 01/2007 – 12/2007 | Direction |

[1] Taiwan Stock Exchange
[2] Shenzhen Growth Enterprise Market
[3] Vector Autoregression
[4] Support Vector Regression
[5] Self-Organizing Fuzzy Neural Network
[6] Hong Kong Stock Exchange
[7] Shanghai Stock Exchange Composite

However, there are still limitations. The major drawback is the large amount of training data required to achieve robust generalizations and satisfactory results due to other factors aside technical indicators affecting the market [14].

As the goal of this work is to improve accuracy when predicting stocks, not only technical indicators must be analyzed but, perceived investors sentiment and media that revolve around those must be analyzed as well. Therefore the next Section 2.2 will introduce SA, a subfield of NLP, and the following Section 2.3 will resume the state-of-the-art approaches combining SA and financial Time Series prediction.

## 2.2 Sentiment Analysis

Natural Language Processing focuses on developing computational techniques to provide machines the ability to process and understand human languages. These abilities are built through models that analyze, identify and classify attributes such as:

- Author: person or entity that expresses a specific opinion;
- Polarity: emotion of author's opinion;
- Subject: entity referred on author's opinion.

While SA can be regarded as a subfield of NLP, its fast growth in recent years [96] has been propelled by numerous applications that transform human-generated information (news, tweets, etc.) about any topic (companies, products, politics, etc.) into a sentiment signal (usually positive or negative).

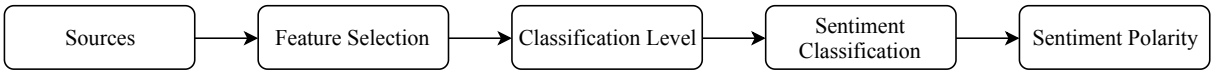



Figure 2.1: Sentiment Analysis phases.

In stock markets, participants take actions (to buy, hold or sell stocks) that are defined and affected by what they read and by what those surrounding them read and share, including the opinions of sources they trust, which are also influenced by market news. From this point of view, the emotional sentiment upon a particular stock or company has become a fundamental part of stock prediction [97, 98], and many authors have included this additional component in their prediction models. Sentiment Analysis main topics are detailed below as structured in Fig. 2.1.

- **Sources** are where authors provide opinions and from where information is retrieved. These supply new perspectives about facts while invoking sentiment within their content creating complementary information to complete SP [99].
  - *News* are published as soon as information is released or discovered. General news, headlines [7, 100] or only specific financial related news [12] can be used in SP. The latter has the advantage of avoiding noise. This type of media is the most popular source due to easy

gathering of information from trustworthy authors or newspapers [101]. However, news are very objective and impartial, a serious challenge for when attempting to determinate sentiment. In addition, news relevancy is very time dependent [98]. Typically causing traders to immediately react after its release [102].

  - *Reviews* in opposition to news, are biased therefore easier to determine sentiment. According to ref. [103], 90% of customer's purchases depends on online reviews.

  - *Social Media Platforms* allow quick spread of information, being Twitter the most common source due to a simple and structured way to retrieve data [104, 105]. However, due to character limitation, tweets are not suited for long pieces of text or fully justified opinions on a subject [106]. In addition, the common use of jargon, misspellings and very few full sentences on posts lead to this type of media requiring a lot of preprocessing when doing SA tasks.

  - *Blogs* more likely express its author's opinion than traditional media. However, there is a need to determine if the opinion is relevant regarding the targeted topic and if the author is trustworthy [107]. An additional drawback is that financial blogs tend to discuss multiple stocks or companies in a single article.

  - *Financial Reports* produced by organizations are also used (e.g. [101]).

- **Feature Selection** is a task of SA to extract, select and reduce text features. When training a model, it is important to reduce time and computation power required. Some methods used are: tokenization, stop-word removal, stemming, term presence and frequency [108].

  The most common and simplest method is to tokenize the document, use word frequency and presence to classify important features [106]. However, this approach assigns importance to terms that may not contribute to overall sentiment. A circumventing approach is Bag-Of-Words (BOW), though losing word order therefore causing inability to understand any linguistic patterns [109]. In addition, BOW does not consider any synonyms, co-references, and pronoun resolution [90]. Other possible alternatives are Noun Phrasing and Named Entities that only use certain parts of speech as features.

- **Classification Level** is essential to define at which scope level SA tasks are going to be performed. Being the main goal to determine and examine the final sentiment, classification can be done at:

  - *Document Level*, the simplest form of SA, assumes the author expresses throughout an entire document a single opinion about a single subject.

  - *Sentence Level* assumes that a document contains multiple opinions. More weight is given to local sentiment because it is easier to agree on sentence than document sentiment [110]. Only subjective sentences are analyzed being objective ones discarded. In addition, it is recommended to have specific strategies to handle different types of sentences such as sarcastic

[111], interrogative [112] and conditional [113, 114].

- *Aspect Level* is useful when a document/sentence refers to more than one entity at a time [115] and focuses on identifying its sentiment. For example, an opinion about a company is made of different aspects, such as profit, wages, conditions, etc. Aspects can be: Explicit, where all nouns are identified; or Implicit, where each explicit aspect relates with other implicit ones (e.g. a review mentioning a heavy object is implicitly referring to its weight);
- *Comparative Level* is used when the author, providing an opinion, is not direct about the entity but instead compares it with others [116].

- **Sentiment Classification** can be divided in three approaches:

  - *Lexicon-Based* approaches collect known terms with preassigned sentiment. The challenge is to adapt/customize from a particular domain to a different one [5, 117–119]. There are three main methods:

    * Manually, where lexicon is created by hand. Commonly more accurate due to careful word selection by linguistic experts but unfeasible due to colossal amount of work required.
    * Dictionary-based, where a set of seed words is expanded through its synonyms and antonyms in the dictionary (e.g. WordNet [1]). The main disadvantage being that is more generalist due to domain independent lexicon also being acquired [88, 120, 121].
    * Corpus-based, where a domain specific corpus of documents is used to expand an existent set of words. Each word sentiment polarity is detected through linguistic connectors such as “and” or “but” [122], and by using synonyms and antonyms polarity [123]. It is the only feasible and efficient choice.

    Nonetheless, some words will always need disambiguation. For example, “growth” in the financial domain is always positive, whereas in medical terms might not be. Some alternative solutions are to detect sentiment based on the co-occurrence with another word with sentiment already assigned (statistical method) or to give the same polarity to semantically close words (semantic method).

  - *Machine Learning* approaches, yielding the best accuracy, can be divided into supervised and unsupervised strategies. These strategies are described with more detail further ahead in Section 3.1.

  - *Hybrid* approaches combine merits of both previous ones. Generally built in a cascade manner so that when one classifier fails, the following tries to classify, and so on until a document is categorized.

[1] https://wordnet.princeton.edu/

Table 2.3: Comparison of text data sources.

| Ref. | Algorithm | Text Type | Source |
|---|---|---|---|
| [61] | SVM, CNN, RNN | News Headlines | Thomson Reuters |
| [94] | CNN, LSTM | News Headlines | Thomson Reuters |
| [92] | CNN | News Headlines | Thomson Reuters<br>Bloomberg |
| [10] | LSTM | News Headlines | Kaggle's Two Sigma |
| [7] | SVM, NN | News Headlines<br>News Articles | Thomson Reuters<br>Bloomberg |
| [6] | Naïve Bayes | News Articles | Yahoo! Finance |
| [88] | Lexicon Based | News Articles | FINET [13] |
| [95] | NN | News Articles | NowNews<br>AppleDaily<br>Liberty Times Net<br>MoneyDJ Finance |
| [90] | LSTM | News Articles | Nikkei Newspaper |
| [87] | SentiWordNet[2] | News Articles<br>Comments | Engadget[3] |
| [109] | SVM, CNN | News Articles<br>Reviews | Thomson Reuters<br>IMDB: movie reviews<br>Electronics product reviews |
| [124] | Attention LSTM, CNN | News Articles<br>Social Media | Baidu News<br>Sina Weibo[8] |
| [125] | SVR[4] | Financial News Articles | Yahoo! Finance |
| [8] | Pre-maid | Financial News Articles | FinSentS Web News Sentiment |
| [126] | LSTM+Attention | Reviews | SemEval-2014 Task 4 [127] |
| [128] | CNN | Reviews | SemEval-2014 dataset |
| [129] | CharSCNN[5] | Movie Reviews<br>Tweets | Stanford Sentiment Treebank<br>Stanford Twitter Sentiment corpus |
| [130] | SVM | Amazon Reviews | Dataset[6] |
| [131] | Attention LSTM | Reviews | Dataset[7] |
| [132] | Bidirectional LSTM | Reviews<br>Micro-blogs | Amazon Reviews<br>Sina Weibo [8] |
| [133] | Attention LSTM | Reviews<br>Yahoo! Answers | SemEval-2015 Task 12 [134]<br>SentiHood [135] |
| [107] | Naïve Bayes<br>SVM | Blogs | Financial Blogs |
| [89] | Naïve Bayes | Forum Texts | guba.eastmoney.com |
| [76] | Lexicon Based | Tweets | Twitter API |
| [83] | Naïve Bayes | Tweets | Sina Weibo[8] |
| [136] | CNN | Tweets | SemEval-2015 Dataset |
| [78] | Random Forest | Tweets | Twitter API |
| [79] | Lexicon Based | Tweets | Twitter API |
| [93] | OpinionFinder<br>Lexicon Based | Tweets | Twitter API |
| [77] | SSN[9] | Tweets | Twitter API |
| [137] | CNN, LSTM | Tweets | Twitter API |
| [138] | Attention Bi-LSTM | Tweets | SemEval-2017 Task 4 [139] |
| [86] | Lexicon Based | StockTwits | Twitter API |
| [101] | Naïve Bayes<br>Decision Trees | Board Messages | Yahoo! Finance Message Board |
| [84] | TSLDA[10] | Board Messages | Yahoo! Finance Message Board |
| [98] | Random Forest | Regulatory Announcements | Dataset[12] |

[1] Topic Sentiment Latent Dirichlet Allocation
[2] https://github.com/aesuli/sentiwordnet
[3] http://www.engadget.com/
[4] Support Vector Regression
[5] Character to Sentence CNN
[6] http://www.cs.jhu.edu/ mdredze/datasets/sentiment/
[7] http://tcci.ccf.org.cn/conference/2013/index.html
[8] http: //weibo.com/
[9] Semantic Stock Network
[10] Topic Sentiment Latent Dirichlet Allocation
[11] Extreme Gradient Boosting
[12] http://www.dgap.de/dgap/News/?newsType=ADHOC
[13] http://www.finet.hk/mainsite/index.htm

- **Sentiment Polarity** is the classification of an author's opinion. In its simplest form it involves just two classes: $\{$positive, negative$\}$. A simple extension is ternary polarity by adding a neutral class [99]. More fined-grained SA can be achieved by detailing positive and negative categories, by adding a category for objective sentences, by categorizing flavors of feelings $\{$fear, rage, joy, interest$\}$, emotions [140] such $\{$Pleasantness, Attention, Sensitivity, Aptitude$\}$ and its level of intensity [97].

  Other approaches, such as review systems, typically use a 3- or 5-star classification system. Usually, qualitative and quantitative systems can be mapped into each other, as long as they have the same number of categories.

  In alternative, some authors [141] have also used a continuous value in a certain range, e.g. $[-1, +1]$.

Table 2.3 summarizes the text sources used in the literature. From the works included is possible to conclude that:

- The most frequent sources are News (12) and Twitter (8).
- News, articles or headlines, are frequent due to its structured form. Headlines are the less sentiment ambiguous form from these three [92]. The most frequent news source is Thomson Reuters because it is the only one without scraping limitations [94].
- Tweets are frequent due to easily detection of sentiment and simple form of accessing and collecting data.
- Again, almost all include a NN model or a comparison between several such models, a major topic of this thesis.

A Stock Prediction system is highly context specific, so it is important to explore multiple financial markets and media sources to build the most accurate possible system [11]. Having this in mind, jointly with EMH strong form, Section 2.3 will resume state-of-the-art approaches combining SA and prediction.

## 2.3 Sentiment Analysis for Time Series Prediction

Stock Prediction deals with an immense amount of trades and complementary information on a daily basis. Furthermore, according to EMH strong form, markets immediately process and adapt to any information released. Therefore, to solely rely on humans to analyze financial and media data in order to efficiently do SP is inconceivable. Consequently, it is necessary to automate both financial and media data process of scraping and examination [22].

Opposed to Section 2.1 where SP only considers individual [93] or multiple [87] stocks, and after Section 2.2 introduction to the field of SA, Section 2.3 includes approaches that use media information SA as additional features to do financial data prediction [84] thereby improving directional movement accuracy [88, 94]. In addition, sometimes it is even possible to predict its interval growth/decline duration [91].

DL techniques, in particular NN models, achieve the highest accuracy [132]. RNN, due to its ability for SA tasks, is the most complete model to build a SP system incorporating media analysis [142]. Several comparisons between RNN models such as Elman, Jordan and LSTM were made [143], having the latter the highest directional accuracy registered throughout our review [89].

The media sources used with ML models are:

- **News** [144] and headlines [145] being the most analyzed type of media. Almost every type of model from Statistical [4, 146] to Naïve Bayes [147], KNN [146], CNN [61] and RNN [61] used this source. Refs. [4, 12, 125, 148] used a more fine-grained scope, only analyzing financial news articles. Ref. [149] compared the impact of major and once-occurrence events, causing dramatic impact on the stock market against minor and more frequent ones, that cause a lower impact on prices. It was concluded from all articles regarding news that:
  - Private information causes insignificant or low changes in stock price [150] and contrarily, public information causes significant changes [151]. Therefore it is consensual that prediction accuracy is primarily influenced by public information. Nonetheless, the use of private information allows the possibility to improve the accuracy a bit further.
  - Markets tend to overreact when presented with bad news [145].
  - It is easier to predict downward trends than upward trends [152].
  - In general, pessimism tends to increase trading volume and lead to predictions of negative returns the next day [153]. However, disappearing within one week.

- **Social Media Platforms** nowadays have more influence in stock performance than conventional media attributes [154]. Not much work has been done over Facebook information due to privacy concerns restraining data availability [155, 156]. However, both trends and searches of Google and Wikipedia are useful tools to measure investors attention and interests [157, 158]. Both have successfully complemented financial data prediction [159] having a direct correlation with volatility and trading volume [160].

- **Twitter** sentiment and message volume showed a consistent correlation with stock movement and traded volume, correspondingly [161–164]. An additional feature is the author's trustworthiness [78, 105]. Similarly to financial news, to use StockTwits instead of tweets for a more fine-grained SA analysis can be beneficial [165]. Alternative approaches proposed to:
  - exploit character- to sentence-level tweets information in order to acquire more context from Tweets [129];
  - create a stock graph based on stocks similarity between historical prices or co-occurrence in the same tweet [77];
  - create a concept map relating different concepts present on tweets and organize them hierarchically [166];
  - visualize the sentimental trajectory of each emotion in retweets through Google Maps [137].

- **Blogs**, chat rooms and discussion boards are tools which users can use to form opinions and quickly share them widely [167]. Information can be searched, diffused and authored with ease. This can make a sentiment about specific stock assets or entire markets go viral [168].

  Refs. [107, 169] demonstrated promising results as the baseline, strongly correlating frequency with stock trading volume and producing significant results regarding price movement. Minor works were proposed with forum texts [89]. Out of this work scope, YouTube data and its comments were analyzed in order to measure online radicalization and its overall usefulness [170].

- **Financial Reports** released by companies, public media or third-party institutes contribute to EMH semi-strong form by fully reflecting publicly-available information. Based on this premise, it is expected that whenever reports enter the market a price changes occur [98]. When the market is in uptrend and optimistic reports were released, the investors reacted positively. On the contrary, in the presence of a downtrend and negative reports were released, the investors tend to react more [171].

  In the real world, in order to provide equal access to all participants, regulators ensure that stock-relevant information is revealed via regulatory disclosures. However, it cannot be guaranteed that no misleading information is present.

Table 2.4 provides an overview of the results reported in the literature in terms of directional prediction accuracy. Each column details a source, mentioned in Section 2.2; each row a NN model further analyzed in Section 3.2; and each cell a financial data prediction work detailed in Section 2.1 (without SA) or in Section 2.3 (with SA).

Table 2.4: Techniques and data sources for stock prediction based on sentiment analysis.

| Source / Model | News | Social Media | Twitter | Blog & Forums | Financial Reports | None |
|---|---|---|---|---|---|---|
| **Statistical** | [4, 90, 146] | [83, 159] | [76, 77, 79] | | | [23, 42, 82, 172, 173], [19, 38, 44, 57, 66], [21, 45, 58, 81] |
| **KNN** | [12, 146] | [12] | | [12] | | [38] |
| **Naïve Bayes** | [12] | [12] | [166] | [12] | [101] | [85] |
| **MLP** | [90, 174] | | [174] | | | [56, 59, 69, 82] |
| **Random Forest** | [8] | [158] | [86, 166] | | [101] | [38, 46, 80, 81, 91] |
| **SVM** | [7, 87, 88, 175], [8, 12, 61, 125] | [12, 83, 158] | [79, 166, 175] | [12, 89] | [84] | [19, 30, 36–38, 46], [21, 34, 39, 81, 89, 176] |
| **NN** | [7, 8, 146] | [158] | [79, 93] | | | [23, 42, 46, 49, 50, 67, 80, 95], [19, 30, 37, 38, 57, 66, 82, 173], [21, 43–45, 47, 51] |
| **CNN** | [92, 174, 177, 178], [61, 94, 179] | | [174, 177, 179] | | | [21, 56–59, 69] |
| **RNN** | [61, 90] | | | | | [21, 56, 58, 59, 67] |
| **LSTM** | [90, 175, 180], [10, 94, 177] | [159] | [175, 177] | [89] | | [21, 56, 58, 65–67, 69, 80, 89] |
| **RL** | | | | | | [62, 72–75] |

In the previous Table 2.1 and Table 2.2 we have provided information about stock data sources and, in Table 2.3 we have provided information text data sources. Each of the aforementioned tables focusing exclusively on one specific topic. On the other hand, Table 2.4 focuses on aggregating those works. From this table we were able to conclude that:

- Nearly half of the articles only use financial data when doing prediction, due to being support literature of Section 2.1.

- Most frequent type of media source used are News and Headlines (grouped together) and Twitter (due to its frequency is distinguished from other social media sources).

- Most works focused on increasing predictability and decreasing stock movement volatility [172]. By combining multiple sources, confidence when predicting is augmented. However, it rarely is employed more than one media source. When taking into consideration EMH strong form, in my opinion this can be viewed as contradiction.

- Accuracy values tend to increase from top to bottom rows. Traditional statistical models, KNN, Naïve Bayes, MLP, Random Forest and SVM have problems integrating SA features into its models due to sparsity problems, or simply can not, hence achieving lower accuracy than DL models.

- LSTM is the model registering the best accuracy overall [89], corroborating the idea that DL models are better suited for incorporating SA when doing Stock Prediction.

However, a word of caution is necessary regarding these models performance. According to [10], academic works about financial predictions are often misleading. It is believed that many papers tend to overfit models due to heavy simulations and exaggerate results to gain recognition. It is also stated that many works can not be generalized, being used only in very specific contexts with artificial data, with many of the simulations being hard to reproduce. In addition, it is likely that the best performing models are developed and kept secret in private companies for competitive reasons.

Since each work in Table 2.4 uses a different dataset, the prediction models cannot be compared directly, even if their evaluation results are available. However, it is fair to say that prediction accuracy tends to increase from the top row to the bottom rows. In general, it is not easy to integrate SA features into statistical models, KNN, Naïve Bayes, MLP, Random Forests and SVM due to sparsity issues. Hence, they achieve lower accuracy than DL models. On the other hand, a LSTM incorporating SA registered the highest directional accuracy [89]. For us, this is not surprising since RNNs are especially appropriate for NLP and SA tasks.

## 2.4 Summary

In conclusion, SP is a field that has been thoroughly explored, and there is a wide variety of models being employed. Since the goal of this work is to improve the accuracy when predicting stock movement, models with and without perceived investors sentiment were the subject of analysis. Having this in mind, we reviewed state-of-the-art approaches both with and without SA. From works including SA it was possible to conclude that news were the most frequent source.

Finally, it is worth noting that many of these works employ ML techniques, which often provide the best results. Especially deep learning models, such as LSTM, appear to be one of the most promising

approaches [69]. However, there are still limitations. Therefore, in the following chapter we will better explore the architecture of deep learning models.

# Chapter 3

# Deep Learning for Stock Prediction

In Chapter 2, we provided a literature review over the main components of Stock Prediction (SP). It was possible to notice that, due to its properties, Deep Learning (DL) techniques are the best performing solution when combining both components in this prediction task. In particular Long Short-Term Memory, a type of Recurrent Neural Network, registered the highest accuracy.

In this chapter, Section 3.1 and Section 3.2 review the origins of DL, the fundamental concepts of Neural Networks and the most well-known models. A reader already familiar with the topic and concepts can start directly in Section 3.3, where Attention is introduced and we discuss how this mechanism can be incorporated into Stock Prediction.

## 3.1 Fundamentals of Deep Learning

Deep Learning [181], a field of Machine Learning, empowers most human-like Artificial Intelligence (AI) tasks. Models are used to parse data, learn from it, and make decisions based on what it has learned using strategies such as:

- Supervised Learning (SL) [182], in which the model should reach the same answer as the solution provided by examples;
- Unsupervised Learning (UL) [182], where there is no correct answer. No explicit instructions on what to do are given, raising challenges [183], solved by methods such as association rules [184], auto-encoders [185], anomaly detection [186] and clustering [187];
- Reinforcement Learning (RL) [188], that attempts to find the optimal way to accomplish a particular goal. This is achieved by trying to predict the best next action in order to earn the biggest final reward possible.

