Università
Ca'Foscari
Venezia

Master's Degree
in Economics and Finance

Final Thesis

Explanatory power of GARCH models using news-based investor sentiment: Applications of LSTM networks for text classification

Supervisor:
Ch. Prof. Loriana Pelizzon
Supervisor:
Ch. Prof. Marco Corazza

Graduand:
Gianluca Anese
Matricolation number: 872000

Academic Year:
2018 / 2019

Acknowledgements

First of all, I would like to thank my supervisors, Prof. Loriana Pelizzon and Prof. Marco Corazza, for their continuous help and guidance through each stage of my thesis.

My sincere thanks also goes to Prof. Michele Costola for his essential support.

In addition, I would like to express my deep gratitude to Ferdinando Gueli and Anthony H. Mannara for offering me the opportunity to work at the Italian Trade Agency in New York and inspiring my interest in innovative topics.

Abstract

Many authors have shown that investors are not fully rational, as the traditional Efficient Markets Hypothesis suggests, and that investor sentiment can have an impact on stock prices. As investor sentiment is not directly measurable, different proxies have been used by researchers. In addition, progress in Natural Language Processing has contributed to the development of new sentiment measures based on text sources obtained by news providers and social media. This work deals with a classification problem on financial news data and defines a reliable proxy for investor sentiment using both dictionary – based and supervised Machine Learning techniques. In particular, LSTMs networks are adopted. The resulting sentiment proxies are used as exogenous variables in the mean and variance equations of a Generalized Autoregressive Conditional Heteroskedasticity model in order to prove the existence of a relationship among them and stock returns and among them and volatility.

Contents

List of Tables

List of Figures

Introduction

According to the Efficient market hypothesis, in presence of no frictions, assets are always traded fairly. Indeed, rational investors price assets by discounting all future cash flows and thus prices incorporate all available information. Even in presence of some irrational investors, their impact on asset prices would be offset by rational ones through arbitrage. Therefore, the best forecast about the future price of an asset is its current price.

In its influential paper [17], Eugene Fama presents a review of empirical tests of market efficiency considering different information sets. Under weak market efficiency, information is represented by stock prices and volumes only, while in the semi-strong form, also other publicly available information is considered, such as earnings announcements and stock splits. Finally, in presence of strong market efficiency, prices reflect all public and private information so that no investor "has monopolistic access to some information" [17]. The empirical evidence presented in his paper seems to support the first two forms of efficiency, while the latter, as expected, produces contradictory results and it is considered as benchmark to test the efficiency hypothesis under the other two information sets.

During the last thirty years, several authors have tried to prove an alternative scenario where prices don't fully reflect all available information, and behavioral finance has emerged as a new approach, assuming an important role, as demonstrated by the award of the Nobel prize in Economic Sciences to the behavioral economist Richard Thaler in 2017. For behavioral economists, investors are not fully rational, and their investment decisions are influenced by emotions.

In their "Survey of Behavioral Finance" [9], Barberis and Thaler describe the psychological aspects that characterize irrational investors, namely beliefs and preferences. In particular, they report evidence showing that people are often overconfident in their judgments and too optimistic about their abilities and prospects. In addition, the likelihood of events is sometimes estimated based on representativeness, that is making simplistic assumptions that consider only incomplete features of a given population. Other times, instead, investors are subject to conservatism, relying heavily on prior convictions without updating them with new sample information. They stick to their beliefs even in presence of new evidence and anchor on their initial estimate. Furthermore, people rely too much on their memory, which might be biased, as not all past events are equally available in their memory.

With respect to investors' preferences, Barberis and Thaler [9] explain that, differently from the traditional expected utility framework, losses are given more weight than gains, investors are more sensitive to changes in probability at high probability levels, and the way an investment is presented can influence the final decision. In addition, people show a certain aversion for unfamiliar situations, where they don't know the probability distribution of an investment.

According to traditional finance, even in presence of irrational investors, arbitrageurs would offset their effect and bring the asset price back to its fundamental value. However, as explained by Barberis and Thaler [9], arbitrage possibility is often limited. Indeed, short sale constraints and implementation costs make this tool imperfect. In addition, it might take a while for asset prices to reach their fundamental value and, in the meantime, arbitrageurs might be forced to liquidate their positions, incurring in big losses. Therefore, in presence of limits to arbitrage, prices might deviate from the fundamental value of the asset, if irrational investors, also called noise traders, express wrong and correlated beliefs.

Given the increasing influence of behavioral finance, many academicians have tried to prove the possibility that market sentiment, defined by Malcolm Baker and Jeffrey Wurgler [7] as "a belief about future cash flows and investment risks that is not justified by the facts at hand" impacts stock prices.

A big challenge about this topic comes from the fact that measuring market sentiment is not straightforward. Researchers have tried to overcome this problem with the use of different proxies. As Zhang [75] points out, there is no commonly accepted definition of investor sentiment and, together with the "propensity to trade on noise rather than information", which is similar to the definition by Baker and Wurgler [7], one may refer to the same term as "investor optimism or pessimism". Following this second definition, professionals have introduced indicators based on surveys to measure the propensity of investors towards long or short positions.

In more recent years, the advent of Big Data analytics and the improvements in Artificial Intelligence have given the possibility to access additional unstructured data sources and integrate the already publicly available information presented by Eugene Fama [17], earnings announcements and dividend splits. For instance, companies can now predict future revenues relying on satellite images, web searches and product reviews. Useful sources to determine the attitude of investors towards a given asset are the multitude of financial news articles and social media contents published every day, accessible in the past only at high costs of time. Indeed, Natural Language Processing allows investors to save time and effort in reading each article and assess its content in real time.

Natural Language Processing can be defined as a field of Artificial Intelligence that allows machines to read and "understand" the content of a text document. It is therefore possible to assess if a text is positive or negative with respect to a given target.

The aim of this work is dealing with a text classification problem on financial news articles in order to derive a reliable proxy for investor sentiment. The classification problem was addressed following both dictionary-based and machine learning techniques. The former was handled using different dictionaries available in the financial literature, while the latter through a supervised machine learning approach that uses a particular type of Recurrent Neural Network (RNN), the Long Short Term Memory (LSTM) model. The news-based proxies obtained from this work were used individually as exogenous variables in the mean and variance equations of a GARCH model fitted on the log returns of the S&P 500 Index. The purpose is trying to improve the explanatory power of the Generalized Autoregressive Conditional Heteroskedasticity model and prove the existence of a relationship between investor sentiment and stock returns or volatility.

Before developing my own proxies for investor sentiment, I started with a preliminary analysis of the relationship between some news-based sentiment proxies provided by Bloomberg Terminal. The aim was assessing the reliability of existent sentiment variables and the research led to some interesting results, which are presented in the Appendix.

Chapter 1 describes the main proxies used in literature to define market sentiment and introduces the main techniques of Natural Language Processing. Chapter 2 presents the model I have developed using both dictionary-based and supervised machine learning techniques. In Chapter 3, I introduce the sentiment proxies derived from the previous models as exogenous variables in the mean and variance equations of a GARCH model fitted on the intraday returns of the S&P 500 Index. Finally, the Appendix shows the results of the analysis of the daily Sentiment Indicators provided by Bloomberg Terminal on the basis of financial news and Twitter contents.

1 Measuring Investment Sentiment

1.1 Investor sentiment proxies

As measuring market sentiment is not easy work, many academicians have tried to use different proxies. Baker and Wurgler [7] relate the use of different proxies to the specific phase of the chain that connects a change in investor sentiment to its effects on securities trades. The first phase of the chain is represented by investors beliefs and useful proxies can be obtained directly by surveying investors to find out their expectations about stock price appreciation or depreciation. The following phases correspond to possible deviations of asset prices from the fundamental value and even consequent reactions of corporate executives, such as capital structure adjustments through equity issues or stocks repurchases, on the basis of stock prices being overvalued or undervalued. In this case, information about investor sentiment can be retrieved indirectly from market-based sources, like trading volume.

One of the drawbacks of survey-based proxies is the possible discrepancy between what respondents say and the actual investment decisions they make. In addition, observations for these measures are not so frequent, as they are registered weekly, monthly, or even quarterly. Market-based proxies, instead, incorporate investors behaviors measuring how they react to their belief and some of them are collected at higher frequency.

1.1.1 Survey-based proxies

A group of four Stock Market Confidence Indexes [44] have been produced by the International Center of Finance at Yale School of Management, based on survey data on investors behavior in the United States continuously collected by Professor Robert Shiller since 1989. Among them, the *One-Year Confidence Index* represents the percentage of respondents that expect a positive variation in the Dow Jones Index in the following year, while the *Buy-on-Dips Confidence Index* shows the percentage of the population expecting a rebound the day after a hypothetical 3% drop of the Dow Jones Index. The *Crash Confidence Index*, instead, denotes the proportion of interviewees that attributes a low probability to a catastrophic stock market crash in the United States. The last index is the *Valuation Confidence Index*, which represents the percentage of investors believing stocks are not overvalued with respect to their fundamental value. Yale University provides parallel indexes for Japanese investors, using surveys conducted since 1989 by Professor Yoshiro Tsutsui of Osaka University, Fumiko Kon-Ya of the Japan Securities Research Institute, and Akiko Kamesaka of Aoyama Gakuin University.