In recent years, DL has become of more frequent use as available amount of training data increased and as computer hardware and software infrastructures improved [189]. In addition, NNs are a general model, that easily adapts to multiple domains significantly outperforming other ML techniques in

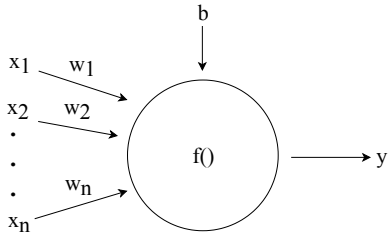
problems such as Computer Vision [190] and Natural Language Processing [191]. Deep Learning has become the state-of-the art approach in many fields, including Stock Prediction.

- **Origins** of Deep Learning can be traced back to AI's definition as "the science and engineering of making intelligent machines, especially computer programs"[1]. AI motivation is to create the ability for a computer to learn from data and intelligent enough to automate tasks that people feel being redundant or time consuming. However, intuitive tasks for people such as face recognition are the real challenge to AI. These are complex to formally describe, therefore hard to solve.

  Machine Learning derives as AI field of study where tasks are performed without explicit instructions. This is based on the premise that solely relying on statistical techniques, algorithms and inference, computers learning can improve with experience. A Neural Network [192] is the most known model of ML along with: Random Forest [193], K-Nearest Neighbour [194], Hidden Markov Model [195], Support Vector Machines [196], Naive Bayes [197], Decision Trees [198], etc.

  Deep Learning[2] refers to the use of deeper NNs, i.e. networks with multiple Hidden Layers (HLs). A NN can have a single or multiple HLs. Deep NNs have two or more, hence the name. The additional HLs allow to build complex concepts out of simpler ones and to eliminate the feature selection and reduction processes necessary for classical ML models to perform well.

- **Neurons and Activation Functions** are the basic units in a NN [199]. As shown in Fig. 3.1, a neuron receives values as inputs ($x$) associated with weights ($w$) assigned by their importance in comparison to other $x$. These $w$ are updated through Back-Propagation (BP), the calculation of a function loss gradient (difference between generated and desired output). The higher a gradient is, faster the learning. BP increases generalization and minimizes output error.




$$y = f(b + \sum_{i=1}^{n} x_i \times w_i) \tag{3.1}$$

Figure 3.1: Single neuron.

Output value ($y$) is calculated as described in Eq. (3.1). $f$ is an AF that can be linear or non-linear. Non-linear AFs are the most used since real data is non-linear, thereby being capable of more complex representations. Sigmoid, $tanh$ and Rectified Linear Unit (ReLU) are the most known non-linear AFs. ReLU, the most commonly used, saves a great amount of time and a complex calculating process [95], effectively improving NNs training speed.

In the next section we will explore NN models in more detail.

[1] 1956 – John MacCarthy
[2] 2000 – Igor Aizenberg

## 3.2 Neural Network Models

Neurons, also called nodes, are organized into groups forming layers and can be one of three types:

- Input Node where each independent variable is represented by one node and, collectively are referred as the Input Layer providing information from outside the NN;
- Hidden Node that performs computations through an AF and do not have a connection with NN's outside. Grouped, form one or more Hidden Layers;
- Output Node responsible for transferring results to NN's outside, jointly forming the Output Layer.

The most common NN models are:

- **Feed-Forward Neural Network (FNN)**, the simplest architecture. Based on the Single-Layer Perceptron [200], its name derives from the fact that information flows one-way, from input to output layer, across a set of densely-connected layers. A FNN is composed of multiple Perceptrons, where all nodes are fully connected to subsequent layers in the network, hence its designation of Multi-Layer Perceptron. There are also Deep FNNs composed by at least two HLs (Fig. 3.2).

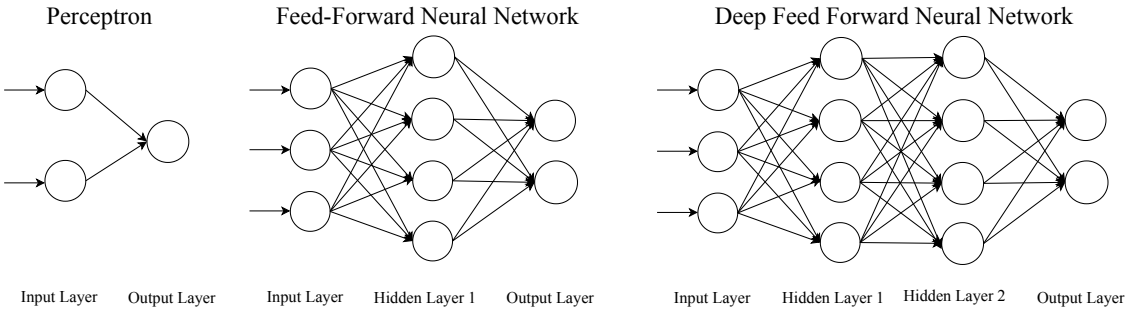



Figure 3.2: Feed-Forward Neural Networks.

  As information always moves in a forward direction, a node has no memory of previously received inputs. A FNN only considers the current input, having no notion of order in time, thus not the best in prediction tasks containing time correlations. Deep FNNs are the most general-purpose NNs being able to provide the best performance given enough layers, neurons in each layer, data and time to train the network [201]. This model is primarily used in Supervised Learning regression and classification tasks where data to be learned is neither sequential nor time dependent usually finding local minima, which is often enough.

- **Convolutional Neural Network (CNN)** was designed to receive as input matrix shaped data thereby requiring a much smaller number of parameters when dealing with 2D data in comparison to other NNs [52]. CNNs are very popular in image processing where the input is 2D, but they are also very successful at processing 1D input such as text sentences and Time Series [53], as previously mentioned in Section 2.1. CNNs are composed by:
  - Convolution Layers comprising a set of independent filters [52, 202]. Each filter trained to detect a specific feature present in an image (e.g. an edge), returning a value representing

how much confidence there is in its presence. The result is a matrix that stores convolutions results over the various parts of an image (Fig. 3.3).

– Pooling Layers [202] aiming to reduce dimensionality of each feature mapped (decrease number of parameters) while maintaining crucial information. Commonly known as downsampling or subsampling, the most common is Max Pooling [203], which selects the largest element within that pooling window. By concentrating information, computation is reduced and the network becomes translation invariant (e.g. a face slightly translated from the middle of an image). By reducing memory consumption more Convolutional Layers can be used, hence more complexity can be represented.

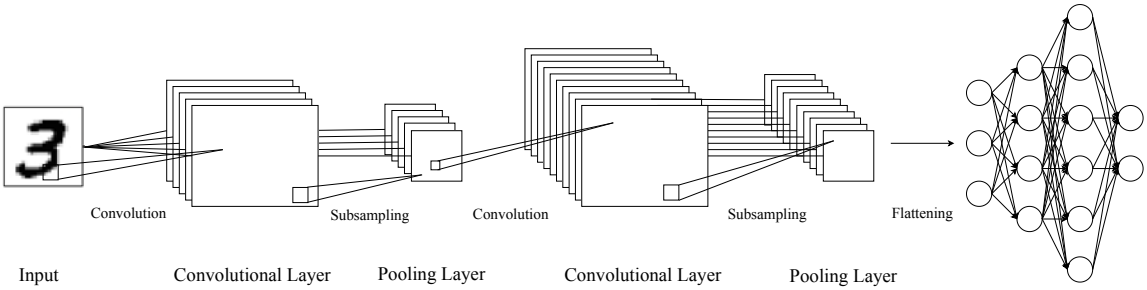
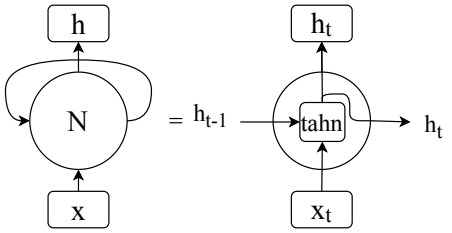



Figure 3.3: Convolutional Neural Network.

In addition, pooling is also used to handle overfitting, a situation when the model cannot generalize future unseen data. This model requires significantly less parameters and can even outperform humans at image recognition tasks [129]. However, the pooling area can not be too large, otherwise too much information is lost and performance decreases. Despite not being rigid about a fixed data size, there is still a need for minimum shape/dimension of data and a concern that input size does not vary to wildly.

- **Recurrent Neural Network (RNN)** differs from previous models since they keep an internal state and have a feedback loop (left of Fig. 3.4) to use that internal state as an additional input at each time step. The recurrence loop provides the ability to consider past activations, hence making more informed predictions [60]. When solving problems involving sequential and temporal data, RNN should be the model used because it is able to handle inputs of arbitrary lengths. For Stock Prediction, this means that it is possible to reuse past stock behavior to predict the future.




$$h_t = f(b + x_t \cdot W + h_{t-1} \cdot U), \quad \text{where } 1 \leq t \leq T \tag{3.2}$$

Figure 3.4: RNN neuron unrolled.

RNN can be seen as a sequence of neurons trained where information passes from one time-step to the next. Eq. (3.2) details the output computation on time-step $t$. Typically, $tanh$ is used as AF ($f$). Instead of BP, Back-Propagation Through Time (BPTT) (the same algorithm for recurrence [204]) is used to update the weights ($W$ and $U$) in order to minimize the output error [205]. However,

since the error of a given time-step depends on the previous one, if the network has a high number of time-steps, BPTT can be computationally expensive [63].

Two additional RNN problems are: Vanishing Gradients [206, 207], where low values are assigned to weights, causing information to rapidly be lost over time and; Exploding Gradients [207, 208], where high values are assigned to weights, giving them excessive importance or causing overflow. In both cases, the state will not be very informative. Elman [209] and Jordan [210] are variants using the previous HL and Output Layer, correspondingly. Nevertheless, memory loss remains challenging.

RNN can only remember recent past, being unable to perform well when a long-term context is required. RNN is not simultaneously robust enough to handle input noise and efficiently trainable by gradient descent when more context is necessary [63].

- **Long Short-Term Memory (LSTM)** [64], a type of RNN, improves learning of long-term dependencies due to $C_t$. In addition to the current input ($x_t$), a LSTM cell also receives the previous LSTM hidden state ($h_{t-1}$) and the previous LSTM cell state $C_{t-1}$. All inputs being provided in a vectorial format. This cell state enables LSTM to better deal with long range dependencies when compared to standard RNNs [211], by enabling to carry these dependencies across multiple time-steps.

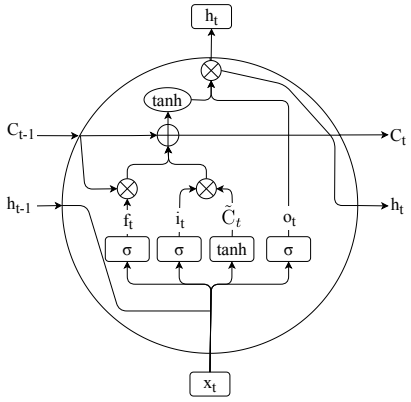



Figure 3.5: LSTM neuron architecture.

$$f_t = \sigma(b_f + x_t \cdot W_f + h_{t-1} \cdot U_f) \tag{3.3}$$

$$i_t = \sigma(b_i + x_t \cdot W_i + h_{t-1} \cdot U_i) \tag{3.4}$$

$$\tilde{C}_t = \tanh(b_c + x_t \cdot W_{\tilde{c}} + h_{t-1} \cdot U_{\tilde{c}}) \tag{3.5}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{3.6}$$

$$o_t = \sigma(b_o + x_t \cdot W_o + h_{t-1} \cdot U_o) \tag{3.7}$$

$$h_t = \tanh(C_t) \odot o_t \tag{3.8}$$

The key difference being that information is transferred along the entire chain with only some minor interactions mitigating RNN gradients growth or decay difficulty in long input sequences [63, 206–208]. This is possible because each LSTM neuron contains an additional three gates (Fig. 3.5):

  - "Forget Gate Layer" (Eq. (3.3)) decides which information received from the past ($C_{\text{t-1}}$) is going to be remembered or forgotten (0 – forget and 1 – remember).
  - "Input Gate Layer" (Eq. (3.4)) decides which values will be updated in the new state and $tanh$ (Eq. 3.5)) creates a vector of possible values ($\tilde{C}_t$) to be added. In order to actually update the state, the old state ($C_{t-1}$) is added with the updated ($i_t$) and candidate ($\tilde{C}_t$) values (Eq. (3.6)).
  - "Output Gate Layer" (Eq. (3.7)) decides which information is going to be outputted ($o_t$) by combining the cell state and the input at time-step $t$ through an element-wise multiplication (denoted by the operator $\odot$). $tanh$ forces the output to be between -1 and 1 (Eq. (3.8)).

As NNs are further researched, slightly different versions are being developed. A very well know alternative is Gated Recurrent Unit (GRU) [68] that only has an update gate. Modifications such as coupling the Input and Forget gates simplify LSTM models without significantly decreasing performance [212]. This variant is attractive because it reduces the number of parameters and the computational cost. GRU merges the cell ($C_t$) and hidden state ($h_t$), and also combines the forget ($f_t$) and input ($i_t$) gates. As it is more efficient computationally, more use has been seen recently. However, removing or changing the Forget Gate and the Output AF significantly impairs performance, proving the mandatory presence of these components. GRU is a simple model when compared to LSTMs, however as a drawback this model exposes the hidden content represented without any control.

## 3.3 Attention Mechanisms

Sequence to sequence problems [213] such as chatbots [214], language translation [215–217] and text summarization [218, 219] are the most common tasks solved by RNNs. However, these are distance dependent and RNNs are not capable to cope with too long sequences. The same situation happens when considering long time-frames for Stock Prediction. Although an improvement is noticed with LSTM models, there are still limitations regarding vanishing gradient [220]. To summarize, relevant information is lost when dealing with long sentences because RNN tends to only remember the previous state and LSTM is unable to compress all information into a fixed-length vector.

Table 3.1: Alignment score functions.

| Name | Equation | Paper |
|---|---|---|
| Content-Based | score$(s_t, h_i) = \cos(s_t, h_i)$ | [221] |
| Additive | score$(s_t, h_i) = v_a^T \cdot \tanh(W_a \cdot [s_t; h_i])$ | [215] |
| Location-Based | $\alpha_{t,i} =$ softmax$(W_a \cdot s_t)$ | [217] |
| General | score$(s_t, h_i) = s_t^T \cdot W_a \cdot h_i$ | [217] |
| Dot-Product | score$(s_t, h_i) = s_t^T \cdot h_i$ | [217] |
| Scaled Dot-Product | score$(s_t, h_i) = \frac{s_t^T \cdot h_i}{\sqrt{n}}$ | [222] |

$s_t$ is the target state at time-step t
$h_i$ is the source hidden state at time-step i
$v_a$ is a trainable weight vector in the Attention layer
$W_a$ is a trainable weight matrix in the Attention layer
$n$ is the dimension of the source hidden state
$T$ is the Transpose

Attention mitigates this constraint [223] by selectively retrieving information from hidden states with an alignment score function [215], with the most known functions being detailed in Table 3.1. The underlying rationale behind Attention is to increase the focus on important parts of the input instead of just trying to remember it afterwards [213]. This method allows to select the most relevant features. Results suggest that the use of attention can capture longer-term dependencies and outperform stand-alone LSTMs [10].

The existing types of Attention are:

- **Sequential Attention mechanism** [215], where the model creates a representation of the input and then attention focus on specific parts of that representation. This process enables a faster

learning and to more accurately represent an input [213], being the main advantage achieving higher performances.

Without attention and when predicting with a RNN model, $h_T$ (final output on the left of Fig. 3.6) contains the representation value for each time-step of the input. In our task, in order to determine a trend, $h_T$ is fed into a softmax layer which outputs the Time Series belonging probability to each of the target classes. However, $h_T$ is not able to store all relevant information given a long input.

In addition, for example in SA tasks, the overall sentiment of a sentence is decided by a small set of words, hence the same importance should not be given to all words, nor more importance to more recent ones. The same happens with financial data and its price peaks and bottoms throughout time.

When using Sequential Attention, this mechanism is typically located at the end of the model. Therefore, allowing this mechanism to use hidden states outputs of previous layers to support predictions (Fig. 3.6). There are two types of Sequential Attention: local, where only some hidden states are used, normally near time-step $t$; and global, where all hidden states are used [217].

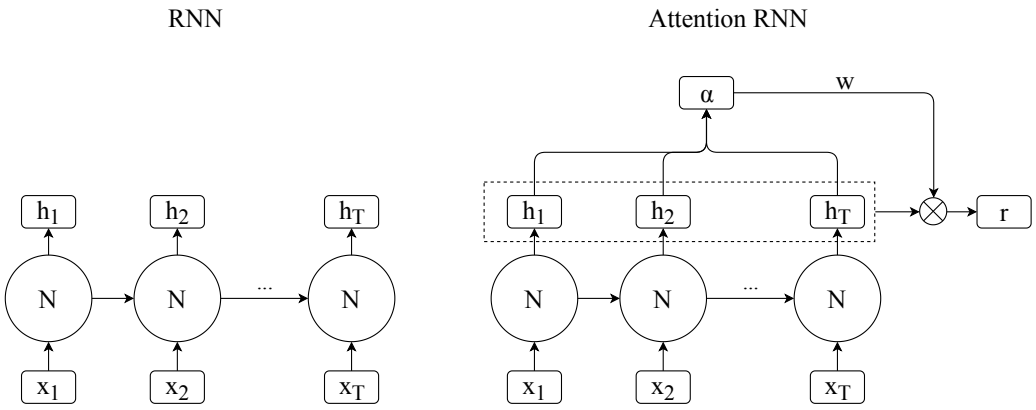



Figure 3.6: RNN vs Attention RNN.

The Attention mechanism is composed by a fully-connected layer that receives hidden states as input (Fig. 3.7). This layer applies the additive alignment score function (Eq. (3.9)) and calculates $e_t$ (Eq. (3.10)), the importance value of $t$ hidden state. Sequentially, all $e_1, e_2, ..., e_T$ will be used to calculate $\alpha_1, \alpha_2, ..., \alpha_T$ through a softmax function (Eq. (3.11)) [126, 133, 138]. Each $\alpha$ captures the probabilistic degree of importance that input $t$ has on the whole input. All verify the properties stated in Eq. (3.12). The higher $\alpha$ the higher importance, and consequently effect $t$ input has on the global category prediction. In order to determine the final category, $r$ (Eq. (3.13)), the equivalent to $h_T$ in RNN without attention, must be fed into a softmax layer which returns the belonging probability to each of the target classes.

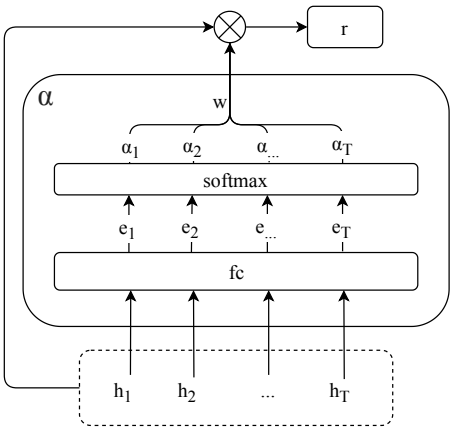



Figure 3.7: Attention mechanism.

$$fc(h_t) = v_a^T \cdot \tanh(b_h + h_t \cdot W_t) \tag{3.9}$$

$$e_t = \text{fc}(h_t), \text{ where } e_t \in [-1, 1] \tag{3.10}$$

$$\alpha_t = \frac{\exp(e_t)}{\sum_{i=1}^{T} e_i}, \ 1 \leq t \leq T \tag{3.11}$$

$$\alpha_t \in [0, 1] \text{ and } w = \sum_{i=1}^{T} \alpha_t = 1 \tag{3.12}$$

$$r = H \cdot w^T, \ H = [h_1, ..., h_T] \quad w = [\alpha_1, ..., \alpha_T] \tag{3.13}$$

- **Parallel Attention mechanism**, was purposely designed for NLP sequence to sequence tasks and is currently state-of-the-art for translation. Created in the Transformer [222], Parallel Attention is a novelty because it does not involve any recurrence. This mechanism solely relies on attention, not requiring prior calculation of all previous hidden states, therefore enabling significantly faster computation and consequently reducing training time. As the importance of every input is calculated at the same time, all inputs interact simultaneously with each other. Therefore, when processing long textual sequences, Parallel Attention improves the extraction of dependencies.

To our best knowledge this model was never applied to Time Series Prediction tasks. Therefore we will be providing a full explanation on how this model was implemented for textual inputs processing. Fig. 3.8 details the part of the Transformer architecture necessary for NLP classification tasks, the Encoder. However, in our context the Transformer architecture can even be more simplified. This will be further explained in Section 4.5.

$$\text{PE}_{(\text{pos};2i)} = \sin\left(\frac{\text{pos}}{10000^{\frac{2i}{d_{\text{model}}}}}\right), \text{ where } 1 \leq i \leq n \tag{3.14}$$

$$\text{PE}_{(\text{pos};2i+1)} = \cos\left(\frac{\text{pos}}{10000^{\frac{2i}{d_{\text{model}}}}}\right), \text{ where } 1 \leq i \leq n \tag{3.15}$$

When working with textual inputs, it is necessary to convert the input words $x_1, ..., x_n$ to vectors, hence the necessary input embedding. In addition, as the Transformer does not have any type of recurrence, positional encoding is necessary to have some notion of order to deal with sequential data (Eq. (3.14) and Eq. (3.15)). Both vectors, having the same size ($d_{\text{model}}$), are summed thus attributing a relative position to each word. The functions $sin$ and $cos$ are used because no training is necessary, there is no need to know maximum length of sequence allowing them to vary in the training and test set.

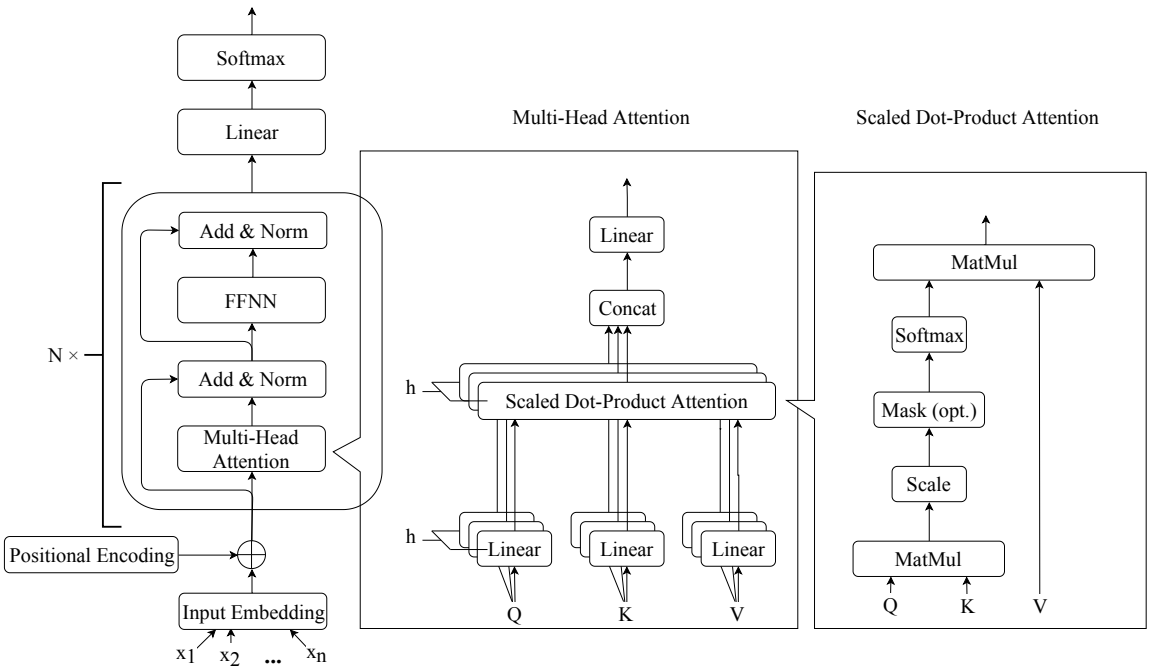



Figure 3.8: Transformer Encoder architecture.

Following embedding and encoding, a stack of $N$ identical components follows, each containing two main layers. The first layer incorporating a Multi-Head Attention mechanism and the second a fully connected NN.

The Multi-Head Attention layer, instead of an individual calculation for each word $x_i$, groups all words forming a matrix $X = [x_1, ..., x_n]$ allowing parallel computation and speeding up the process. By multiplying $X$ with matrices $W_q$ (Eq. (3.16)), $W_k$ (Eq. (3.17)) and $W_v$ (Eq. (3.18)) all queries ($Q$), keys ($K$) and values ($V$) are calculated at the same time. For word $i$ its encoding is $x_i$, its query $Q_i$, its key $K_i$ and its value $V_i$ (being $i$ the row in each matrix). Representing $Q_i$ the target state ($s_t$), $K_i$ the current hidden state ($h_i$) and $V_i$ all hidden states. This is relevant because each word can interact with all other words at the same time, both backwards and forwards, a significant difference when compared to RNNs.

$$Q = X \cdot W_q \tag{3.16}$$

$$K = X \cdot W_k \tag{3.17}$$

$$V = X \cdot W_v \tag{3.18}$$

Following, the scaled dot-product alignment score function (Table 3.1) can be adapted to matrices and applied to all time-steps simultaneously (Eq. (3.19)). Similarly to Sequential Attention, a softmax function (Eq. (3.11)) is applied to obtain a probabilistic importance value for each input term $x$. A scaling factor of $\sqrt{d_K}$ (dimension of $K$) is used to guarantee a mean of 0 and variance of 1 thereby improving numerical stability as $Q$, $K$ and $V$ grow obtaining a gentler softmax. Then, each probabilistic value is multiplied by the value of each word $V$ resulting in $Z$ (similarly to Eq. (3.13)). Each row of matrix $Z$ represents the weight of each input $x_i$ (right of Fig. 3.8).

$$Z = \text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V \tag{3.19}$$

All the aforementioned is only for one Head of the Transformer. In ref. [222] $h = 8$, meaning that exist 8 Parallel Attention heads focusing on different features thereby allowing the representation of multiple subspaces. Ultimately, with enough parameters it is possible to simulate a convolution. Each head can focus on the next word; or on the previous word; or in identifying identical or related words in the same sentence; or in other sentences. Or in case of NLP tasks, analyze the three most important features in a sentence: "Who?", "Did what?" and "To whom?". However, the next layer is expecting a single matrix not $h$ matrices. Therefore, these $h$ matrices are concatenated into one (middle of Fig. 3.8).

The second layer, is applied to all positions independently and with different parameters ($W_1$ and $W_2$) in each layer (Eq. (3.20)).

$$\text{FNN}(x) = b_2 + \text{max}\left(0, b_1 + x \cdot W_1\right) \cdot W_2 \tag{3.20}$$

Both layers have an Add & Normalize component following them (Eq. (3.21)). Adding allows to carry positional information to next layers and as NN is computationally expensive, normalization can speed-up the training [224].

$$\text{Add \& Norm}\left(x\right) = \text{LayerNorm}\left(x + \text{Sub-layer}(x)\right) \tag{3.21}$$

A Linear layer then performs dimensionality reduction/augmentation to a predefined space of dimension $d$. In the end, similarly to previously mentioned approaches, a softmax layer predicts a sentence belonging probability to each of the target classes.

The Transformer was initially designed for NLP sequence to sequence tasks. Nonetheless in our context, Time Series Prediction, the Transformer architecture can be simplified.

As already previously mentioned, when doing classification tasks with a Transformer-like model the decoder component can be removed. In addition, when working with numerical data as financial Time Series, the embedding layers typically used for NLP tasks are not necessary, hence can be removed.

Additional modifications to this model can be made. As choosing between one and several heads of attention. This choice relies in whether try to leverage multiple layers of attention in parallel in order to enhance the extraction of dependencies from the input sequence or not. An interesting approach is the Self-Attention Network (SANet) [225], a Single-Head Attention approach.

## 3.4 Summary

Previously, Chapter 2 detailed several approaches to SP with and without SA. In that chapter, we concluded that DL models are the most frequently employed due to its adequacy when performing SP tasks. In this Chapter 3, we detailed the fundamentals of those models, the most relevant architectures and the Attention mechanisms. The goal was to provide a broader view in order to better leverage the financial and sentiment features when predicting stock's movement. The next chapter presents our proposed approach and a case study. In that chapter it will be detailed what type of data is going to be incorporated, how it was obtained, the preprocessing performed, the characteristics of each NN model, and how performance will be evaluated.

# Chapter 4

# Case Study and Approach

In conformity with the main goal of this dissertation, which is to determine how much Sentiment Analysis (SA) helps improving Stock Prediction (SP) accuracy, this chapter describes the experimental approach and corresponding evaluation methodology employed on a specific case study. To quantify the benefit of incorporating SA into SP we will mainly employ Deep Learning (DL) models, which came out as the most successful approach from our literature review in Chapter 2.

Our work also contains a comparison of models with, without and solely relying on Attention. In particular, Parallel Attention its a novelty and state-of-the-art approach for NLP tasks that to our best knowledge was never employed in Time Series Prediction (TSP) tasks.

In order to achieve these goals, this chapter is subdivided into: Section 4.1 where we detail the origin of the data; Section 4.2 where the features of each dataset used are described; Section 4.3 details the preprocessing done on that data; Section 4.4 explains the input layers for each model; Section 4.5 characterizes the different models employed, explaining the architectures used to compare performance when doing SP with and without SA while incorporating different types of Attention; and Section 4.6 specifies how the results are going to be evaluated and which metrics are used. The next chapter will analyze the results obtained from these experiments.

## 4.1 Data Source

In Chapter 2, several works were analyzed with the most common data sources being described and summarized. From this review it was possible to conclude that news were the most often used data source. Therefore, in alignment with the goal of this work, for our experimental approach two types of data are necessary:

- **Market data**, with typical indicators about stock prices, trading volumes and calculated returns. This type of data can be acquired from popular financial platforms, previously detailed in aforementioned Table 2.1 and Table 2.2 on pages 8 and 9, respectively.

- **News data**, referring to a specific stock asset or set of assets and the corresponding sentiment

for each news item. This type of data can be acquired from previously detailed main sources in aforementioned Table 2.3 on page 13.

An alternative to the previously mentioned data sources is to retrieve data from prepared datasets with fixed and past time frame such as the ones available in Kaggle (`https://www.kaggle.com`) or SemEval [126, 128, 133, 136, 138] competitions.

In our evaluation, we report on a series of experiments that we performed during the Kaggle competition *Two Sigma: Using News to Predict Stock Movements*.[1] The aim of this event was very much aligned with our goal, since it focused on understanding how news sentiment might influence stocks future return. One key aspect of this competition was that it provided two different datasets: one containing market data and other with already analyzed news data with assigned sentiment.

The option of utilizing a prepared dataset enabled us to skip the data retrieval phase, allowing us to place our focus on testing different model architectures instead.

## 4.2 Data Description

The original dataset contained market and news data from the beginning of 2007 to the end of 2016. The market data included information on $\sim$3000 US-listed companies, containing over 4 million samples with 14 features such as date, asset code, asset name, daily open and close prices, daily trading volume, and open-to-open and close-to-close returns. These returns were calculated both daily and for a 10-day period, and also both in raw and market-residualized form (i.e. by removing the movement of the market as a whole and leaving only the movement inherent to the asset).

On the other hand, the news data was collected from 30 different news media companies such as Reuters News and Business Wire. The news dataset contained about 9 million samples (daily news items) regarding the same market dataset companies or related ones. The news dataset was much larger than the market dataset due to typically existence of multiple news items per stock on a given day. For each news item, the sentiment was given as a probability distribution over 3 classes: positive, neutral and negative. Measures for the novelty and volume counts of each news item were also available, where novelty was calculated by comparing the asset-specific content of a news item against a cache of previous news items, and the volume was calculated by counting how many news items mentioned the asset within a certain time frame. However, for this work we used only the sentiment features of each news item as sentiment signals for each company in a determinate date.

As for the target variable, the competition used the open-to-open market-residualized return over a 10-day period into the future, with this feature being an integrating part of the market dataset. To calculate this variable, two metrics are necessary: the open-to-open return over a 10-day period of a specific asset, and the open-to-open return over a 10-day period of that specific asset index. The open-to-open return over a 10-day period for an asset is calculated by the ratio between the difference of the open price on $t$ and $t-10$ days, and the open price on $t-10$ day, multiplied by 100. Concerning the

[1] `https://www.kaggle.com/c/two-sigma-financial-news`

return of an index, this is the sum of all its assets open-to-open returns over a 10-day. To obtain the market-residualized return for an asset, it is necessary to subtract the open-to-open return over a 10-day period of a specific asset by the same metric of its corresponding index. In essence, the target variable represents the actual movement of a particular asset by removing its inherent fluctuations caused by the market as a whole.

## 4.3 Data Preprocessing

To answer the main goal of this work, i.e. to quantify by how much news sentiment influence the future return of stocks, we used 2 categorical and 15 numerical features as our data. The categorical features were asset code and asset name, common in both the market and news datasets. We used both features because each asset name (i.e. company) may have several asset codes, but each asset code belongs to a single company. This is due to a company being present in more than one stock exchange market, hence having a specific asset code for each market. As for the numerical features, 12 of those features came from the market dataset, and the remaining ones correspond to the 3 aforementioned sentiment classes from news data.

As we want to determine the influence of news SA influence on SP accuracy, we will compare models using solely the market data versus models incorporating the aforementioned sentiment classes of news data. This can be translated into two sets of inputs: models without sentiment, using 2 categorical and 12 numerical features from the market dataset; and models with sentiment, using 2 categorical features, 12 numerical features from the market dataset and 3 numerical features from the news dataset.

The sentiment classes have to be mapped from the original news data to the market dataset. This mapping is done by using the categorical features as primary key, as both features are common elements in the two datasets. When joining the news data sentiment classes with the market data, it may be the case that there are several news items for a given asset on a given date. In this case, we group those news items together and compute their average sentiment. On the other hand, it may be the case that there are actually no news items for a given asset on a particular date. In this case, we have tried three different approaches to fill in the sentiment for the asset, namely:

- *propagation*, where we use the sentiment of the previous date as the current sentiment, hence propagating the sentiment across multiple days;
- *balancing*, where we use a uniform probability distribution over the three classes (positive, neutral and negative, with $\frac{1}{3}$ probability for each);
- *neutralizing*, where we assign zero probability to the positive and negative classes, and 1.0 to the neutral class.

The goal, as identified by the target variable, is to predict stock returns over a 10-day period of time into the future. Therefore, we need to apply a training window technique to input several samples into the model in order to make a prediction.

As observed in Fig. 4.1, the training window can vary across the following options:

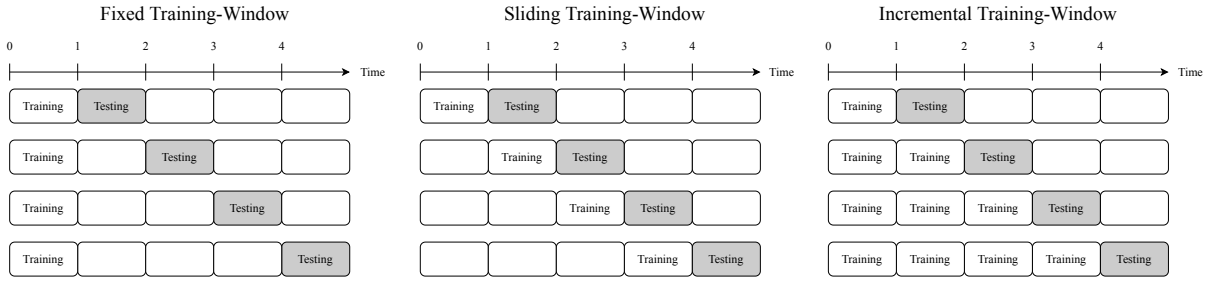



Figure 4.1: Training-window options.

- *fixed*, where the training-window will always be the same. This approach cause poorly accurate predictions due to information not being up-to-date derived from the existing gap between training- and testing-window.

- *sliding*, where the training-window will have a predetermined size and will move alongside with the testing-window. This approach guarantees that the last training point is adjacent to the first testing point, thus mitigating the aforementioned drawback.

- *incremental*, where training-window will start as in the fixed approach, however growing by consequent inclusion of predicted-windows. It is important to mention that new training included will have the true values and not the predicted ones.

We opted for a *sliding* training-window as the *fixed* training-window would under-perform due to aforementioned reasons and the *incremental* training-window being ruled out due to memory constraint.

The window size chosen was 10, meaning that when predicting at $t$ time-step we will use information from $t-10$ to $t$ time-steps. For us, the underlying rational was to use the previous 10 time-steps in order to predict the 10-days returns movement, our target variable. Intuitively, in order to create a training-window for a specific asset it is required for that particular asset to have at least 9 more days of history. As it was verified that some assets did not verify this constraint we opted by removing those assets.

After joining both data sources according to the above described process, the new dataset was split into 2007–2014 for training, 2015 for validation, and 2016 for testing. Therefore, ending with $\sim$ 2.760.000 samples for training, $\sim$ 578.000 samples for validation and $\sim$ 577.000 samples for testing.

As for the target variable, the competition used the open-to-open market-residualized return over a 10-day period into the future. In our experiments, and following common practice in the literature reviews, we used only the directional movement (up or down) of this target variable. Therefore, turning our problem into a binary classification task.

Preprocessing was implemented in Python (`https://www.python.org`) relying on packages such as numpy [226] (`https://numpy.org`), scikit-learn [227] (`https://scikit-learn.org/stable`) and pandas [228] (`https://pandas.pydata.org`).

## 4.4 Model Input

In the previous section it was detailed how data was preprocessed and the different approaches we chose to fill *Null* values. This resulted in four different datasets, namely:

- *Market only*, containing data solely retrieved from the market dataset;
- *Market+News propagated*, using the same data as the *Market only* dataset plus the sentiment signals propagated throughout time;
- *Market+News balanced*, using the same data as the *Market only* dataset plus sentiment signals with $\frac{1}{3}$ probability for each class when no news exist;
- *Market+News neutralized*, using the same data as the *Market only* dataset plus sentiment signals with 1.0 probability to the neutral class and 0.0 probability to the remaining sentiment classes when no news exist;

The way the categorical and numerical features have been combined in order to be provided as input to a model is illustrated in Fig. 4.2. This is the general approached used for all models with the exception of Feed-Forward Neural Network (FNN) and the Transformer-like models.

In essence, the categorical features go through an embedding layer before being concatenated together with the numerical features. We have chosen to use an embedding layer instead of other approaches such as a one-hot encoding vector or similar alternatives due to the dimensionality present in the categorical features. As described in Section 4.2, there are $\sim$ 3000 companies, which means that there are $\sim$ 3000 asset names and even more asset codes to represent, as one asset name can have multiple asset codes. This characteristic would lead to an enormous number of dimensions if we have chosen a one-hot encoding approach to represent the categorical features, hence the embedding layer. The embedding layer used has an input dimension with the same size as the number of different values existent in the categorical feature and represents the corresponding feature into a three-dimensional output.

As presented in Fig. 4.2, the general input block provides an input of 10 time-steps at a time for all models. The only exception being Feed-Forward Neural Network (FNN) models which use a single time-step, as represented in Fig. 4.3. Regarding Fig. 4.2 and Fig. 4.3, there is an additional relevant difference to highlight: Fig. 4.2 input is the *Market+News* dataset constituted by market and sentiment features, whereas Fig. 4.3 input is solely composed by market features, hence from *Market only* dataset. The difference is the number of features fed to the *numerical_input* layer.

Concerning the Transformer-like model, Fig. 4.4 details how the 10 time-steps input is fed to this model. To allow parallel computation, after the categorical and numerical features are concatenated the data is reshaped, hence allowing all time-steps to be processed at the same time.

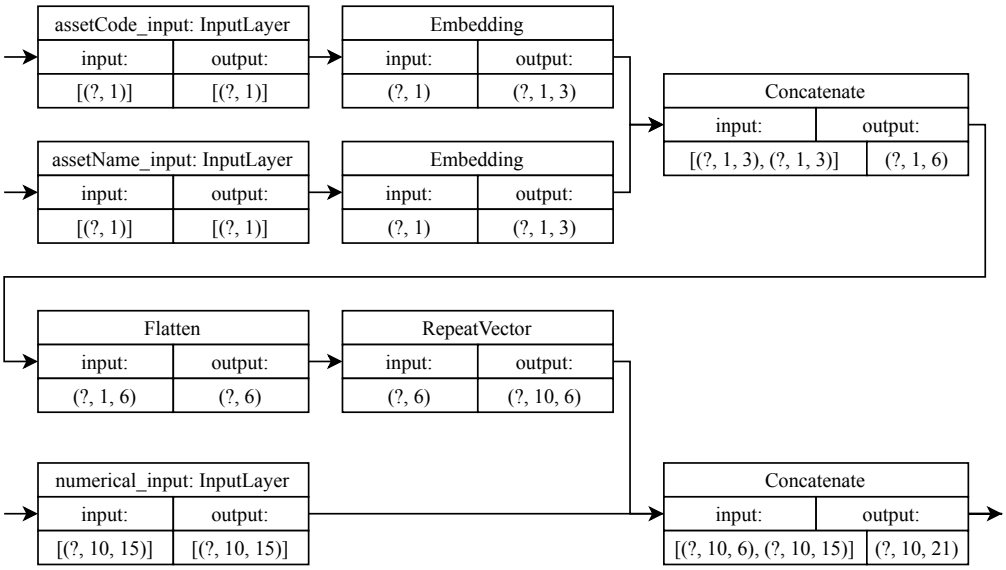



Figure 4.2: General Input block.

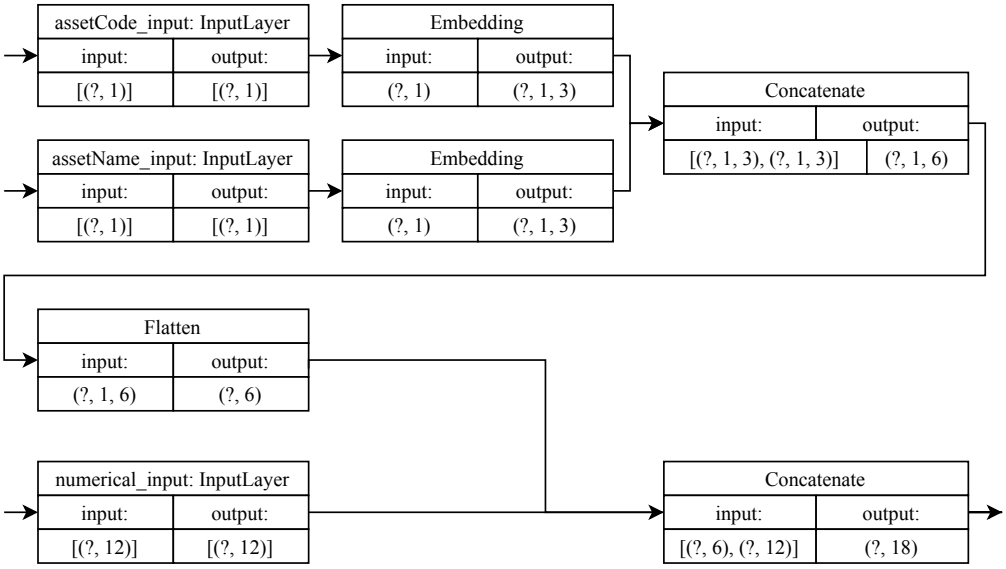



Figure 4.3: FNN Input block.