Another Survey-based proxy, the *Index of Investor Optimism* [59], has been produced by UBS and the Gallup Organization from 1996 to 2007 and is made available by the Roper Center for Public Opinion Research at Cornell University. The interviewees are households holding an investment portfolio of $10,000 or more, and the index summarizes the answers to seven questions regarding their own expectation about the overall investment environment and their personal financial situation.

In order to show that sentiment of Wall Street strategists and that of individual investors are uncorrelated, Fisher and Statman [18] make use of three survey-based sentiment proxies. To account for large investors sentiment, they use *Merrill Lynch* data about sentiment of Wall street sell-side strategists, available since 1985. It is compiled monthly and is useful as a contrary indicator[1]. For medium-sized investors, they refer to *Investors Intelligence Sentiment Index*, produced by the investment services company Chartcraft, which surveys more than 130 investment newsletter writers and monitors their recommendations. According to Investopedia [33], this index is a contrarian indicator, meaning it predicts a reversal of the stock trend so that traders can take opposite positions to that of average investors. Data has been compiled weekly since 1964. A solution for small investors comes from the *American Association of Individual Investors* [29], which proposed a measure based on weakly surveys in 1987. As its website points out, this index has been widely adopted by professionals, and the weekly survey results are published in important financial publications, such as Barron's and Bloomberg.

1.1.2 Market-based proxies

Academicians have worked on various proxies for market sentiment that are retrieved from stock prices, such as liquidity, volatility, equity issues.

Baker and Stein [3], in their effort to explain why increases in liquidity predict lower subsequent returns, highlight that *trading volume* and *liquidity* can be good proxies for investment sentiment. Indeed, in presence of short-sale constraints, pessimistic irrational investors stay out of the market, while optimistic ones are active in the market, increasing trading volumes. Under the assumption that noise traders underreact to order flows, that represent the imbalance between bids and

[1]A contrary indicator is a type of indicator that can be used for contrarian investing strategies, where investors make opposite decisions with respect to the market. Contrary indicators are useful to predict a reversal of the stock trend.

asks, as soon as they enter the market, they create more liquidity, being their views different from those of rational investors. Therefore, liquidity and volumes can provide a good proxy for the presence of irrational investors with positive sentiment. Trading volume is also used by Chuang, Ouyang and Lo [12], who show the impact of investor sentiment on excess returns in Taiwan stock market. Lee and Swaminathan [38], in their attempt to show that past trading volume, measured by average daily turnover[2], predicts both the magnitude and the persistence of price momentum, find evidence that variations in trading volume measure fluctuations in investor sentiment. Interesting empirical results have been obtained by Liu [40]. Using two survey-based measures, the Investor's Intelligence Index and that of the American Association of Individual Investors, as well as a liquidity measure, he shows that investor sentiment Granger-causes market liquidity. Similar results have been obtained for trading volume.

Kumar and Lee (2006) examine the correlation among retail investors' trades and note that they tend to buy or sell stocks in concert with each other. They build different portfolios of stocks and show that their *buy-sell imbalances*[3] behave similarly and can represent a proxy for retail investor sentiment.

Another proxy for market sentiment comes from *dividend premiums*, defined by Baker and Wurgler [4] as "difference between the average market-to-book ratio of dividend payers and non-payers", which are expected to be inversely related to sentiment. Baker and Wurgler prove that managers try to cater to investors high demand for dividend, when they believe this will lead to dividend premium. For Baker and Wurgler, one of the reasons for variations in investors demand for dividend paying stocks over time, is their risk tolerance. Investors may demand dividend paying stocks in case of pessimistic sentiment, as these stocks are perceived as safer, and prefer non dividend stocks, assumed to have higher growth potential, in case of optimistic sentiment.