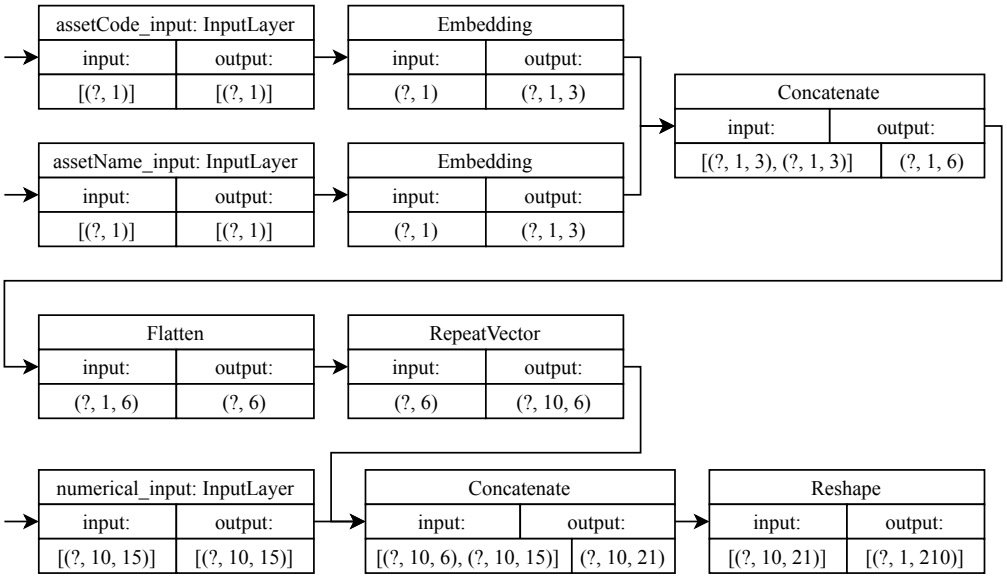



Figure 4.4: Transformer Input block.

## 4.5 Model Architecture

After detailing the input format, this section introduces the architectural variations and specificities of each model. To compare each set of features and different models performance, an extensive experiment was conducted where several hyper-parameters were varied.

The number of layers that compose each model have been alternated between 1, 2 and 4. As well as the number of neurons forming each layer, starting from the number of input features to 100 neurons. In bidirectional Long Short-Term Memory (LSTM) models case until 200 neurons. Regarding neurons, we tested two approaches. Firstly, we kept the number of neurons constant throughout layers and, secondly, we reduced the number of neurons present in each layer. The reduction was defined by dividing the original number of neurons by the cardinality of each layer, i.e. 1, 2, 3 or 4.

All models were trained through back-propagation in combination with the Adam optimization algorithm. We also applied the binary cross-entropy as loss function to be optimized.

In order to control overfitting and improve the generalization capabilities of the models, we decided to also use dropout regularization layers. This is a technique where neurons are randomly dropped from a layer during training, i.e. each neuron has a fixed probability $p$ independent of other neurons to be temporarily removed from the network during training. Therefore, allowing to prevent neurons from co-adapting too much [229]. We started by using no dropout and then included dropout layers with probabilities of 0.2 and 0.5.

In addition, we varied the training batch size. A batch is the number of samples that a model trains with before updating its internal parameters, aforementioned referred as weights in Section 3.1. For FNN, RNN, LSTM and bidirectional LSTM models, training batches varied between 1% and 10%. For CNN and Transformer-like models a training batch of 1% was used. Due to memory restrictions, LSTM and bidirectional LSTM models with Attention used a 0.1% training batch.

Also, we have trained all models for several epochs. An epoch is the total number of times that the

training dataset will be passed into the model in order to adjust its weights. As too few training epochs can lead to a model underfitting, i.e. a model unable to generalize well by not learning enough patterns from data, the models were set to have a training of 5000 epochs. As too many training epochs can lead to a model overfitting, i.e. a model focus on a specificity of the data and over specializes also leading to poor performance, the models also employed Early Stopping on validation loss with a patience of 10 epochs. This allows to stop training once the model validation loss stops decreasing for 10 epochs in a row.

In order to provide a more detailed view, some additional comments are necessary regarding CNN and Transformer-like models.

As above explained, when experimenting with CNN models we vary the layers between 1, 2 and 4, and used a batch of 1% of the training dataset. Additionally, in this model other hyper-parameters were tested. For instance, we changed the kernel size value between 2, 3 and 5, and the number of filters between 2, 3, 5, 10, the number of features in the input and 100.

Regarding the Transformer-like model, we decided to alter the $d_{model}$ and experiment with a different number of heads and layers. When observing aforementioned explained Eq. 3.14 and 3.15, there are two hyper-parameters $pos$ and $d_{model}$. We decided to maintain the value of $pos$ as the original, since it does not have a direct correlation with the size of the input. On the contrary, we altered $d_{model}$ value. This hyper-parameter was experimented in the original paper [222], and was defined because typical NLP tasks performed with the Transformer model were of variable and unknown size. $d_{model}$ in the original paper was defined with a value of 512, hence allowing this number of words to be inputted. This value guarantees a good performance and at the same time an extensive number of inputs in parallel. In our task, as the input size is fixed to $\#features \times 10$, we chose to alter $d_{model}$ value to our input size. Other hyper-parameters, such as the number of neurons in the FNN layer, were kept untouched.

However, there is a constraint defined in the model stating that the dimension of the model ($d_{model}$) must be divisible by the number of Attention heads the model incorporates (Eq. (4.1)). Therefore, we were restricted to experiment the number of heads with the values 5, 6, 7, 9, 10 and 14. Values divisible by the input size. We also varied the stack of $N$ components between 2, 4, 6 and 8.

$$\frac{d_{model}}{\#heads} == 0 \tag{4.1}$$

Prediction Models were, as the Preprocessing, implemented in Python (`https://www.python.org`) using the Keras API on top of TensorFlow (`https://www.tensorflow.org`). The experiments were conducted in Google Colab (`https://colab.research.google.com`) and Jupyter Notebooks (`https://jupyter.org`).

## 4.6 Evaluation Metrics

In order to be able to compare either different models or different datasets performance an evaluation metric must be chosen. Some papers of Table 2.4 on page 16, due to being regression tasks, determine

the difference between predicted and actual values, hence consequently using metrics that do not apply to our problem such as Mean Squared Error (MSE), Root MSE, Mean Absolute Percent Error (MAPE), Cosine Proximity or similar.

As our target is to predict direction (upwards or downwards) of Stock Prediction, this work is a classification task to predict the right "class" of the test set instances: True, if our system predicts a down or up trend and in fact the stock price went down or up, correspondingly; False, if it went in the opposite direction as the predicted one. The comparison of models performance will be done with the metric on Eq. (4.2), which describes the percentage of correct predictions against all predictions made. This metric is commonly referred as Accuracy.

$$\text{Accuracy} = \frac{\text{\# of correct predictions}}{\text{\# of predictions}} \times 100 = \frac{\text{TP + TN}}{\text{TP + TN + FP + FN}} \times 100 \tag{4.2}$$

In this context, a correct prediction is an agreement between the model output (an up or down prediction) and the actual market trend. This turns out to be equivalent to binary accuracy as in a binary classification problem, which is one the most common approaches in the literature.

As when preprocessing our target variable we labeled down trends as 0 and up trends as 1, a True Positive (TP) is when a model predicts an up trend correctly, and a True Negative (TN) is when a model predicts a down trend correctly. False Positive (FP) and False Negative (FN) cases occur when a model predicts a trend contrary to the one that actually occurred, downwards and upwards correspondingly. Therefore, Eq. (4.2) can be extended to the formula on the left part.

Accuracy is the measure chosen to determine which architecture variation performs best within each Neural Network (NN) type of model. Nonetheless, additional measures will be used to compare each type of NN model confidence when predicting both positive and negative classes. Matthews Correlation Coefficient (MCC) and Area Under the ROC Curve (AUC) are the main measures used to make more complete comparisons.

MCC is useful because it only generates high scores if the model is able to predict both high percentages of TP and TN, as can be perceived from Eq. 4.3. MCC ranges from -1, an inverse prediction to 1, a perfect prediction. A value of 0 corresponds to an average random prediction.

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP + FP})(\text{TP + FN})(\text{TN+FP})(\text{TN+FN})}} \tag{4.3}$$

The AUC is a metric that measures the True Positive Rate against the False Positive Rate. The greater the AUC value the more number of instances are correctly classified, hence higher the number of TP and TN.

More fine-grained metrics such as Recall, Precision, Specificity and Negative Predictive Value (NPV) will not be analyzed by themselves as they focus solely on Positive or Negative instances. These are helpful metrics when one class has more importance than the other, e.g. identifying card fraud or terrorists faces. As down and up trends have equal importance, it is our opinion that these metrics of comparison alone do not add any additional benefit. We will only briefly comment and compare on each model ability to better identify a trend, and the confidence on that identification when predicting both

upwards or downwards trends.

In essence, as shown in Eq. 4.4, Recall, also commonly referred as Sensitivity or True Positive Rate, is the ratio between the correct number of up trends predicted and the total number of up trends, even if they were incorrectly identified as a down trend (i.e. measures the model accuracy on classifying up trends).

$$\text{Recall} = \text{Sensitivity} = \text{TPR} = \frac{\text{TP}}{\text{TP + FN}} \tag{4.4}$$

Precision, as shown in Eq. 4.5, also commonly referred as Positive Predictive Value (PPV), is the ratio formed by the number of correctly up trends divided by the total predictions of up trends. In other words, Precision measures the hit accuracy when identifying up trends (i.e. from all classified labels as up trends the ones that actually are up trends).

$$\text{Precision} = \frac{\text{TP}}{\text{TP + FP}} \tag{4.5}$$

A complementary remark about the up trend classifying metrics is that Precision can be improved at the expense of Recall. Therefore, in order to equally weight Precision and Recall the F1-measure will also be computed (Eq. 4.6).

$$\text{F}_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.6}$$

Specificity, as shown in Eq. 4.7, is the ratio formed by the number of correctly predicted down trends and the total number of down trends existing, even if they were identified as an up trend. It gives an overview similar to Recall for down trends.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN + FP}} \tag{4.7}$$

NPV, as shown in Eq. 4.8, is the ratio formed by the number of correctly predicted down trends divided by the total predictions of down trends. In other words, NPV measures the hit accuracy when identifying down trends (i.e. from all classified labels as down trends the ones that actually are down trends). It gives an overview similar to Precision for down trends.

$$\text{NPV} = \frac{\text{TN}}{\text{TN + FN}} \tag{4.8}$$

Note that all metrics referred evaluate model performance after training. However, as explained in Section 3.1, a loss function is required to train the models. Since none of the above metrics are differentiable, a differentiable loss function must be selected. The typical choice for classification problems is to choose a loss function based on some form of cross-entropy, since this is more effective in punishing misclassification, where the loss value tends to infinity. In contrast, distance-based measures, such as MSE or similar, are specifically devised for regression tasks, and should not be employed here. Since what we have is a binary classification problem, we will use binary cross-entropy (BCE) as loss function:

$$\text{BCE} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log p_i + (1-y_i)\log(1-p_i)\right] \tag{4.9}$$

In the formula above, $N$ is the number of training samples, $y_i \in \{0, 1\}$ is the true label of sample $i$, and $p_i \in [0, 1]$ is the corresponding prediction produced by the model.

In the next chapter we will present the results obtained with the data described in Section 4.1, with the architectural variations detailed in Section 4.5 in order to find the best performing model with the metrics presented in this section.

# Chapter 5

# Experimental Results

In this chapter, we will detail the results obtained from the proposed approach in previous Chapter 4. For each Neural Network (NN) model we detail its best architecture and discuss the corresponding hyper-parameters. In particular, we will also analyze the impact of introducing both recurrent and attention mechanisms.

As detailed in Section 4.5, we have experimented several architectural variations for each type of NN model, namely: FNN (Section 5.1), CNN (Section 5.2), RNN (Section 5.3), LSTM (Section 5.4), bidirectional LSTM (Section 5.5), LSTM with Attention (Section 5.6), bidirectional LSTM with Attention (Section 5.7), and Transformer (Section 5.8). To conclude this chapter, Section 5.9 provides a comparison aggregating all models and best performances where more broader conclusions are drawn.

## 5.1 Feed-Forward Neural Network

In the first experiment with a simple FNN model (Fig. 3.2), we tested 48 combinations of hyper-parameters. Having each model taken in average between 1 and 2 minutes to train. As we had 4 different datasets to experiment our models with, the training process took about 4 hours in order to obtain results.

As mentioned in the previous chapter, we varied training batches from 10% to 1%, experimented architectures with a different number of layers (1,2 and 4) with no dropout and with dropout rates of 0.2 and 0.5. Additionally, we used several different quantities of neurons in each layer, as described in Section 4.5.

In Table 5.1 are presented the architectures with the best performance, i.e. the ones achieving the highest accuracy. In order to better understand the table, a notation explanation is required: $\downarrow$ represents the fact that the number of neurons keeps decreasing across layers, and = represents the fact that it is kept constant across layers. When the number of neurons decreases, it corresponds to the number of input features divided by a factor of 1, 2, 3, 4, etc. When it is kept constant, it is equal to the number of input features. The underline highlights the highest accuracy achieved for a particular dataset and the bold highlights the best accuracy obtained with a FNN model overall. This terminology will be used

Table 5.1: Accuracy comparison of FNN models.

| FNN | Market only | Market + News propagated | Market + News balanced | Market + News neutralized |
|---|---|---|---|---|
| **4 layers ↓ neurons** | 0.501 | 0.532 | 0.531 | 0.532 |
| **4 layers 100 neurons 0.2 dropout** | **0.533** | 0.501 | **0.533** | **0.533** |

↓ represents the number of neurons decreasing across layers.

throughout the whole chapter when presenting results.

By looking at Table 5.1, this experiment already provides an interesting conclusion: when using an FNN, the sentiment features do not have any influence in the output predictions As can be observed, the same model using three different datasets obtained the same accuracy. In our view, this is due to FNN models do not incorporate any information on past behavior. Therefore, this type of model is unable to better leverage the sentiment history present. Both with and without the sentiment features, this model scored a test accuracy of 53.3%.

Additionally, another conclusion is possible to draw. As can be perceptible in Table 5.1, more complex models have better results. Having all the best performing FNN models a 4 layers architecture and a training batch of 1%.

Fig. 5.1 details the best FNN architecture. This is composed by 4 layers each formed by 100 neurons. A training batch of 1% is used and a 0.2 dropout rate was employed. This model achieved a 53.3% accuracy.

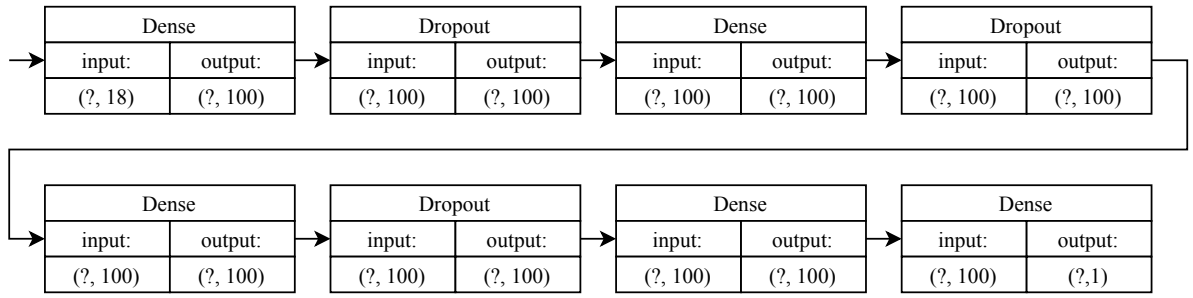



Figure 5.1: Best FNN architecture.

Fig. 5.2 details the loss function performance and corresponding accuracy throughout training. This model achieved a Matthews Correlation Coefficient (MCC) of 0.070 and an Area Under the ROC Curve (AUC) of 0.532. Related with up trends the results of additional measures were a Recall of 0.728, a Precision of 0.524 and a $F_1$Score of 0.609. For down trends the FNN model obtained a Specificity of 0.336 and Negative Predictive Value (NPV) of 0.552.

As our dataset is balanced, having approximately the same number of up and down trends, the above presented metrics allow us to conclude that a FNN architecture can considerably better identify up trends (72.8%) than down trends (33.6%). In fact, this particular FNN model scored the highest Precision from all NN models experimented, reaching the highest $F_1$Score as well. In our opinion, this is caused by the stock market being in constant growth throughout recent years combined with this model inability to capture market reversions, hence predicting more uptrends than downtrends. When comparing certainty on classification, this is very similar. 52.4% on up trends versus 55.2% on down trends.

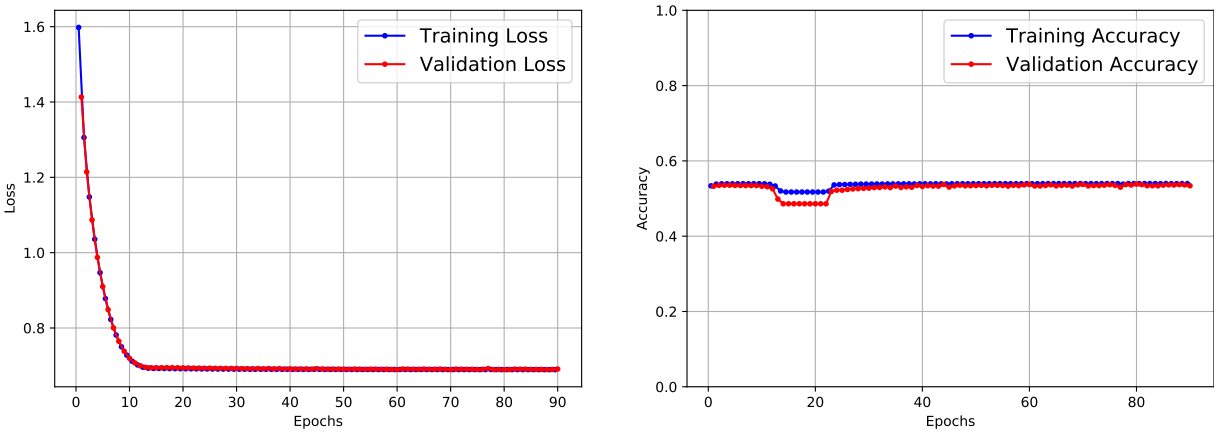



Figure 5.2: FNN performance during training.

## 5.2 Convolutional Neural Network

Using CNN models (Fig. 3.3) we tested 42 different architectural variations. As mentioned in Section 4.5, in addition to varying the number of layers between 1,2 and 4 layers, we have also varied the kernel size between 2, 3 and 5, and the number of filters between 2, 3, 5, 10, the number of features in the input and 100. With CNN models we did not varied the batch size, having always used a training batch of 1%. Each model took between 5 and 15 minutes to run, having the 4 datasets taken around 2 and half days to be tested.

CNN models, by using a 10 time-step window into the past, improved the prediction accuracy. When analyzing Table 5.2, it is possible to see a 1.8% accuracy increase if we compare the best CNN model against the best FNN model.

Table 5.2: Accuracy comparison of CNN models.

| CNN | Market only | Market + News propagated | Market + News balanced | Market + News neutralized |
|---|---|---|---|---|
| **4 layers 100 filters kernel=2** | **0.551** | 0.538 | 0.540 | 0.548 |
| **1 layers 100 filters kernel=3** | 0.539 | 0.539 | 0.540 | 0.539 |
| **2 layers 10 filters kernel=3** | 0.538 | 0.540 | 0.538 | 0.535 |
| **2 layers 100 filters kernel=3** | 0.539 | 0.539 | 0.540 | 0.539 |
| **2 layers 10 filters kernel=5** | 0.538 | 0.538 | 0.541 | 0.537 |
| **2 layers # features filters kernel=5** | 0.538 | 0.538 | 0.540 | 0.537 |
| **2 layers 100 filters kernel=5** | 0.538 | 0.543 | 0.543 | 0.542 |

In addition, by observing the architectures detailed in Table 5.2, it is possible to draw the same conclusion as in previous Section 5.1 with FNN models. The best performing CNN model has a 4 layers architecture. In addition, it uses 100 filters and a kernel size of 2. This model achieved an accuracy of 55.1%. However, the test accuracy was higher without the sentiment features (55.1%) than with those features included (54.8%). We attribute this fact to overfitting, and dropout did not help in this case.

More detailed information about this model architecture can be observed in Fig 5.3. This model, using the *Market only* dataset, achieved a 0.097 MCC score, a 0.547 AUC and 0.591 $F_1$Score. Due to additional measures used, we can conclude that the model can better identify up trends than down trends having a Recall of 0.649 against a Specificity of 0.446. Regarding certainty on classification, the

model has a Precision of 0.542 and a NPV of 0.557.

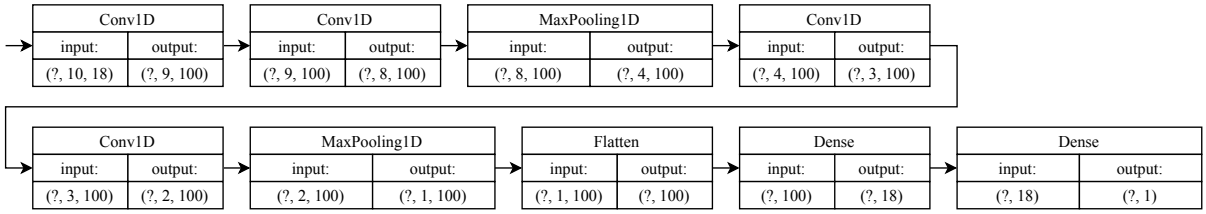



Figure 5.3: Best CNN architecture.

Fig. 5.4 details the loss function performance and corresponding accuracy throughout training.

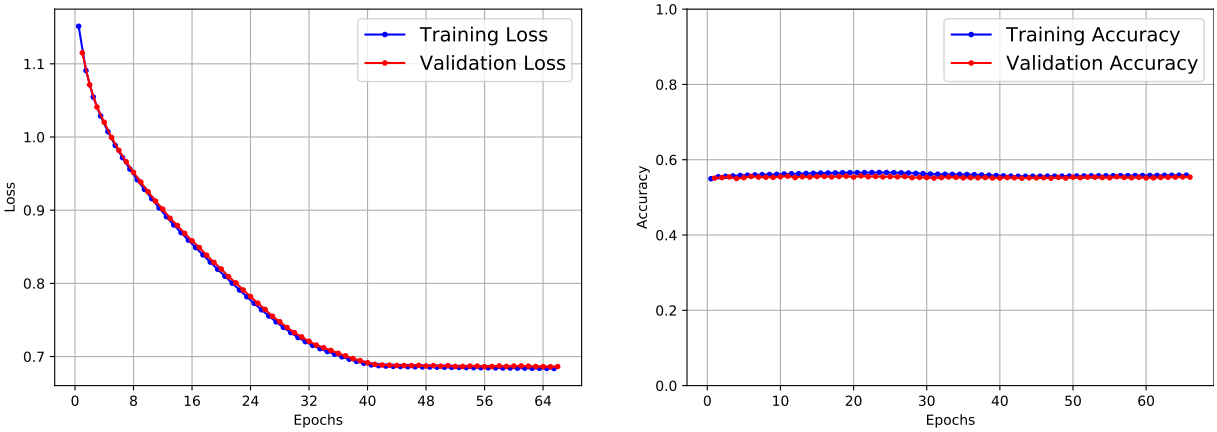



Figure 5.4: CNN performance during training.

## 5.3 Recurrent Neural Network

For RNN models we have made a similar evaluation as the one detailed FNN. The number of layers, neurons, training batches and dropout rates were varied in the same way as detailed in Section 5.1. With this type of NN (Fig. 3.4) we have tested 39 different sets of hyper-parameters. Each variation took around 40 minutes to train, having the 4 datasets results been determined after around 4 days from the starting of the RNN models experiment.

Table 5.3: Accuracy comparison of RNN models.

| RNN | Market only | Market + News propagated | Market + News balanced | Market + News neutralized |
|---|---|---|---|---|
| **2 layers ↓ neurons** | 0.545 | 0.547 | **0.552** | 0.550 |
| **4 layers ↓ neurons** | 0.546 | 0.541 | 0.545 | 0.537 |
| **4 layers 100 neurons** | **0.552** | 0.538 | 0.546 | 0.545 |
| **4 layers 100 neurons 0.2 dropout** | 0.541 | 0.539 | 0.541 | 0.541 |

↓ represents the number of neurons decreasing across layers.

RNN is the first model in our experiment employing recurrence mechanisms. As can be noted in Table 5.3, when we employ these mechanisms into our models a subtle improvement in accuracy occurs, having the best RNN model outperformed the best CNN model by 0.1%. When we make use of a 10

time-step window jointly with recurrence mechanisms we can observe a more noticeable improvement. The RNN model outperformed all FNN models by at least 1.9%.

The RNN model achieving the highest accuracy has a 2 layer architecture, with no dropout and a training batch of 1%. Again, all best performing models used a training batch of 1%. Regarding neurons, the quantity is decreasing throughout layers with the approach described in Section 5.1. This model achieved an accuracy of 55.2%, and is described with more detailed in Fig 5.5.

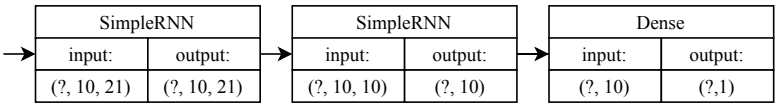



Figure 5.5: Best RNN architecture.

Fig. 5.6 details the loss function performance and corresponding accuracy throughout training of the RNN model detailed in Fig 5.5. This model achieved an MCC of 0.103, an AUC of 0.552, Recall of 0.581, Precision of 0.551, $F_1$Score of 0.566, Specificity of 0.522 and NPV of 0.552.

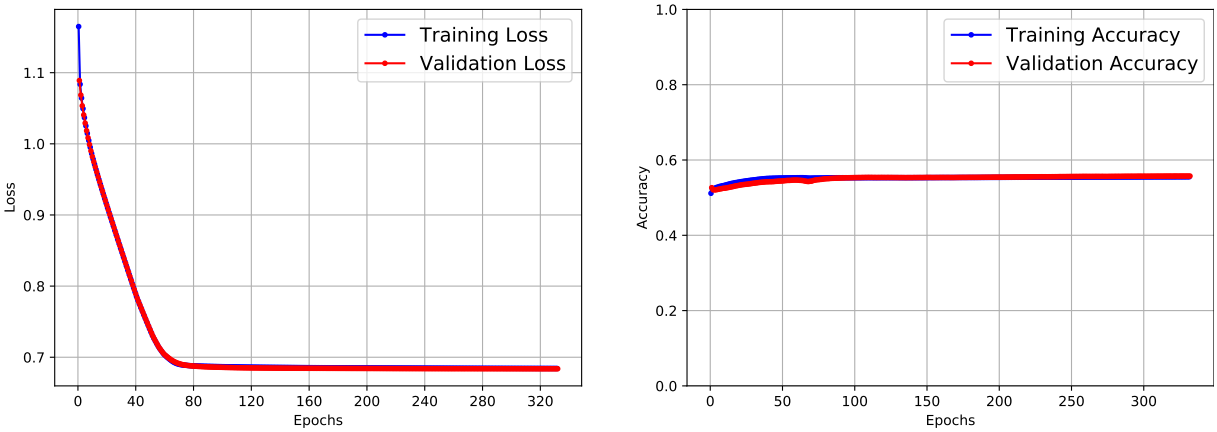



Figure 5.6: RNN performance during training.

Contrary to the previous models, this model's Recall is rather similar to its Specificity, leading us to conclude that RNN models identify down trends almost as well as up trends. It is also noticeable that the capability to identify up trends has been descending (Recall ratio decline from FNN to RNN), while the capability to identify down trends has been improving (Specificity ratio rising from FNN to RNN). In our opinion, mainly due to the introduction of recurrence mechanisms, which enable better identification of market reversions.

Regarding certainty when identifying both up and down trends, we can notice that it has been significantly rising for up trends and remaining constant for down trends.

## 5.4 Long Short-Term Memory

LSTM models (Fig. 3.5) experimental approach was the same as the one employed in previous models experiments. The number of layers, neurons, batches and dropout rates was varied in the same way for both FNN, RNN and LSTM models. These models training took longer as the neurons composing their

layers are more complex, hence requiring more computational time. It lasted 5 days.

As can be noted in Table 5.4, when we employ a more powerful recurrence mechanisms into our models, an accuracy improvement of 0.1% is noted. Although very modest, when comparing against RNN models, accuracy improved from 55.2% to 55.3% when the sentiment features were included.

Table 5.4: Accuracy comparison of LSTM models.

| LSTM | Market only | Market + News propagated | Market + News balanced | Market + News neutralized |
|---|---|---|---|---|
| **1 layer** | 0.551 | 0.545 | 0.542 | 0.542 |
| **4 layers ↓ neurons** | $\underline{0.552}$ | 0.540 | 0.542 | 0.540 |
| **4 layers 100 neurons 0.2 dropout** | 0.543 | $\underline{0.551}$ | **0.553** | $\underline{0.551}$ |
| **4 layers 100 neurons 0.2 dropout** | $\underline{0.552}$ | 0.539 | 0.547 | 0.547 |

↓ represents the number of neurons decreasing across layers.

As in FNN and CNN experiments, we observe that one of our more complex LSTM models trained achieved the best score. Intuitively, this leaves the following open question: if we increase the number of layers in this type of models the accuracy increase as well? Unfortunately, due to memory constraints, we are unable to provide an informed answer.

Another particularity when comparing FNN, RNN and LSTM models training is that almost all best performing models use a batch of 1%. In our opinion, this is due to more optimizations being done in the same amount of epochs, hence the model generalizes better and accuracy increases. On the contrary, with a larger batch (10%) the model training significantly degrades its quality, hence affecting the model ability to generalize.

The LSTM model achieving the highest accuracy has its architecture detailed in Fig 5.7. The model has a 4 layer architecture, with 100 neurons per layer, makes use of a dropout layer with a 0.2 rate and uses a batch of 1%. This model achieved an accuracy of 55.3%.

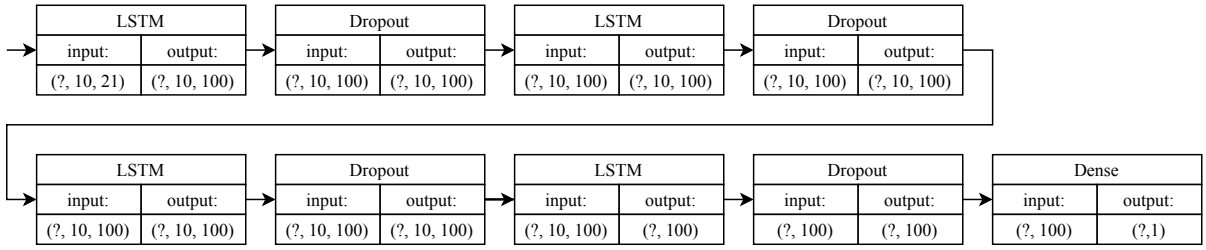



Figure 5.7: Best LSTM architecture.

Lastly, Fig. 5.8 details the loss function performance and corresponding accuracy throughout training of the LSTM model detailed in Fig 5.7. This model achieved an MCC of 0.109, an AUC of 0.553, Recall of 0.673, Precision of 0.545, $F_1$Score of 0.603, Specificity of 0.433 and NPV of 0.567.

When comparing this model with the previous RNN model, we observe that the LSTM better identifies up trends (67.3% vs 58.1%). However, its performance degrades when identifying down trends (43.3% vs 52.2%). Regarding certainty when identifying trends, the confidence remains more or less the same, although LSTM have a slight improvement when classifying down trends. In our opinion, due to the fact of worst deciding when a trend is downwards. The LSTM better identification up trends cause them to have an higher $F_1$Score when compared to RNN (0.603 versus 0.566).


Figure 5.8: LSTM performance during training.

## 5.5 Bidirectional LSTM

In this section we detail the experiments done with bidirectional LSTM models. This model is very similar to a standard LSTM, however with twice the neurons per layer given the same hyper-parameters. The bidirectionality allows the models to analyze inputs in both directions, forwards and backwards.

The same hyper-parameter and architectural variations were made with bidirectional LSTM as with previously discussed RNN and LSTM models, and as detailed in Section 5.1 for FNN models. However, due to the number of neurons present in each layer, a memory limit was reached. As a result, when training, a 0.1% batch was used for all architectures. We have tested 25 hyper-parameter and architectural variations with bidirectional LSTM models, having this experience lasted for about 5 days. The best results are detailed in Table 5.5.

Table 5.5: Accuracy comparison of bidirectional LSTM models.

| Bidirectional LSTM | Market only | Market + News propagated | Market + News balanced | Market + News neutralized |
|---|---|---|---|---|
| **1 layer 100 neurons** | 0.542 | 0.543 | 0.543 | 0.548 |
| **1 layer 100 neurons 0.2 dropout** | 0.541 | 0.540 | 0.544 | 0.547 |
| **2 layers ↓ neurons** | 0.543 | 0.541 | 0.545 | 0.542 |
| **2 layers ↓ neurons 0.2 dropout** | 0.542 | 0.541 | 0.542 | 0.540 |
| **2 layers ↓ neurons 0.5 dropout** | 0.542 | 0.541 | 0.540 | 0.540 |
| **4 layers ↓ neurons** | 0.541 | 0.541 | 0.540 | 0.539 |
| **4 layers ↓ neurons 0.2 dropout** | 0.541 | 0.548 | 0.540 | **0.549** |
| **4 layers ↓ neurons 0.5 dropout** | 0.541 | 0.541 | 0.547 | 0.547 |

↓ represents the number of neurons decreasing across layers.

In the presented table is possible to notice that simpler models tend to perform well. In our opinion, due to each layer having twice the neurons when compared with previous approaches, hence being able to represent more complex relations with simpler models. In this case, it is also perceptible that dropout has a bigger effect than with previous models, hence helping to prevent overfit in simpler models and increasing their accuracy. A great number of models made use of dropout, especially with a 0.2 probability rate.

Nonetheless, although we have employed an extra direction of analysis to our input, therefore theoretically enabling this model to better represent more complex relationships, bidirectional LSTM per-

formance was worst when compared to aforementioned recurrent models. We justified this due to the overfit verified on more complex models, particularly the ones using 200 neurons per layer.

Fig. 5.9 details the best performing bidirectional LSTM architecture achieving an accuracy of 54.9%. This model is composed by a 4 layers architecture, containing each layer a decreasing number of neurons (as explained in Section 5.1) and leveraging a dropout rate of 0.2. The training batch size used was of 0.1% due to memory constraints.

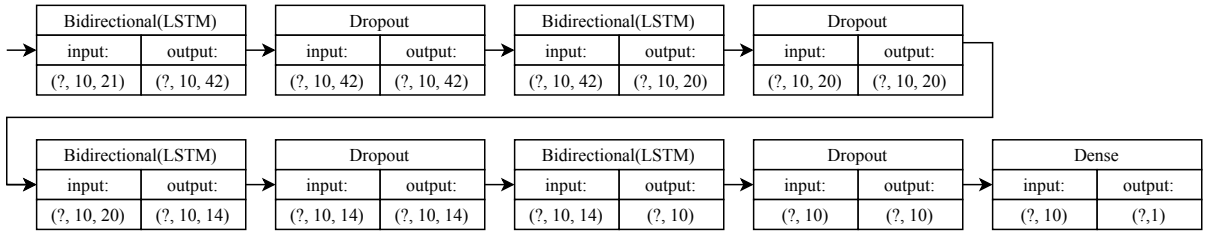



Figure 5.9: Best bidirectional LSTM architecture.

Fig. 5.10 details the loss function performance and corresponding accuracy throughout training of the bidirectional LSTM model detailed in Fig 5.9. This model achieved an MCC of 0.098, an AUC of 0.548, Recall of 0.627, Precision of 0.545, $F_1$Score of 0.583, Specificity of 0.470 and NPV of 0.555. When compared to an LSTM, bidirectional LSTM most accurate model better identifies down trends (47.0% versus 43.3%). However, it performs worst when identifying up trends (52.7% versus 67.3%).

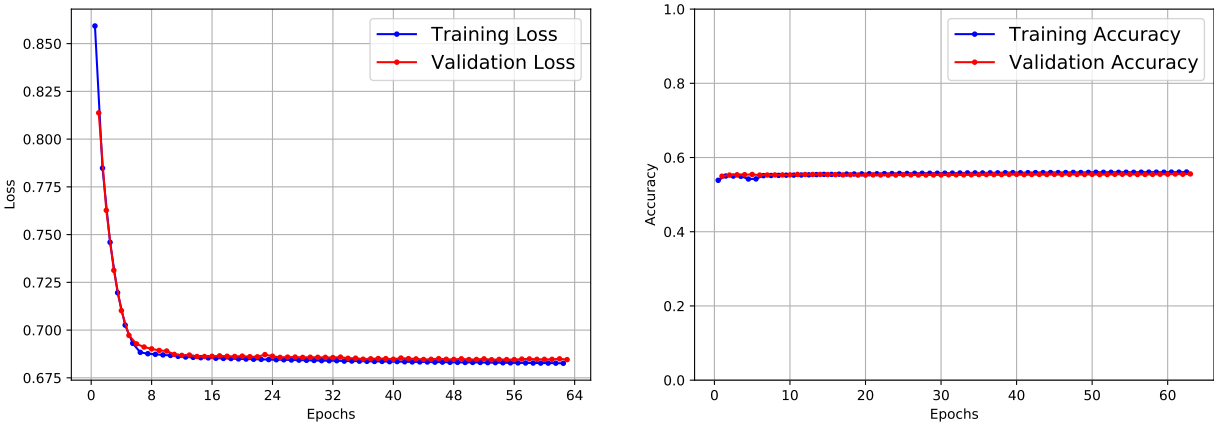



Figure 5.10: Bidirectional LSTM performance during training.

We conclude that recurrence mechanisms are essential in order to leverage information from the past. However, a bidirectional LSTM did not improve the results, which is understandable, since the most recent history is probably more relevant for prediction than the distant past, so only the forward direction is useful.

## 5.6 LSTM with Attention

In the following sections will employ the Attention Mechanisms previously detailed in Section 3.3. The present Section 5.6 and the following Section 5.7 will make use of Sequential Attention mechanisms,

where an Additive alignment score function is used. Section 5.8 will experiment a recently proposed model, where a Parallel Attention mechanism is the core of predictions.

In this section we detail the experiments using LSTM models with an Additive Attention mechanism. This model is very similar to a standard LSTM, however it has attached at the end of the LSTM layers an attention mechanism to increase focus on specific parts of the input. This mechanism allows to selectively highlight the most relevant features of the input and give them more importance when predicting a trend. A more detailed explanation was already provided in Section 3.3.

The same hyper-parameter and architectural variations were made with this kind of NN as the ones discussed in previous sections and as detailed in Section 4.5. We have tested 25 hyper-parameter and architectural variations with LSTM with Attention models, having this experience lasted for about 6 days. However, as this model employs an extra layer (Attention layer), a memory limit was reached. Therefore, when training these models, as it happened with bidirectional LSTM experiments, the different architectures solely used a 0.1% training batch.

As can be observed in Table 5.6, the addition of an attention layer to recurrence mechanisms and to a 10 time-step window dramatically increased accuracy. Having the LSTM with Attention outperformed all models without Attention by at least 4.7%.

Table 5.6: Accuracy comparison of LSTM models with Attention.

| LSTM with Attention | Market only | Market + News propagated | Market + News balanced | Market + News neutralized |
|---|---|---|---|---|
| **1 layer** | 0.503 | 0.594 | 0.503 | 0.503 |
| **2 layers ↓ neurons** | 0.503 | $\underline{0.599}$ | 0.598 | 0.598 |
| **2 layers ↓ neurons 0.2 dropout** | $\underline{0.598}$ | 0.595 | $\underline{0.600}$ | 0.592 |
| **2 layers = neurons** | 0.595 | 0.594 | 0.596 | 0.590 |
| **2 layers = neurons 0.2 dropout** | 0.503 | 0.592 | 0.503 | 0.503 |
| **4 layers ↓ neurons** | 0.503 | 0.503 | 0.592 | 0.507 |
| **4 layers ↓ neurons 0.2 dropout** | 0.503 | 0.503 | 0.581 | 0.578 |
| **4 layers = neurons** | 0.503 | 0.596 | 0.599 | 0.596 |
| **4 layers = neurons 0.2 dropout** | 0.590 | 0.593 | 0.591 | **0.600** |
| **4 layers 100 neurons** | 0.591 | 0.588 | 0.593 | 0.593 |

↓ represents the number of neurons decreasing across layers.
= represents the number of neurons being kept constant across layers.

In addition, it is possible to perceive that simpler models leveraging attention mechanisms also perform well, particularly the 2 layers architectures. Furthermore, as a lot of computation is performed when training this type of model, we can also observe that dropout has a key role.

Fig. 5.11 details the best performing LSTM with Attention architecture. This model is composed by 4 layers, each of the layers containing the same number of neurons as the number of input features. Additionally, this model also makes use of a 0.2 dropout rate. The batch during training as aforementioned explained is 0.1% of the training data. An LSTM model with additive attention achieves 59.8% accuracy without sentiment features, and reaches 60.0% when those features are included.

Fig. 5.12 details the loss function performance and corresponding accuracy throughout training of the highest accuracy achieving LSTM with Attention model detailed in Fig 5.11. Another model has also attained the same accuracy result however, the detailed model achieved an higher MCC (0.199 versus 0.197). Therefore, being selected as the best performing model. This model also achieved an AUC of

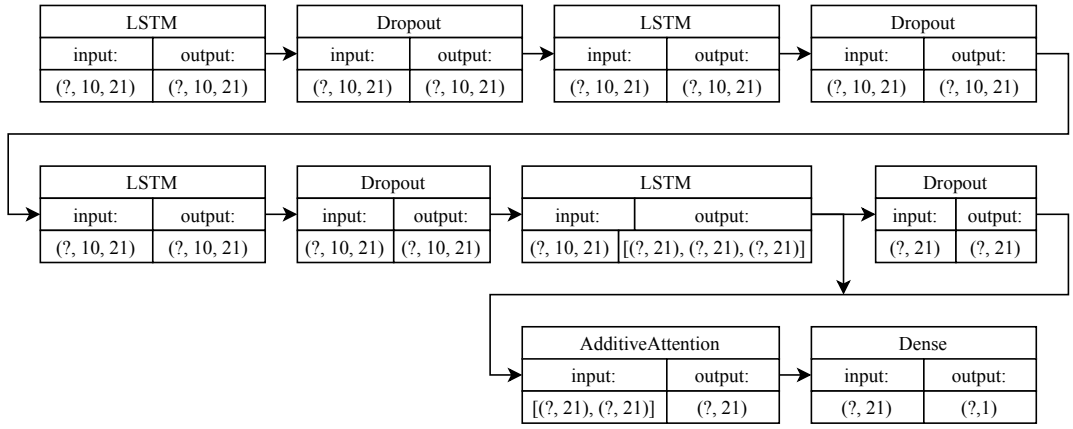



Figure 5.11: Best LSTM with Attention architecture.

0.600, a Recall of 0.612, a Precision of 0.600, a $F_1$Score of 0.606, a Specificity of 0.588 and a NPV of 0.600.

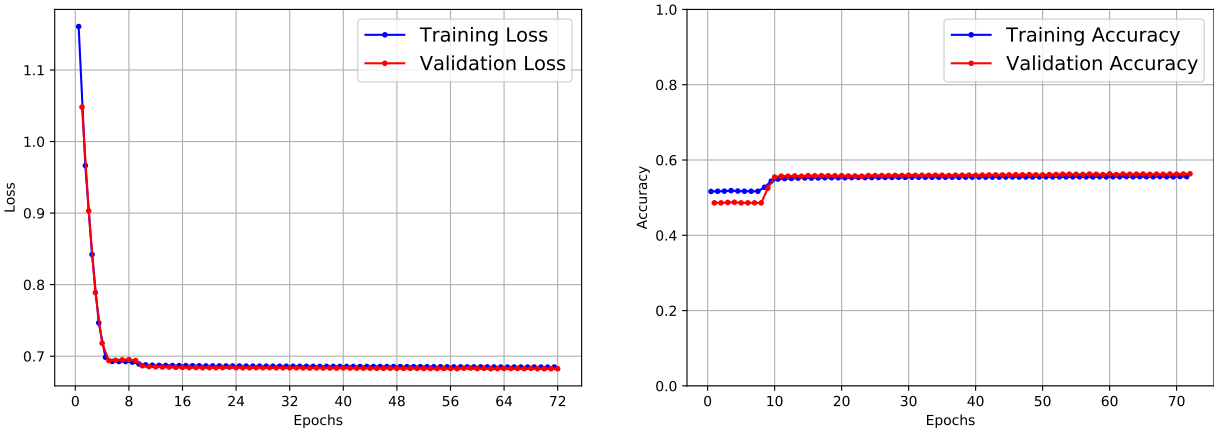



Figure 5.12: LSTM with Attention performance during training.