Risk tolerance is also the reason behind the use of *mutual fund flows* as proxy for sentiment. Indeed, examining the holdings of the mutual funds, it is possible to assess whether investors are moving towards mutual funds that invest in less risky or riskier assets. Frazzini and Lamont [20] note that irrational investors tend to transfer money from mutual funds with low recent returns to those with high past returns, even though this practice is disadvantageous and reduces wealth in the long run. They use flows from a mutual fund to another to measure individual market sentiment towards particular stocks and show that this variable can predict future stock returns. Before this work, Brown et al. [11] provide evidence that daily

[2]Turnover is defined by Lee and Swaminathan [38] as "the ratio of the number of shares traded to the number of shares outstanding".

[3]Buy-sell imbalance is defined as difference between buying and selling volume over the total trading volume

mutual fund flows can be considered as proxy for investor sentiment. In particular, they find a strong negative correlation between Japanese funds explicitly named "bull", that bet on the rise of Japanese stock market, and those named "bear".

Other studies deal with mutual funds, but with specific focus on *mutual funds premium* (or discount). Zweig [76] proves that unexpected changes in investor expectations can have an impact on the fund premium (or discount), defined as the difference between fund market price and market value of the stocks held by the specific closed-end fund, expressed in percentage terms with respect to the market value of the stocks. Zweig recognizes that irrational investors might over-discount the value of new information and changes in closed-end funds premiums can be used as measure of sentiment. Lee and Thaler [37] point out that closed-end funds are owned primarily by individual investors and are, on average, traded at a discount, due to the negative correlation between investor sentiment and funds discount, which bring about additional risk to that of the assets held by the funds. Neal and Wheatley [53] use closed-end funds discount as proxy for sentiment and find that it can significantly predict the size premium, that is the difference in returns between small and large firms, and the returns of small firms, which are held mainly by individual investors.

Bandopadhyaya and Jones [8] provide a comparative analysis of other two sentiment measures computed by the CBOE, *implied volatility index (VIX)* and *Put-Call volume ratio*, showing that the latter explains better variations in the S&P index that are not explained by economic factors. VIX, also called the Fear Index, is a measure of implied volatility retrieved by S&P 500 index options and reaches high values when sentiment is low, because investors are assumed to be risk averse. Put-Call volume ratio consists in the ratio of put options and call options trading volume. Buyers of put options are considered to be pessimistic, while buyers of call options are optimistic, thus when this ratio is greater than one, sentiment is assumed to be negative, and when it is lower than one it represents positive sentiment. The PCR is widely used by practitioners as a contrarian indicator.

Some authors, such as Baker and Wurgler [7], have used *IPO first day returns* to measure market sentiment. The fact that IPOs are often followed by a strong reaction in the stock price and produce abnormal first day returns is widely treated by academicians. The first evidence comes from Stoll and Curley [68], who unsuccessfully try to prove the existence of an equity gap, defined as difference in the cost of funds for small and large firms with same riskiness, but demonstrate that returns are particular high following IPOs. To conduct this analysis, they measure the cost of funds as the ex-post rate of return of small firms which go public, and they find that stocks price increases substantially between the initial offering date and the first market date, far more than the S&P Index during the same time period.

1.1.3 Corporate finance-based proxies

Corporate finance is another part of the chain of reactions triggered by variations in market sentiment. Baker and Stein [3], for instance, while using liquidity as proxy for investor sentiment, note that managers might easily time the market for Seasoned Equity Offerings (SEOs)[4], following waves of high liquidity. Executives have more information about the real growth potential of stocks and do time the market when the opportunity comes, as proved by Graham and Harvey [24], who survey 392 anonymous CFOs and ask what factors affect managers decisions about issuing common stocks. They find that 67% of CFOs consider how much stocks are undervalued or overvalued by the market and 63% admit a recent increase in price might be a good reason for issuing new shares.

Baker and Wurgler [5] find that the *rate of new equity issues to the total issues of equity and debt* predicts stock market returns and that this negative relation is explained by market timing. Indeed, in presence of correlated sentiment, irrational investors can cause numerous stocks to be overvalued, leading these firms to make similar financing decisions. Variations in the aggregate level of equity issues are explained by mispricing and can be used to measure variations in market sentiment. In addition, Baker and Wurgler show that this variable is correlated with other proxies for investor sentiment, such as insider sales of stocks, value weighted closed-end fund discount and changes in consumers' expectations of business conditions.

Another situation where managers show their ability to time the market is Initial Public Offering. Michelle Lowry [42] show that investor optimism is among the major determinants of IPO waves. She finds that investor sentiment, measured by closed-end funds discount and post-IPO market returns, explain IPO volumes with high significance. Also Baker and Wurgler [7] include *IPO volumes* among the proxies for market sentiment and note that investment bankers consider specific windows of opportunity for IPOs, when investor sentiment is higher.