When compared to the best performing models without Attention, this LSTM can better identify down trends (58.8% vs 52.2% of an RNN model). However, although having a much better general accuracy, this model does not have such performance when dealing with up trends.

A final remark is that the best performing LSTM with Attention model outperforms any previous discussed model regarding certainty on identifying both trends by at least 3.3%.

## 5.7 Bidirectional LSTM with Attention

As explained in the previous Section 5.7, the model detailed in this section also employs an Additive Attention mechanism. However, making use of a bidirectional LSTM model instead. The purpose of this particular experiment is to take advantage of three powerful features available: LSTM recurrent neurons, bidirectional layers and Attention mechanisms.

Our intention is to use LSTM neurons to longer propagate information throughout the network, while bidirectional layers analyze both directions most important events in the Time Series (TS). From past

to present and from present to past. And, after this process is finished and information is represented, Attention mechanisms will highlight the more relevant parts in order to provide a more detailed representation to achieve a more accurate prediction.

As in previous sections, we followed the same logic to test different architectural variations. We have varied: the number of layers between 1, 2 and 4; the number of neurons between the number of input features and 100; and employed dropout layers with 0.2 and 0.5 rates. This experiment endured for 6 days, the same time as the previous section experiment and we have tested 20 hyper-parameter and architectural variations. In this model training, as in the previous model employing Attention mechanisms, a memory constraint exists. Therefore, when training this model different architectures we solely used training batches of 0.1%.

As can be observed in Table 5.7, the addition of bidirectional layers to the network improved accuracy, having the best performing bidirectional LSTM with Attention outperformed models without Attention by at least 5% and the LSTM with Attention model by 0.4%.

Table 5.7: Accuracy comparison of bidirectional LSTM models with Attention.

| **Bidirectional LSTM with Attention** | **Market only** | **Market + News propagated** | **Market + News balanced** | **Market + News neutralized** |
|---|---|---|---|---|
| **1 layer** | 0.503 | $\underline{0.603}$ | 0.501 | 0.501 |
| **2 layers = neurons 0.2 dropout** | 0.594 | 0.595 | 0.599 | **0.604** |
| **4 layers ↓ neurons** | 0.506 | 0.602 | $\underline{0.602}$ | 0.596 |
| **4 layers ↓ neurons 0.2 dropout** | 0.503 | 0.574 | 0.590 | 0.556 |
| **4 layers = neurons** | 0.599 | 0.596 | 0.599 | 0.602 |
| **4 layers = neurons 0.2 dropout** | $\underline{0.600}$ | 0.542 | 0.590 | 0.574 |

↓ represents the number of neurons decreasing across layers.
= represents the number of neurons being kept constant across layers.

From observing this table is also perceptible that simpler models performed better. We can observe that no model using layers with 200 neurons performed reasonable, otherwise it would be present in this table. In our opinion, this is due to these models tendency to overfit, in which not even dropout help the case. The only possible solution is to models with fewer neurons. Hence the best performing models being the more simpler ones presented in Table 5.7. Therefore, the question posed in Section 5.4 can be partly answered.

Fig. 5.13 details the best performing bidirectional LSTM with Attention architecture. This model architecture contains 2 layers, both containing the same number of neurons as the number of input features. The batch during training as aforementioned explained is 0.1% and this model makes use of dropout layers with a 0.2 rate to help avoiding overfit. The bidirectional LSTM model achieved an accuracy of 60.4%.

Fig. 5.14 details the loss function performance and corresponding accuracy throughout training of the bidirectional LSTM with Attention model detailed in Fig 5.13. This model achieved an MCC of 0.208, an AUC of 0.604, a Recall of 0.587, a Precision of 0.610, a $F_1$Score of 0.598, a Specificity of 0.621 and a NPV of 0.598.

When compared to all models, the bidirectional LSTM with Attention model has the best MCC, AUC, Precision and Specificity overall. Which translates in the aforementioned model having the most accu-

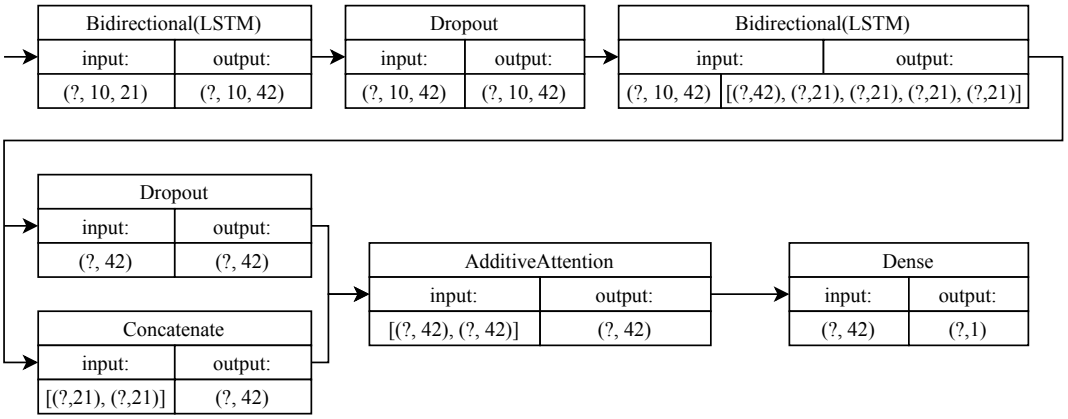



Figure 5.13: Best bidirectional LSTM with Attention architecture.

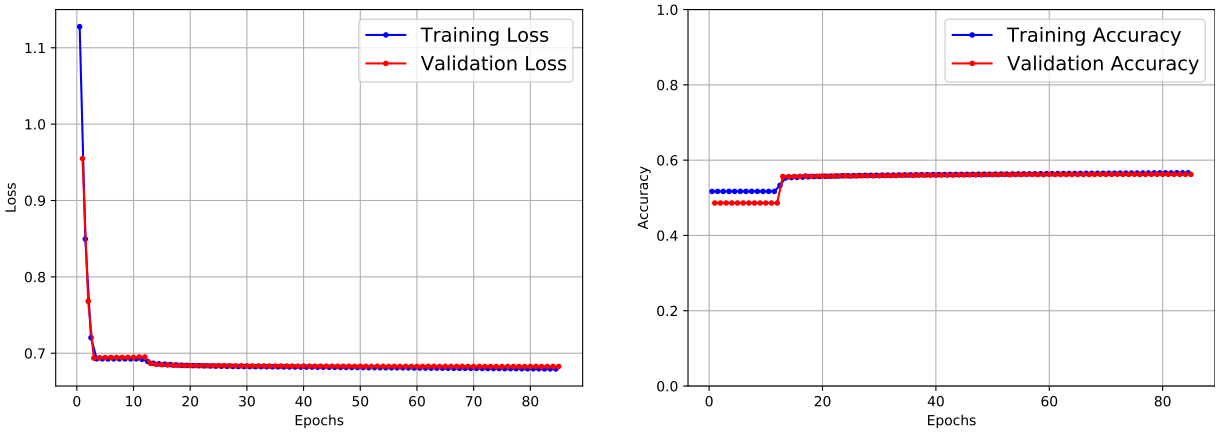



Figure 5.14: Bidirectional LSTM with Attention performance during training.

racy when identifying down trends and being the more certain when predicting up trends. However, a drawback is that almost all models have a better accuracy when solely identifying up trends.

## 5.8 Transformer

In this section we will present the experimental results of a model relying solely on attention. As introduced in Section 3.3, the Transformer model is a novelty because it does not involve any recurrence mechanisms, hence allowing parallel computation. A difference to previous presented models where the computation in order to learn patterns and dynamics from TS was performed sequentially.

In addition, due to solely relying on attention, this model does not require prior calculation of all hidden states, therefore enabling to obtain results significantly faster and, consequently reducing training time. This model was purposely created for Natural Language Processing (NLP) tasks, but we decided to adapt it and employ it in our Time Series Prediction (TSP) task. The experiment is new, because to our best knowledge this model was never applied to predict stock assets movement.

The structural modifications done in order to adapt the model to our task are detailed in Section 3.3 and Section 4.5. Throughout the experiments realized with the Transformer-like model we have varied the number of stack $N$ components between 2, 4, 6 and 8, and the number of attention heads present

in each component. The number of attention heads varied differently for the 3 datasets with sentiment features and for the dataset without sentiment features. This was due to the datasets having a different number of input features jointly with the constraint mentioned in Section 4.5 and presented in Eq. (4.1). *Market only* dataset has 18 input features versus 21 input features for the other 3 datasets using sentiment features. Consequently, $d_{model}$ has to be modified in accordance.

For the dataset without sentiment features, we have varied the number of attention heads between 5, 6, 9 and 10 having in total 16 different architectural versions. When testing datasets using sentiment features, we varied the number of attention heads between 5, 6, 7, 10 and 14 having a total of 20 different architectural versions. This experiment was realized throughout 4 days.

In Table 5.8 are presented the Transformer best hyper-parameter variations. As it can be observed, the best model uses a stack of 6 components, as in the original paper. Each one of these components containing a Multi-Head Attention mechanism with 5 attention heads, a FNN layer with 8000 neurons and two Add & Norm layers. All layers were already previously detailed in Section 3.3.

Table 5.8: Accuracy comparison of Transformer models.

| Transformer | Market only | Market + News propagated | Market + News balanced | Market + News neutralized |
|---|---|---|---|---|
| **5 Heads and 2 components** | 0.547 | 0.553 | 0.545 | 0.542 |
| **5 Heads and 6 components** | 0.543 | 0.547 | **0.557** | $\underline{0.550}$ |
| **6 Heads and 2 components** | 0.543 | 0.546 | 0.549 | $\underline{0.550}$ |
| **9 Heads and 4 components** | $\underline{0.556}$ | NA | NA | NA |
| **14 Heads and 2 components** | 0.550 | 0.545 | 0.544 | 0.543 |
| **14 Heads and 4 components** | NA | $\underline{0.552}$ | 0.544 | 0.545 |

In alignment with ref. [222], the best Transformer-like architecture for our task has the same number of components as the original model architecture for NLP tasks. Nonetheless, it should be noted that architectures using 4 components also performed well.

Contrasting with the number of components in the architecture is the number of attention heads. To our task, a decrease from the original 8 to 5 heads of attention was verified. Almost half the ones presented in ref. [222]. We attribute this to our datasets having significantly less features when comparing with translation tasks, hence extra heads of attention would dramatically increase complexity and cause the model to overfit.

Fig. 5.15 details the loss function performance and corresponding accuracy throughout training of the Transformer-like model. This model achieved an MCC of 0.108, an AUC of 0.556, a Recall of 0.557, Precision of 0.556, a $F_1$Score of 0.557, a Specificity of 0.550 and NPV of 0.551.

The contribution of a model solely relying on Attention mechanisms is mainly two-fold. First, we wanted to compare performance against models relying on recurrence mechanisms. Second, we wanted to understand if the increasing of accuracy when adding attention to models with recurrence was due to a particular combination of both mechanisms or if Attention alone was a more robust solution.

The conclusions from the results obtained are clear. Employing each mechanism individually results in achieving similar accuracy (55.3% and 55.7%). Having recurrent models more accuracy when identifying up trends and the Transformer-like model more accuracy when identifying down trends. The LSTM

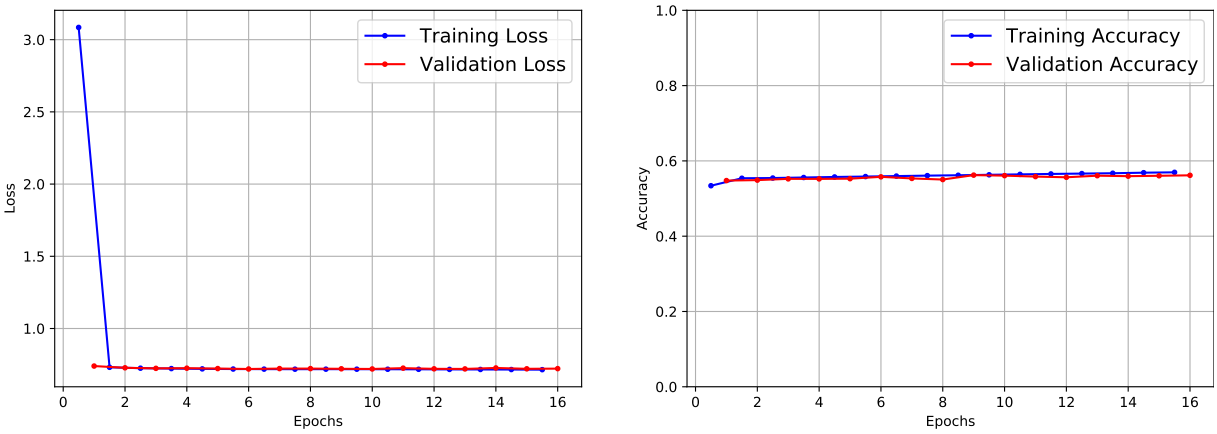



Figure 5.15: Transformer performance during training.

and bidirectional LSTM models have a Recall of 67.3% and 61.2% respectively, versus 55.7% from the Transformer model. The Transformer-like model has 55.0% Specificity versus 43.3% and 49.5%, correspondingly for LSTM and bidirectional LSTM models.

The main conclusion reached is that combining both mechanisms, attention and recurrence, definitely increases accuracy. Although Recall is lower when compared with recurrent models, Specificity is 13% higher. Meaning that models using both attention and recurrence can significantly better predict down trends.

## 5.9 Comparison and Discussion

The first experiment, with a simple FNN (Section 5.1), already provides an interesting conclusion: when using this model, the sentiment features do not have any influence in the output predictions. In our view, this is due to the fact that the FNN does not incorporate any information on past behavior. Both with and without the sentiment features, this model scored a test accuracy of 53.3%.

Using a Convolutional Neural Network (CNN) (Section 5.2) with a 10 time-step window into the past improved the prediction accuracy. However, the test accuracy was higher without the sentiment features (55.1%) than with those features included (54.8%). We attribute this result to a prior assumption of structure importance in a spatial domain of CNN convolutions, where closer features are prioritized. Moreover, as the most accurate CNN model made use of a kernel with size 2, it lacked ability to better capture the influence of the 3 existent sentiment categories. Additionally, dropout did not help to prevent overfit in this case.

A simple RNN (Section 5.3) provided the same accuracy (55.2%) with and without sentiment features, but a LSTM (Section 5.4) improved from 55.2% to 55.3% when the sentiment features were included. We conclude that recurrence mechanisms are essential in order to leverage information from the past. However, a bidirectional LSTM (Section 5.5) did not improve the results, which is understandable, since the most recent history is probably more relevant for prediction than the distant past, so only the forward direction is useful.

It is our opinion that the difference between RNN and LSTM models is not so noticeable due to the

fixed-length training window. If instead we have used an incremental training window, it is our belief that the LSTM results would outperform more clearly the RNN.

The accuracy improves even further when attention mechanisms are employed. An LSTM model with additive attention (Section 5.6) achieves 59.8% accuracy without sentiment features, and reaches 60.0% when those features are included. Moreover, a bidirectional LSTM with attention (Section. 5.7) improves that result even further to 60.4%. Here, bidirectionality is definitely beneficial because it helps the attention mechanism learn which time-steps are the most relevant for prediction. The results suggest that attention mechanism can select the relevant features of an input and improve prediction. It also implies that input features are not equally important, a consistent conclusion with our work review.

We have also experimented with a Transformer-like architecture, which relies on attention but not recurrence. Here the results were inferior, with 55.6% accuracy without sentiment features, and an increase to 55.7% when those features were included. This model achieved similar result as models solely using recurrence when using sentiment features, and sightly better results when predicting without sentiment features.

In our opinion, the Transformer model did not perform as expected. This type of attention performs well when repeated inputs exist due to the usage of dot-product as alignment score function. Contrarily to NLP tasks, Stock Prediction (SP) has similar trends but values rarely repeat. Therefore, we believe this was the reason for under-performance.

Table 5.9 provides an overview of the overall results, where it becomes apparent that the inclusion of sentiment features can account for an improvement of at most 0.6% in test accuracy, while the use of recurrence and attention mechanisms can provide an improvement of 5.6% over a baseline CNN that uses the same input data.

Table 5.9: Accuracy comparison of different models.

| Models | Market only | Market + News propagated | Market + News balanced | Market + News neutralized |
|---|---|---|---|---|
| **FNN** | 0.533 | 0.532 | 0.533 | 0.533 |
| **CNN** | **0.551** | 0.543 | 0.543 | 0.548 |
| **RNN** | 0.552 | 0.547 | **0.552** | 0.550 |
| **LSTM** | 0.552 | 0.551 | **0.553** | 0.551 |
| **Bidirectional LSTM** | 0.543 | 0.548 | 0.547 | **0.549** |
| **LSTM with Attention** | 0.598 | 0.599 | 0.600 | **0.600** |
| **Bidirectional LSTM with Attention** | 0.600 | 0.603 | 0.602 | **0.604** |
| **Transformer** | 0.556 | 0.552 | **0.557** | 0.550 |

When comparing models, the Transformer-like model is far more complex than the remaining. This model has almost 6,000,000 trainable parameters whereas most of the presented models do not even reach 100,000 trainable parameters. Only the LSTM and bidirectional LSTM models, due to having 4 layers and 100 neurons per layer reach bigger values. Just over 300,000 and 800,000 trainable parameters, respectively. We can conclude that a direct correlation does not exist between a more complex model, i.e. a model with more trainable parameters, and its better performance.

Table 5.10 presents an overview of the additional measures introduced earlier in Section 4.6 to provide a more detailed explanation of the results of each model. It seems logical that the best performing

model in terms of accuracy also has the best score in almost every category. The only unexpected result is the FNN model, that although having the best accuracy when identifying uptrends, scored the lowest accuracy from all. We justify this due to the stock market being in a constant growth in recent years combined with this model inability to capture market reversions, hence predicting more uptrends than downtrends.

Table 5.10: Comparison of additional measures.

| Models | MCC | AUC | Recall | Precision | $F_1$-score | Specificity | NPV |
|---|---|---|---|---|---|---|---|
| **FNN** | 0.070 | 0.532 | **0.728** | 0.524 | **0.609** | 0.336 | 0.552 |
| **CNN** | 0.097 | 0.547 | 0.649 | 0.542 | 0.591 | 0.446 | 0.557 |
| **RNN** | 0.103 | 0.552 | 0.581 | 0.551 | 0.566 | 0.522 | 0.552 |
| **LSTM** | 0.109 | 0.553 | 0.673 | 0.545 | 0.603 | 0.433 | 0.567 |
| **Bidirectional LSTM** | 0.098 | 0.548 | 0.627 | 0.545 | 0.583 | 0.470 | 0.555 |
| **LSTM with Attention** | 0.199 | 0.600 | 0.612 | 0.600 | 0.606 | 0.588 | **0.600** |
| **Bidirectional LSTM with Attention** | **0.208** | **0.604** | 0.587 | **0.610** | 0.598 | **0.621** | 0.598 |
| **Transformer** | 0.108 | 0.556 | 0.557 | 0.556 | 0.557 | 0.550 | 0.551 |

Fig. 5.16 aggregates both training (top row) and validation (bottom row) losses of all the best performing NN models. On the left, there is a zoom out comparison, of all models training. From the left part of the figure, we can observe that RNN required by far the most quantity of epochs to be trained ($\sim$ 330 epochs), whereas the Transformer-like model required the least ($\sim$ 15 epochs), having the remaining models all required between $\sim$ 40 and $\sim$ 90 epochs.

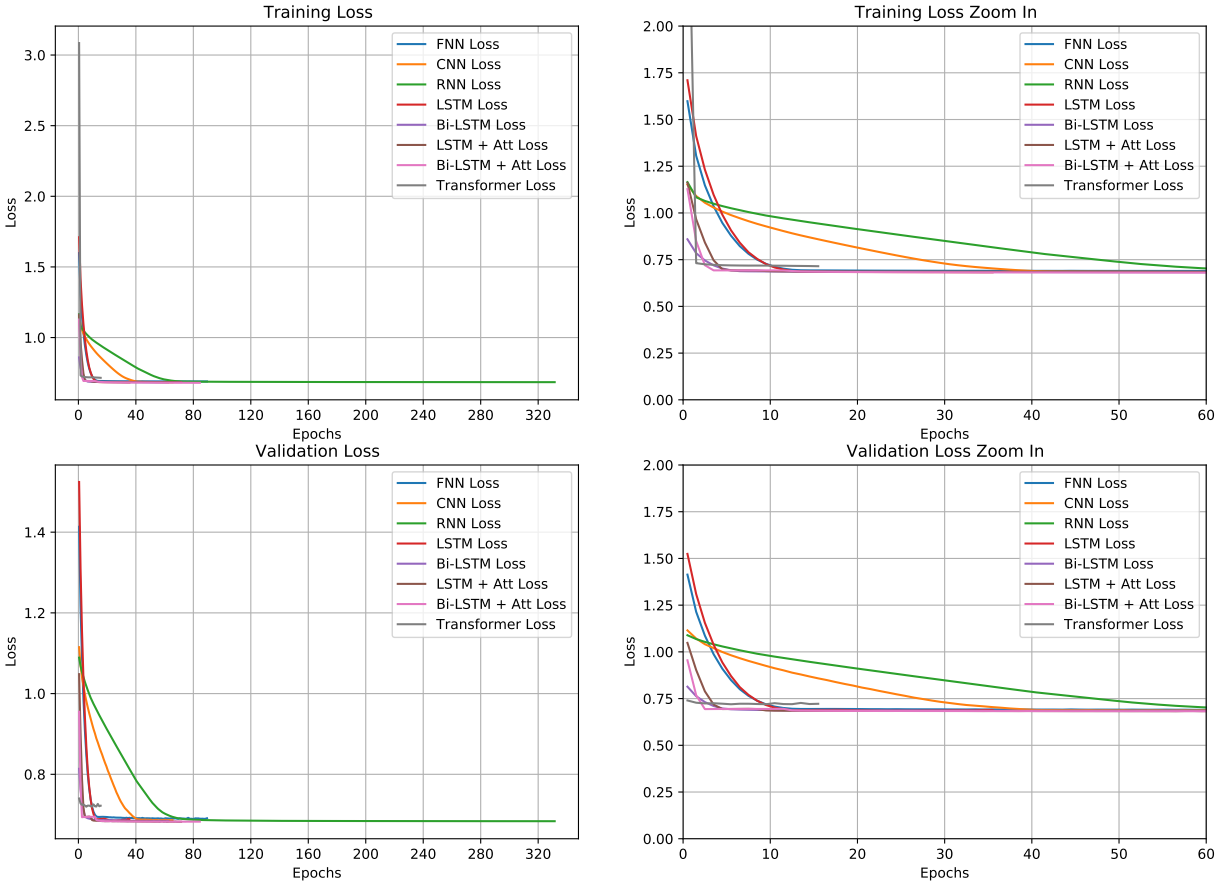



Figure 5.16: Training and Validation Loss during training.

As can be observed in Fig. 5.16, less steeper slopes belong to less complex models: a 2 layers RNN and a 2 kernel size CNN. On the contrary, regarding noticeably larger slopes, a great majority of the remaining models have a 4 layers architecture with 100 neurons in each layer. Therefore, we can conclude, for this particular dataset, that the more complex a model is, i.e. the more trainable parameters a model has, the faster it learns.

To reinforce the previous conclusion another observation can be added. It is possible to verify that the Transformer-like model started with the highest loss. On that note, due to this model being the one with less training epochs and all models having similar loss values at the end of training, it is possible to state that the more complex model obtained the fastest loss decrease. Therefore, it can also be concluded that for this particular dataset the most complex model can be considered the fastest learner. And although not the highest accuracy scorer, the Transformer-like model achieved similar results to recurrent models with significantly less training. Consequently, when solely comparing recurrent mechanisms against attention, we can conclude that the latter learns faster.

In summary, according to our experiments and results performed on this particular dataset, we conclude that the incorporation of Sentiment Analysis (SA) in SP can account for at most a 0.6% improvement in predicting the upward or downward movement of stocks, while the choice of model and the use of more sophisticated learning mechanisms such as recurrence and attention can provide a significantly larger improvement with respect to simpler baseline models.

# Chapter 6

# Conclusion

From the literature review presented in Chapter 2, we expected that Sentiment Analysis (SA) would improve stock prediction. However, our results suggest that the improvement is rather modest compared to our initial expectation.

On the other hand, our findings point to a possible avenue for the continued improvement of prediction models and their accuracy. By making use of learning mechanisms such as recurrence and attention, and others that may appear along the way, it is possible to keep improving the results. It is also apparent that such mechanisms will work better in combination rather than in isolation; specifically, recurrence with attention works better than either of those mechanisms alone.

In addition, we found that propagating sentiment by keeping past sentiment when there are no news is not the best approach. In this respect, the market seems to forget past sentiment in a matter of days, to the point that it is no longer possible to clearly determine the current sentiment as being positive, negative, or neutral.

As an advantage, due to employing Deep Learning models, our approach can adapt to different markets without any architectural modification. The only requisite is to train the model with specific data from the corresponding market. In addition, it is our belief that this system can be employed to other tasks rather than Stock Prediction (SP) directional movement such as price or volatility prediction. Out of our scope, this system can possibly be employed to medical, life or car insurance calculation based on the users statistics, history and social media posts.

Moreover, our approach can be considered a generic framework for modeling various non-linear dynamical stock assets interactions. As manifested in our results section, this approach can model multivariate Time Series (TS) for a 10 day time period with reasonable accuracy. However, the framework proposed is extensible and can be easily adapted to model both univariate and/or different time lengths into the future with minimum modifications.

To conclude, we want to directly answer the question we present in Section 1.3. The inclusion of Sentiment Analysis over a pure Technical Analysis approach when doing Stock Prediction can account for an improvement of at most 0.6% in our test accuracy. Nonetheless, the use of recurrence and attention mechanisms can provide an improvement of 5.6% over a baseline using the same input data.

## 6.1 Main Contributions

In this work we provided a number of contributions, which can briefly summarized as follows:

- a review over the main techniques used for Stock Prediction;
- an overview, in the form of comprehensive tables, of the main sources of market data, the main sources of news data, and the relationship between market and news sources;
- a review of different types of Neural Network (NN) models and their characteristics, with a view towards their application to Stock Prediction;
- an introduction to Attention mechanisms, where we discuss how to modify a mechanism that was purposely built for Natural Language Processing (NLP) tasks;
- a detailed approach on how to retrieve, preprocess and analyze information in order to build a SP model;
- a detailed description on how to build a SP model, and the construction of several DL models for SP;
- the construction of an innovative SP model solely based on Attention mechanisms and inspired by the Transformer model;
- a methodology in order to measure the influence of Sentiment Analysis when doing Stock Prediction;
- a comparison of the results obtained with different models, and an explanation for the performance achieved with each model;
- a demonstration of the importance of both recurrence and attention mechanisms in SP;
- the quantification of how much SA and different types of NN mechanisms influence SP.

An interesting novelty of this work was the implementation of a Parallel Attention mechanism for SP. Our results point to the fact that including only recurrence mechanisms or only attention mechanisms produce a similar effect. However, combining both mechanisms improves performance significantly.

Some of the main contributions of this work are to be presented and published as a paper at the *Fifth Workshop on MIning DAta for financial applicationS* (MIDAS 2020) at the *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases* (ECML-PKDD 2020) which is due to take place in Ghent, Belgium, in September 2020.

## 6.2 Future Work

Despite the interesting results, there are many ideas to explore as future work. In this work, our prediction target was binary, in the form of a directional movement. As limitation, our approach is only able

to predict assets price increasing and decreasing movements. Considering this fact, we do not know if they are affordable and how much they are going to move. Given the time restrictions, it was impossible for us to evaluate how well our project would do in the real world.

In future work, we plan to apply similar models for regression tasks such as predicting price and volatility. In addition, the kind of evaluation that we provide here is by no means the end of the story; to derive actual benefits in the real-world, other components, such as a trading strategy to enter and exit the market, are necessary. Additionally, it can be added the determination of an optimal number of stocks to buy or sell in order to maximize overall profits; to consider the risk of each asset bought having also in mind its transaction costs and volume constraints.

# Bibliography


[1] E. F. Fama. The Behavior of Stock-Market Prices. *The Journal of Business*, 38(1):34–105, Jan. 1965.

[2] R. N. Elliott. *The Wave Principle*. 1938.

[3] E. F. Fama. Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance*, 25(2):383–417, May 1970.

[4] R. P. Schumaker and H. Chen. Textual Analysis of Stock Market Prediction Using Financial News Articles. In *Americas Conference On Information Systems*, volume 3, pages 1422–1430, Dec. 2006.

[5] I. Maks and P. Vossen. A verb lexicon model for deep sentiment analysis and opinion mining applications. *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, pages 10–18, June 2011.

[6] V. Lavrenko, M. Schmill, D. Lawrie, P. Ogilvie, D. Jensen, and J. Allan. Mining of Concurrent Text and Time Series. *In Proceedings of the 6th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining Workshop on Text Mining*, pages 37–44, 2000.

[7] X. Ding, Y. Zhang, T. Liu, and J. Duan. Using Structured Events to Predict Stock Price Movement: An Empirical Investigation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1415–1425, Oct. 2014.

[8] B. Weng, L. Lu, X. Wang, F. M. Megahed, and W. Martinez. Predicting short-term stock prices using ensemble methods and online data sources. *Expert Systems with Applications*, 112:258–273, Dec. 2018.

[9] R. Luss and A. D'Aspremont. Predicting abnormal returns from news usingtext classification. *Quantitative Finance*, 15(6):999–1012, June 2015.

[10] T. Hollis, A. Viscardi, and S. E. Yi. A Comparison of LSTMs and Attention Mechanisms for Forecasting Financial Time Series. *arXiv:1812.07699*, Dec. 2018.

[11] A. K. Nassirtoussi, S. Aghabozorgi, T. Y. Wah, and D. Ngo. Text mining for market prediction: A systematic review. *Expert Systems with Applications*, 41(16):7653–7670, nov 2014.

[12] A. K. Nassirtoussi, S. Aghabozorgi, T. Y. Wah, and D. C. L. Ngo. Text mining of news-headlines for FOREX market prediction: A Multi-layer Dimension Reduction Algorithm with semantics and sentiment. *Expert Systems with Applications*, 42(1):306–324, Jan. 2015.

[13] H. Markowitz. Portfolio Selection. *The Journal of Finance*, 7(1):77–91, Mar. 1952.

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

[15] S. Emerson, R. Kennedy, L. O'Shea, and J. O'Brien. Trends and Applications of Machine Learning in Quantitative Finance. In *8th International Conference on Economics and Finance Research (ICEFR 2019)*, June 2019.

[16] A. J. Hussain, A. Knowles, P. J. G. Lisboa, and W. El-Deredy. Financial time series prediction using polynomial pipelined neural networks. *Expert Systems with Applications*, 35(3):1186–1199, Oct. 2008.

[17] J. C. B. Gamboa. Deep Learning for Time-Series Analysis. *arXiv:1701.01887*, Jan. 2017.

[18] B. Vanstone and G. Finnie. An empirical methodology for developing stock market trading systems using artificial neural networks. *Expert Systems with Applications*, 36(3):6668–6680, Apr. 2009.

[19] W. Huang, Y. Nakamori, and S.-Y. Wang. Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, 32(10):2513–2522, Oct. 2005.

[20] G. E. P. Box and G. Jenkins. *Time Series Analysis, Forecasting and Control*. Wiley, 5 edition, 1990.

[21] W. Long, Z. Lu, and L. Cui. Deep learning-based feature engineering for stock price movement prediction. *Knowledge-Based Systems*, 164:163–173, Jan. 2019.

[22] M.-E. Wu, C.-H. Wang, and W.-H. Chung. Using trading mechanisms to investigate large futures data and their implications to market trends. *Soft Computing*, 21(11):2821–2834, June 2017.

[23] R. Arévalo, J. García, F. Guijarro, and A. Peris. A dynamic trading rule based on filtered flag pattern recognition for stock market price forecasting. *Expert Systems with Applications*, 81:177–192, Sept. 2017.

[24] J. M. Poterba and L. H. Summers. Mean reversion in stock prices: Evidence and Implications. *Journal of Financial Economics*, 22(1):27–59, Oct. 1988.

[25] S.-H. Hsu, J. J. P.-A. Hsieh, T.-C. Chih, and K.-C. Hsu. A two-stage architecture for stock price forecasting by integrating self-organizing map and support vector regression. *Expert Systems with Applications*, 36(4):7947–7951, May 2009.

[26] W. L. Tung and C. Quek. Financial volatility trading using a self-organising neural-fuzzy semantic network and option straddle-based approach. *Expert Systems with Applications*, 38(5):4668–4688, May 2011.

[27] R. F. Engle. Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica: Journal of the Econometric Society*, 50(4):987–1007, July 1982.

[28] T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, 31(3):307–327, Apr. 1986.

[29] J. Fang, B. Jacobsen, and Y. Qin. Predictability of the simple technical trading rules: An out-of-sample test. *Review of Financial Economics*, 23(1):30–45, Jan. 2014.

[30] Y. Kara, M. A. Boyacioglu, and Ö. K. Baykan. Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. *Expert Systems with Applications*, 38(5):5311–5319, May 2011.

[31] G. S. Atsalakis and K. P. Valavanis. Surveying stock market forecasting techniques–Part II: Soft computing methods. *Expert Systems with Applications*, 36(3):5932–5941, Apr. 2009.

[32] J. Döpke, U. Fritsche, and C. Pierdzioch. Predicting recessions with boosted regression trees. *International Journal of Forecasting*, 33(4):745–759, Oct. 2017.

[33] E. Kayacan, B. Ulutas, and O. Kaynak. Grey system theory-based models in time series prediction. *Expert Systems with Applications*, 37(2):1784–1789, Mar. 2010.

[34] F. D. Paiva, R. T. N. Cardoso, G. P. Hanaoka, and W. M. Duarte. Decision-making for financial trading: A fusion approach of machine learning and portfolio selection. *Expert Systems with Applications*, 115:635–655, Jan. 2019.

[35] M. S. Lauretto, B. B. C. Silva, and P. M. Andrade. Evaluation of a Supervised Learning Approach for Stock Market Operations. *arXiv:1301.4944*, Jan. 2013.

[36] L.-J. Cao and F. E. H. Tay. Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks*, 14(6):1506–1518, Nov. 2003.

[37] K. Kim. Financial time series forecasting using support vector machines. *Neurocomputing*, 55 (1-2):307–319, Sept. 2003.

[38] C.-J. Huang, D.-X. Yang, and Y.-T. Chuang. Application of wrapper approach and composite classifier to the stock trend prediction. *Expert Systems with Applications*, 34(4):2870–2878, May 2008.

[39] J. Sen and T. D. Chaudhuri. Decomposition of Time Series Data of Stock Markets and its Implications for Prediction – An Application for the Indian Auto Sector. In *Proceedings of the 2nd National Conference on Advances in Business Research and Management Practices (ABRMP '2016), Kolkata, India, January 8-9, 2016.*, 01 2016.

[40] R. C. Cavalcante, R. C. Brasileiro, V. L. F. Souza, J. P. Nobrega, and A. L. I. Oliveira. Computational Intelligence and Financial Markets: A Survey and Future Directions. *Expert Systems with Applications*, 55:194–211, Aug. 2016.

[41] G. Zhang, B. E. Patuwo, and M. Y. Hu. Forecasting with artificial neural networks: The state of the art. *International journal of forecasting*, 14(1):35–62, Mar. 1998.

[42] E. Chong, C. Han, and F. C. Park. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, 83:187–205, Oct. 2017.

[43] B. Moews, J. M. Herrmann, and G. Ibikunle. Lagged correlation-based deep learning for directional trend change prediction in financial time series. *Expert Systems with Applications*, 120:197–206, Apr. 2019.

[44] N. Kohzadi, M. S. Boyd, B. Kermanshahi, and I. Kaastra. A comparison of artificial neural network and time series models for forecasting commodity prices. *Neurocomputing*, 10(2):169–181, Mar. 1996.

[45] G. P. Zhang. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, 50:159–175, Jan. 2003.

[46] S. P. Chatzis, V. Siakoulis, A. Petropoulos, E. Stavroulakis, and N. Vlachogiannakis. Forecasting stock market crisis events using deep and statistical machine learning techniques. *Expert Systems with Applications*, 112:353–371, Dec. 2018.

[47] C. Krauss, X. A. Do, and N. Huck. Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. *European Journal of Operational Research*, 259(2):689–702, June 2017.

[48] J. G. D. Gooijer and R. J. Hyndman. 25 years of time series forecasting. *International Journal of Forecasting*, 22(3):443–473, 2006.

[49] A. Arévalo, J. Nino, G. Hernandez, and J. Sandoval. High-Frequency Trading Strategy Based on Deep Neural Networks. In Springer, editor, *Lecture Notes in Computer Science*, volume 9773, pages 424–436, 08 2016.

[50] M. Dixon, D. Klabjan, and J. H. Bang. Classification-based financial markets prediction using deep neural networks. *Algorithmic Finance*, 6(3-4):67–77, Jan. 2017.

[51] F. Zhou, H.-m. Zhou, Z. Yang, and L. Yang. EMD2FNN: A strategy combining empirical mode decomposition and factorization machine based neural network for stock market trend prediction. *Expert Systems with Applications*, 115:136–151, Jan. 2019.

[52] Y. Kim. Convolutional Neural Networks for Sentence Classification. *arXiv:1408.5882*, Oct. 2014.

[53] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, July 2019.

[54] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. In *Arxiv*, 2016. URL `https://arxiv.org/abs/1609.03499`.

[55] C.-H. Cho, G.-Y. Lee, Y.-L. Tsai, and K.-C. Lan. Toward Stock Price Prediction Using Deep Learning. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, UCC '19 Companion, pages 133–135, New York, NY, USA, Dec. 2019. Association for Computing Machinery. ISBN 9781450370448. doi: 10.1145/3368235.3369367.

[56] M. Hiransha, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman. NSE Stock Market Prediction Using Deep-Learning Models. *Procedia Computer Science*, 132:1351–1362, 2018.

[57] E. Hoseinzade and S. Haratizadeh. CNNpred: CNN-based stock market prediction using a diverse set of variables. *Expert Systems with Applications*, 129:273–285, Sept. 2019.

[58] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman. Stock price prediction using LSTM, RNN and CNN-sliding window model. In *International Conference on Advances in Computing, Communications and Informatics*, Sept. 2017.

[59] L. Di Persio and O. Honchar. Artificial neural networks architectures for stock price prediction: Comparisons and applications. *International Journal of Circuits, Systems and Signal Processing*, 10:403–413, 2016.

[60] I. Sutskever, J. Martens, and G. E. Hinton. Generating Text with Recurrent Neural Networks. In *Proceedings of the 28th International Conference on Machine Learning*, pages 1017–1024, 2011.

[61] M. R. Vargas, B. S. L. P. De Lima, and A. G. Evsukoff. Deep learning for stock market prediction from financial news articles. In *International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications*, pages 60–65, June 2017.

[62] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai. Deep Direct Reinforcement Learning for Financial Signal Representation and Trading. *IEEE Transactions on Neural Networks and Learning Systems*, 28(3):653–664, Mar. 2017.

[63] Y. Bengio, P. Simard, and P. Frasconi. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE transactions on neural networks*, 5(2):157–166, Mar. 1994.

[64] S. Hochreiter and J. Schmidhuber. Long Short Term Memory. *Neural Computation*, 9(8):1735–1780, Nov. 1997.

[65] D. Nelson, A. Pereira, and R. de Oliveira. Stock market's price movement prediction with LSTM neural networks. *International Joint Conference on Neural Networks*, May 2017.

[66] H. Y. Kim and C. H. Won. Forecasting the volatility of stock price index: A hybrid model integrating LSTM with multiple GARCH-type models. *Expert Systems with Applications*, 103:25–37, Aug. 2018.

[67] Y. Baek and H. Y. Kim. ModAugNet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module. *Expert Systems with Applications*, 113:457–480, Dec. 2018.

[68] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078*, June 2014.

[69] S. Kim and M. Kang. Financial series prediction using Attention LSTM. *arXiv:1902.10877*, Feb. 2019.

[70] M. P. Austin, G. Bates, M. A. H. Dempster, V. Leemans, and S. N. Williams. Adaptive systems for foreign exchange trading. *Quantitative Finance*, 4(4):37–45, Aug. 2004.

[71] R. T. F. Nazário, J. L. e Silva, V. A. Sobreiro, and H. Kimura. A literature review of technical analysis on stock markets. *The Quarterly Review of Economics and Finance*, 66:115–126, Nov. 2017.

[72] F. Bertoluzzo and M. Corazza. Testing different Reinforcement Learning configurations for financial trading: Introduction and applications. *Procedia Economics and Finance*, 3:68–77, Jan. 2012.

[73] P. C. Pendharkar and P. Cusatis. Trading financial indices with reinforcement learning agents. *Expert Systems with Applications*, 103:1–13, Aug. 2018.

[74] S. Almahdi and S. Y. Yang. An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications*, 87:267–279, Nov. 2017.

[75] G. Jeong and H. Y. Kim. Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning. *Expert Systems with Applications*, 117:125–138, Mar. 2019.

[76] J. Si, A. Mukherjee, B. Liu, Q. Li, H. Li, and X. Deng. Exploiting Topic based Twitter Sentiment for Stock Prediction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 24–29, Aug. 2013.

[77] J. Si, A. Mukherjee, B. Liu, S. J. Pan, Q. Li, and H. Li. Exploiting Social Relations and Sentiment for Stock Prediction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1139–1145, Oct. 2014.

[78] B. Dickinson and W. Hu. Sentiment Analysis of Investor Opinions on Twitter. *Social Networking*, 4(3):62–71, July 2015.

[79] A. Mittal and A. Goel. Stock Prediction Using Twitter Sentiment Analysis. *Standford University, CS229*, 2012.

[80] T. Fischer and C. Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, Oct. 2018.

[81] T. K. Lee, J. H. Cho, D. S. Kwon, and S. Y. Sohn. Global stock market investment strategies based on financial network indicators using machine learning techniques. *Expert Systems with Applications*, 117:228–242, Mar. 2019.

[82] E. Guresen, G. Kayakutlu, and T. U. Daim. Using artificial neural network models in stock market index prediction. *Expert Systems with Applications*, 38(8):10389–10397, Aug. 2011.

[83] H. Wang, S. Lu, and J. Zhao. Aggregating multiple types of complex data in stock market prediction: A model-independent framework. *Knowledge-Based Systems*, 164:193–204, Jan. 2019.

[84] T. H. Nguyen and K. Shirai. Topic Modeling based Sentiment Analysis on Social Media for Stock Market Prediction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 1354–1364, July 2015.

[85] L. S. Malagrino, N. T. Roman, and A. M. Monteiro. Forecasting stock market index daily direction: A Bayesian Network approach. *Expert Systems with Applications*, 105:11–22, Sept. 2018.

[86] A. A. Nasseri, A. Tucker, and S. de Cesare. Quantifying StockTwits semantic terms' trading behavior in financial markets: An effective application of decision tree algorithms. *Expert Systems with Applications*, 42(23):9192–9210, Dec. 2015.