1.1.4 Composite proxies

As all these proxies are imperfect and no commonly accepted one has been found, some authors have produced proxies which combine some of the measures already listed.

One of the most famous works is that of Baker and Wurgler [6], who study the effect of market sentiment on the cross section of stock returns, finding that it has a higher impact on small, young, volatile, unprofitable, non-dividend paying and distressed stocks. For this aim, they make use of a *composite sentiment index* that involves six different proxies: the closed-end fund discount ($CEFD$), the

[4]Seasoned Equity Offerings are equity issues made by listed companies. These companies have already held their Initial Public Offering and seek new equity capital to fund their operations.

NYSE share turnover ($TURN$), the number of IPOs ($NIPO$), the average first-day returns on IPOs ($RIPO$), the equity share in new issues (S) and the dividend premium (P^{D-ND}). To separate the sentiment component from the idiosyncratic one, principal component analysis [34] has been applied to all the proxies and their lags, so that a first-stage index[5] with 12 coefficients, named loadings, can be determined. Then, Baker and Wurgler compute the correlation between the first-stage index and the 12 proxies and obtain the final composite index through a principal component analysis of the correlation matrix of six of these variables. Indeed, for each proxy's lead or lag, they select only the one with highest correlation with the first-stage index. Coefficients have been re-scaled so that the index has unit variance and the results are presented in Equation (1).

$$\begin{aligned} SENTIMENT_t = &-0.241 CEFD_t + 0.242 TURN_{t-1} + 0.253 NIPO_t \\ &+ 0.257 RIPO_{t-1} + 0.112 S_t - \ 0.283 P^{D-ND}_{t-1} \end{aligned} \tag{1}$$

In a different paper, Baker and Wurgler [7], use the same variables for their *operational sentiment index*, but they follow a different procedure. They point out that these variables, might still include some common variability that is related to economic fundamentals, instead of market sentiment. Therefore, they regress each proxy on a set of macroeconomic variables and keep the residuals. Then, they compute the average of all these residuals and obtain the final index.

Exploiting the six proxies used by Baker and Wurgler [6], Huang et al. [28] produce a new *aligned investor sentiment index.* In their paper, they explore the impact of investor sentiment on stock market return, showing that their index performs better than many macroeconomic variables and is more efficient that Baker and Wurgler's index itself. They note that some of the common variation captured by principal component analysis is not representative of investor sentiment, leading to errors in market return predictions. Therefore, they apply partial least square (PLS) method using a two-stage approach. First, they assume that realized future returns R_{t+1} are explained by the unobservable investor sentiment S_t, as in Equation (2).

$$R_{t+1} = \alpha + \beta S_t + \epsilon_{t+1} \tag{2}$$

Then, they use OLS method to regress each proxy $x_{i,t}$ on R_{t+1}, as shown in Equation (3).

$$x_{i,t} = \pi_{i,0} + \pi_{i,1} R_{t+1} + u_{i,t}, \quad \text{for} \quad t = 1, .., T \tag{3}$$

Finally, they use the slope coefficient $\pi_{i,1}$ (also called factor loading), which measures the sensitivity of each proxy to investor sentiment (expressed by the instrument R_{t+1}), in a second OLS regression. They run T cross sectional OLS regressions, one for each time period, using the proxies $x_{i,t}$ as dependent variables and

[5]The first-stage index is represented by the linear combination of twelve proxies (all six proxies and their lagged values) that maximizes the variance

the corresponding factor loadings as explanatory variables.

$$x_{i,t} = c_t + S_t^{PLS}\hat{\pi}_i + v_{i,t}, \quad \text{for} \quad i = 1, .., N \tag{4}$$

The slope coefficient S_t^{PLS} represents the Aligned investor sentiment index.