[87] S. Deng, T. Mitsubuchi, K. Shioda, T. Shimada, and A. Sakurai. Combining Technical Analysis with Sentiment Analysis for Stock Price Prediction. In *Ninth International Conference on Dependable, Autonomic and Secure Computing*, pages 800–807, Dec. 2011.

[88] X. Li, H. Xie, L. Chen, J. Wang, and X. Deng. News impact on stock price return via sentiment analysis. *Knowledge-Based Systems*, 69:14–23, Oct. 2014.

[89] J. Li, H. Bu, and J. Wu. Sentiment-aware stock market prediction: A deep learning method. In *International Conference on Service Systems and Service Management*, June 2017.

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

[91] J. Zhang, S. Cui, Y. Xu, Q. Li, and T. Li. A novel data-driven stock price trend prediction system. *Expert Systems with Applications*, 97:60–69, May 2017.

[92] X. Ding, Y. Zhang, T. Liu, and J. Duan. Deep Learning for Event-Driven Stock Prediction. *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 2327–2333, July 2015.

[93] J. Bollen, H. Mao, and X. Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, Mar. 2011.

[94] J. Prosky, X. Song, A. Tan, and M. Zhao. Sentiment Predictability for Stocks. *arXiv:1712.05785*, Dec. 2017.

[95] M.-Y. Day and C.-C. Lee. Deep learning for financial sentiment analysis on finance news providers. In *International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1127–1134, Aug. 2016.

[96] M. Mäntylä, D. Graziotin, and M. Kuutila. The evolution of sentiment analysis - A review of research topics, venues, and top cited papers. *Computer Science Review*, 27:16–32, Feb. 2018.

[97] A. Devitt and K. Ahmad. Sentiment Polarity Identification in Financial News: A Cohesion-based Approach. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 984–991, June 2007.

[98] S. Feuerriegel and H. Prendinger. News-based trading strategies. *Decision Support Systems*, 90: 65–74, Oct. 2016.

[99] L. M. Rojas-Barahona. Deep learning for sentiment analysis. *Language and Linguistics Compass*, 10(12):701–719, Dec. 2016.

[100] M. Melvin and X. Yin. Public Information Arrival, Exchange Rate Volatility, and Quote Frequency. *The Economic Journal*, 110(465):644–661, July 2000.

[101] V. Sehgal and C. Song. SOPS: Stock Prediction Using Web Sentiment. In *Proceedings of the 7th IEEE International Conference on Data Mining*, pages 21–26, Oct. 2007.

[102] J. R. Nofsinger. The impact of public information on investors. *Journal of Banking & Finance*, 25 (7):1339–1366, July 2001.

[103] L. Peng, G. Cui, M. Zhuang, and C. Li. What do seller manipulations of online product reviews mean to consumers? *HKIBS Working Paper Series 070-1314*, Jan. 2014.

[104] W. Medhat, A. Hassan, and H. Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, 5(4):1093–1113, Dec. 2014.

[105] E. D. Brown. Will Twitter Make You a Better Investor? A Look at Sentiment, User Reputation and Their Effect on the Stock Market. *Southern Association for Information Systems*, pages 36–42, Mar. 2012.

[106] E. Cambria, B. Schuller, Y. Xia, and C. Havasi. New Avenues in Opinion Mining and Sentiment Analysis. *IEEE Intelligent Systems*, 28(2):15–21, Mar. 2013.

[107] N. O'Hare, M. Davy, A. Bermingham, P. Ferguson, P. Sheridan, C. Gurrin, and A. F. Smeaton. Topic-Dependent Sentiment Analysis of Financial Blogs. In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 9–16, Nov. 2009.

[108] K. Ravi and V. Ravi. A survey on opinion mining and sentiment analysis: Tasks, approaches and applications. *Knowledge-Based Systems*, 89:14–46, Nov. 2015.

[109] R. Johnson and T. Zhang. Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. *arXiv:1412.1058*, Dec. 2014.

[110] T. Nasukawa and J. Yi. Sentiment analysis: capturing favorability using natural language processing. In *Proceedings of the 2nd international conference on Knowledge capture*, pages 70–77, Oct. 2003.

[111] D. Davidov, O. Tsur, and A. Rappoport. Semi-Supervised Recognition of Sarcastic Sentences in Twitter and Amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 107–116, July 2010.

[112] R. Deepak and E. Hovy. Learning surface text patterns for a Question Answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 41–47, July 2002.

[113] D. G. Maynard and M. A. Greenwood. Who cares about sarcastic tweets? Investigating the impact of sarcasm on sentiment analysis. In *Language Resources and Evaluation Conference*, May 2014.

[114] R. Narayanan, B. Liu, and A. Choudhary. Sentiment analysis of conditional sentences. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, volume 1, pages 180–189, Aug. 2009.

[115] E. Kontopoulos, C. Berberidis, T. Dergiades, and N. Bassiliades. Ontology-based sentiment analysis of twitter posts. *Expert Systems with Applications*, 40(10):4065–4074, Aug. 2013.

[116] X. Ding, B. Liu, and L. Zhang. Entity discovery and assignment for opinion mining applications. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1125–1134, July 2009.

[117] A. Moreno-Ortiz and J. Fernández-Cruz. Identifying Polarity in Financial Texts for Sentiment Analysis: A Corpus-based Approach. *Procedia-Social and Behavioral Sciences*, 198:330–338, July 2015.

[118] J. Ruppenhofer, M. Ellsworth, M. Schwarzer-Petruck, C. R. Johnson, and J. Scheffczyk. *FrameNet II: Extended theory and practice*. n.p., Sept. 2010.

[119] A. Esuli and F. Sebastiani. SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining. In *International Conference on Language Resources and Evaluation*, volume 6, pages 417–422, May 2006.

[120] E. C. Dragut, C. Yu, P. Sistla, and W. Meng. Construction of a sentimental word dictionary. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1761–1764, Oct. 2010.

[121] W. Peng and D. H. Park. Generate Adjective Sentiment Dictionary for Social Media Sentiment Analysis Using Constrained Nonnegative Matrix Factorization. In *Fifth International AAAI Conference on Weblogs and Social Media*, July 2011.

[122] V. Hatzivassiloglou and K. R. McKeown. Predicting the Semantic Orientation of Adjectives. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 174–181, July 1997.

[123] S.-M. Kim and E. Hovy. Determining the Sentiment of Opinions. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04. Association for Computational Linguistics, Aug. 2004.

[124] B. Li, K. Zhou, W. Gao, X. Han, and L. Zhou. Attention-based LSTM-CNNs for uncertainty identification on Chinese social media texts. In *The 2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, pages 609–614, Dec. 2017.

[125] R. P. Schumaker and H. Chen. A quantitative stock prediction system based on financial news. *Information Processing and Management*, 45(5):571–583, May 2009.

[126] Y. Wang, M. Huang, and L. Zhao. Attention-based LSTM for Aspect-level Sentiment Classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, Nov. 2016.

[127] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. Manandhar. SemEval-2014 Task 4: Aspect Based Sentiment Analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEvalval 2014)*, pages 27–35, Aug. 2014.

[128] S. Poria, E. Cambria, and A. Gelbukh. Aspect extraction for opinion mining with a deep convolutional neural network. *Knowledge-Based Systems*, 108:42–49, Sept. 2016.

[129] C. Dos Santos and M. Gatti. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, Aug. 2014.

[130] X. Glorot, A. Bordes, and Y. Bengio. Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 513–520, 2011.

[131] X. Zhou, X. Wan, and J. Xiao. Attention-based LSTM Network for Cross-Lingual Sentiment Classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 247–256, Nov. 2016.

[132] K. Shuang, Z. Zhang, H. Guo, and J. Loo. A sentiment information Collector–Extractor architecture based neural network for sentiment analysis. *Information Sciences*, 467:549–558, Oct. 2018.

[133] Y. Ma, H. Peng, and E. Cambria. Targeted Aspect-Based Sentiment Analysis via Embedding Commonsense Knowledge into an Attentive LSTM. In *Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5876–5883, 2018.

[134] M. Pontiki, D. Galanis, H. Papageorgiou, S. Manandhar, and I. Androutsopoulos. SemEval-2015 Task 12: Aspect Based Sentiment Analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495, June 2015.

[135] M. Saeidi, G. Bouchard, M. Liakata, and S. Riedel. SentiHood: Targeted Aspect Based Sentiment Analysis Dataset for Urban Neighbourhoods. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1546–1556, Dec. 2016.

[136] A. Severyn and A. Moschitti. Twitter Sentiment Analysis with Deep Convolutional Neural Networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 959–962, Aug. 2015.

[137] S. Yoo, J. Song, and O. Jeong. Social media contents based sentiment analysis and prediction system. *Expert Systems with Applications*, 105:102–111, Sept. 2018.

[138] C. Baziotis, N. Pelekis, and C. Doulkeridis. DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, 2017.

[139] S. Rosenthal, N. Farra, and P. Nakov. SemEval-2017 Task 4: Sentiment Analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518, Aug. 2017.

[140] V. Loia and S. Senatore. A fuzzy-oriented sentic analysis to capture the human emotion in Web-based content. *Knowledge-Based Systems*, 58:75–85, Mar. 2014.

[141] L. Barbaglia, S. Consoli, and S. Manzan. Monitoring the business cycle with fine-grained, aspect-based sentiment extraction from news. In *Mining Data for Financial Applications*, volume 11985 of *LNAI*, pages 101–106. Springer, 2020.

[142] O. Irsoy and C. Cardie. Opinion Mining with Deep Recurrent Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 720–728, Oct. 2014.

[143] P. Liu, S. Joty, and H. Meng. Fine-grained Opinion Mining with Recurrent Neural Networks and Word Embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1433–1443, Sept. 2015.

[144] E. Gilbert and K. G. Karahalios. Widespread worry and the stock market. In *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*, pages 58–65, May 2010.

[145] V. Niederhoffer. The Analysis of World Events and Stock Prices. *The Journal of Business*, 44(2): 193–219, Apr. 1971.

[146] B. Wuthrich, V. Cho, S. Leung, D. Permunetilleke, K. Sankaran, and J. Zhang. Daily stock market forecast from textual web data. In *International Conference on Systems, Man, and Cybernetics*, volume 3, pages 2720–2725, Oct. 1998.

[147] W. Antweiler and M. Z. Frank. Do Us Stock Markets Typically Overreact to Corporate News Stories? *SSRN Electronic Journal*, 2006.

[148] R. P. Schumaker, Y. Zhang, C.-N. Huang, and H. Chen. Evaluating sentiment in financial news articles. *Decision Support Systems*, 53(3):458–464, June 2012.

[149] Y. Gurin, T. Szymanski, and M. T. Keane. Discovering News Events That Move Markets. In *Intelligent Systems Conference*, Sept. 2017.

[150] K. R. Ahern and D. Sosyura. Who Writes the News? Corporate Press Releases during Merger Negotiations. *The Journal of Finance*, 69(1):241–291, Feb. 2014.

[151] C. Vega. Stock price reaction to public and private information. *Journal of Financial Economics*, 82(1):103–133, Oct. 2006.

[152] Y. Hu, K. Liu, X. Zhang, L. Su, E. W. T. Ngai, and M. Liu. Application of evolutionary computation for rule discovery in stock algorithmic trading: A literature review. *Applied Soft Computing*, 36: 534–551, Nov. 2015.

[153] P. C. Tetlock. Giving Content to Investor Sentiment: The Role of Media in the Stock Market. *The Journal of finance*, 62(3):1139–1168, June 2007.

[154] Y. Yu, W. Duan, and R. Cao. The impact of social and conventional media on firm equity value: A sentiment analysis approach. *Decision Support Systems*, 55(4):919–926, Nov. 2013.

[155] J. Bukovina. Social media big data and capital markets—An overview. *Journal of Behavioral and Experimental Finance*, 11:18–26, Sept. 2016.

[156] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Oct. 2013.

[157] Z. Da, J. Engelberg, and P. Gao. In Search of Attention. *The Journal of Finance*, 66(5):1461–1499, Sept. 2011.

[158] B. Weng, M. A. Ahmed, and F. M. Megahed. Stock market one-day ahead movement prediction using disparate data sources. *Expert Systems with Applications: An International Journal*, 79: 153–163, Aug. 2017.

[159] R. Xiong, E. P. Nichols, and Y. Shen. Deep Learning Stock Volatility with Google Domestic Trends. *arXiv:1512.04916*, Dec. 2015.

[160] M. S. Drake, D. T. RoulStone, and J. R. Thornock. Investor Information Demand: Evidence from Google Searches Around Earnings Announcements. *Journal of Accounting Research*, 50(4): 1001–1040, Sept. 2012.

[161] M. Makrehchi, S. Shah, and W. Liao. Stock Prediction Using Event-Based Sentiment Analysis. In *Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 01, pages 337–342, Nov. 2013.

[162] K. Cortis, A. Freitas, T. Daudert, M. Huerlimann, M. Zarrouk, S. Handschuh, and B. Davis. SemEval-2017 Task 5: Fine-Grained Sentiment Analysis on Financial Microblogs and News. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 519–535, 2017.

[163] T. O. Sprenger, A. Tumasjan, P. G. Sandner, and I. M. Welpe. Tweets and Trades: the Information Content of Stock Microblogs. *European Financial Management*, 20(5):926–957, Nov. 2014.

[164] T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing Contextual Polarity in Phrase-level Sentiment Analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 347–354, Oct. 2005.

[165] C. Oh and O. Sheng. Investigating Predictive Power of Stock Micro Blog Sentiment in Forecasting Future Stock Price Directional Movement. In *International Conference on Information Systems*, volume 4, pages 2860–2877, Dec. 2011.

[166] B. Li, K. C. C. Chan, C. Ou, and S. Ruifeng. Discovering public sentiment in social media for predicting stock movement of publicly listed companies. *Information Systems*, 69:81–92, Sept. 2017.

[167] O. Oh, M. Agrawal, and H. R. Rao. Community Intelligence and Social Media Services: A Rumor Theoretic Analysis of Tweets During Social Crises. *MIS Quarterly*, 37(2):407–426, June 2013.

[168] N. Oliveira, P. Cortez, and N. Areal. On the Predictability of Stock Market Behavior Using StockTwits Sentiment and Posting Volume. In *Progress in Artificial Intelligence*, pages 355–365, 2013.

[169] E. J. Ruiz, V. Hristidis, C. Castillo, A. Gionis, and A. Jaimes. Correlating Financial Time Series with Micro-blogging Activity. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, pages 513–522, Feb. 2012.

[170] A. Bermingham, M. Conway, L. McInerney, N. O'Hare, and A. F. Smeaton. Combining Social Network Analysis and Sentiment Analysis to Explore the Potential for Online Radicalisation. In *2009 International Conference on Advances in Social Network Analysis and Mining*, pages 231–236, July 2009.

[171] W. Yang, D. Lin, and Z. Yi. Impacts of the mass media effect on investor sentiment. *Finance Research Letters*, 22:1–4, Aug. 2017.

[172] S. Agarwal, S. Kumar, and U. Goel. Stock market response to information diffusion through internet sources: A literature review. *International Journal of Information Management*, 45:118–131, Apr. 2019.

[173] T. Hollis. Deep Learning Algorithms Applied to Blockchain-Based Financial Time Series. Technical report, University of Manchester, 2018.

[174] S. Kar, S. Maharjan, and T. Solorio. RiTUAL-UH at SemEval-2017 Task 5: Sentiment Analysis on Financial Data Using Neural Networks. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 877–882, Aug. 2017.

[175] T. Cabanski, J. Romberg, and S. Conrad. HHU at SemEval-2017 Task 5: Fine-Grained Sentiment Analysis on Financial Data using Machine Learning Methods. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 832–836, Aug. 2017.

[176] G. Zhiqiang, W. Huaiqing, and L. Quan. Financial time series forecasting using LPP and SVM optimized by PSO. *Soft Computing*, 17(5):805–818, May 2013.

[177] D. Ghosal, S. Bhatnagar, M. S. Akhtar, A. Ekbal, and P. Bhattacharyya. IITP at SemEval-2017 Task 5: An Ensemble of Deep Learning and Feature Based Models for Financial Sentiment Analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 899–903, Aug. 2017.

[178] Y. Mansar, L. Gatti, S. Ferradans, M. Guerini, and J. Staiano. Fortia-FBK at SemEval-2017 Task 5: Bullish or Bearish? Inferring Sentiment towards Brands from Financial News Headlines. *arXiv:1704.00939*, Apr. 2017.

[179] L. Pivovarova, L. Escoter, A. Klami, and R. Yangarber. HCS at SemEval-2017 Task 5: Sentiment Detection in Business News Using Convolutional Neural Networks. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 842–846, Aug. 2017.

[180] A. Moore and P. Rayson. Lancaster A at SemEval-2017 Task 5: Evaluation metrics matter: predicting sentiment from financial news headlines. *arXiv:1705.00571*, May 2017.

[181] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[182] R. Sathya and A. Abraham. Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification. *International Journal of Advanced Research in Artificial Intelligence*, 2 (2):34–38, 02 2013.

[183] M. Längkvist, L. Karlsson, and A. Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, June 2014.

[184] R. Agrawal, T. Imieliński, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, 22(2):207–216, June 1993.

[185] P. Baldi. Autoencoders, Unsupervised Learning, and Deep Architectures. *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 27:37–50, July 2011.

[186] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3), July 2009.

[187] A. K. Jain, M. N. Murty, and P. J. Flynn. Data Clustering: A Review. *ACM computing surveys (CSUR)*, 31(3):264–323, Sept. 1999.

[188] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, May 1996.

[189] Y. Bengio, A. Courville, and P. Vincent. Representation Learning: A Review and New Perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, Mar. 2013.

[190] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[191] R. Collobert and J. Weston. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167, July 2008.

[192] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, Dec 1943.

[193] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[194] P. Cunningham and S. Delany. k-Nearest Neighbour Classifiers. *Multiple Classifier Systems*, 34 (8):1–17, Mar. 2007.

[195] S. Fine, Y. Singer, and N. Tishby. The Hierarchical Hidden Markov Model: Analysis and Applications. *Machine Learning*, 32(1):41–62, Jul 1998.

[196] T. Joachims. Text categorization with Support Vector Machines: Learning with many relevant features. *European Conference on Machine Learning (ECML)*, 1398:137–142, 1998.

[197] A. McCallum and K. Nigam. A Comparison of Event Models for Naive Bayes Text Classification. In *Learning for Text Categorization: Papers from the 1998 AAAI Workshop*, volume 752, pages 41–48, July 1998.

[198] J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1(1):81–106, Mar 1986.

[199] I. A. Basheer and M. Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43(1):3–31, Dec. 2000.

[200] F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. *Psychological Review*, 65(6):386–408, 1958.

[201] D. Svozil, V. Kvasnicka, and J. Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 39(1):43–62, Nov. 1997.

[202] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A Convolutional Neural Network for Modelling Sentences. *arXiv:1404.2188*, Apr. 2014.

[203] G. Tolias, R. Sicre, and H. Jégou. Particular object retrieval with integral max-pooling of CNN activations. *International Conference on Learning Representations*, 2015.

[204] M. Boden. A guide to recurrent neural networks and backpropagation. *the Dallas project*, 2002.

[205] P. J. Werbos. Backpropagation Through Time: What It Does and How to Do It. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[206] S. Hochreiter. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6 (2):107–116, Apr. 1998.

[207] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *Proceedings of the 30th International Conference on International Conference on Machine Learning*, 28:1310–1318, June 2013.

[208] R. Pascanu, T. Mikolov, and Y. Bengio. Understanding the exploding gradient problem. *arXiv:1211.5063*, Nov. 2012.

[209] J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, Mar. 1990.

[210] M. I. Jordan. Serial Order: A Parallel Distributed Processing Approach. In *Advances in Psychology*, volume 121, pages 471–495. Elsevier, Jan. 1997.

[211] A. Karpathy. The Unreasonable Effectiveness of Recurrent Neural Networks. `https://www.bibsonomy.org/bibtex/236c6041ff8f00e4d94b521b3c5ebf032/andreashdez`, May 2015. URL `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`.

[212] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, July 2017.

[213] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[214] O. Vinyals and Q. V. Le. A Neural Conversational Model. *arXiv:1506.05869*, June 2015.

[215] D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473*, 2014.

[216] O. Firat, K. Cho, and Y. Bengio. Multi-Way, Multilingual Neural Machine Translation with a Shared Attention Mechanism. *arXiv:1601.01073*, 2016.

[217] M.-T. Luong, H. Pham, and C. D. Manning. Effective Approaches to Attention-based Neural Machine Translation. *arXiv:1508.04025*, Aug. 2015.

[218] R. Nallapati, B. Zhou, C. dos Santos, Ç. Gulì‡lçehre, and B. Xiang. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In *The 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, pages 280–290, Aug. 2016.

[219] A. M. Rush, S. Chopra, and J. Weston. A Neural Attention Model for Abstractive Sentence Summarization. *arXiv:1509.00685*, Sept. 2015.

[220] S. Hochreiter. Recurrent Neural Net Learning and Vanishing Gradient. *International Journal Of Uncertainity, Fuzziness and Knowledge-Based Systems*, 6(2):107–116, 1998.

[221] A. Graves, G. Wayne, and I. Danihelka. Neural Turing Machines. *arXiv:1410.5401*, Oct. 2014.

[222] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010, Dec. 2017.

[223] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A Structured Self-attentive Sentence Embedding. *International Conference on Learning Representations*, Mar. 2017.

[224] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer Normalization. *arXiv:1607.06450*, 2016.

[225] G. Letarte, F. Paradis, P. Giguère, and F. Laviolette. Importance of Self-Attention for Sentiment Analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 267–275, 2018.

[226] S. van der Walt, S. C. Colbert, and G. Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science and Engineering*, 13(2):22–30, Mar. 2011.

[227] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *The Journal of Machine Learning Research*, 12:2825–2830, Nov. 2011.

[228] W. Mckinney. pandas: a Foundational Python Library for Data Analysis and Statistics. *Python High Performance Science Computer*, 01 2011.

[229] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56): 1929–1958, 2014.