Rupande et al. [61] explore the relation between investor sentiment and stock return volatility on the Johannesburg Stock Exchange market and propose a daily sentiment composite index based on a set of proxies. Given the constraint on the availability of data, they use the exchange rate between the South African Rand and the US dollar (Exch), the prime rate (Prime)[6], the 90-day Treasury Bill rate (Treasury), the repo rate (Repo)[7], the trading volume (Volume), the volume-weighted average price changes (Vwap), the South African Volatility Index (Savi). In their paper, they follow the same procedure of Baker and Wurgler [6] and use principal component analysis to produce InvSent, the index represented in Equation (5).

$$\begin{aligned} InvSent_t = &\theta_1 Exch_{t-\ 1/t} + \theta_2 Prime_{t-\ 1/t} + \theta_3 Treasury_{t-\ 1/t} + \theta_4 Repo_{t-\ 1/t} \\ &+ \theta_5 Volume_{t-\ 1/t} + \theta_6 Vwap_{t-\ 1/t} + \theta_7 Savi_{t-\ 1/t} \end{aligned} \tag{5}$$

1.2 Natural Language Processing for Sentiment Analysis

The proxies presented so far are correlated with market sentiment because they represent its effect on stock markets or corporate executive decisions. Investors, as well as companies and individuals, produce a huge quantity of data, and being able to analyze them can give access to additional important information to determine their sentiment. News articles and social media contents, for instance, may represent useful sources to assess investors' optimism or pessimism, either because investors write them and express their opinions in a less biased way than common surveys, or because they influence market sentiment changes.

Recent developments in Artificial Intelligence and Machine Learning techniques have contributed to uncover new patterns in huge amounts of structured and unstructured data and allow to process information at relatively low costs. Data is structured when it is organized in spreadsheets under a standardized format. Elements can be grouped together and classified on the basis of various features.

[6] According to Investopedia [31], the prime rate is defined as the average interest rate charged by the largest banks to their most creditworthy customers and it is based on the federal funds overnight rate.

[7] Repo rates are implied by Repurchase Agreements, defined by Investopedia [32] as a sort of short term borrowing where one party sells government bonds to another party and agrees to buy them back in the future.

However, most of the available data is unstructured [19] and can not be stored as standardized information. It is the case of qualitative information sources such as photos, text documents, satellite images, video and audio files. Natural language processing can provide a useful tool to assess the level of optimism of these text sources and determine new proxies for market sentiment. Academicians have explored two main approaches to Natural Language Processing of financial text sources: lexicon-based and Machine Learning approach.

In the next paragraphs I review the main papers concerning sentiment analysis with text data, using both lexicon and Machine Learning approach. I also present the finest pre-processing techniques that are commonly used to prepare raw data to be fed into the learning algorithms.

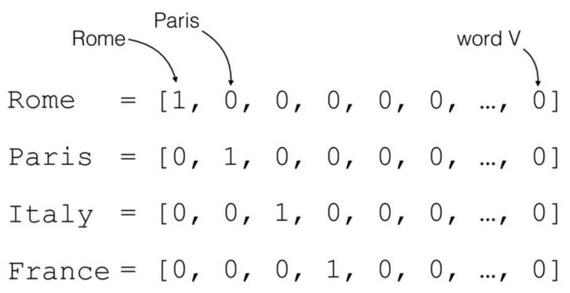
1.2.1 Pre-processing phase

Pre-processing is an essential phase in Natural Language Processing, because it transforms text data in a form that can be fed into Machine Learning algorithms. The first step is *tokenization*, consisting in splitting a text string into an ordered sequence of *tokens*, which after being pre-processed, will be the input of Natural Language Processing algorithm. A token is an ordered set of characters, which is given a particular meaning, and may include a word, a group of words or punctuation. Nassirtoussi et al. [52] divide pre-processing in three parts, namely feature selection, dimensionality reduction and feature representation.

Bag of Words is the most widely used *feature selection* method because of its simplicity and consists in dealing with each word expressed by the token as a single and separate observation, without considering repetitions and the order of the words. A similar technique consists in using n-grams in a Bag of Words model, that means combining two or more consecutive words in order to keep their joint meaning.

Another important part is *dimensionality reduction*, as a high number of features can worsen the efficiency of learning algorithms. Some of the most common techniques are stop-words removal and stemming. The former consists in removing all the words that do not add significant information to sentences, such as punctuation, articles or prepositions, while the latter is a process that reduces words to their root form, in order to limit dictionary size. Other solutions are defining a minimum occurrence limit, such that words with low frequency are discarded, and converting all letters to lower case.

The last part of the pre-processing phase is *feature representation*, that is expressing the features, or words in case of a bag of words model, as numeric values in order to feed them into the learning algorithm. Nassirtoussi et al. [52] mention *binary representation* and *Term Frequency-Inverse Document Frequency (TF-IDF)* as the most common techniques. In fact, the simplest representation method is certainly *one-hot vector encoding* that consists in creating an ordered dictionary that assigns an integer value to each word and then representing each word included in the documents as a sparse vector that has same dimension of the dictionary. All elements of the vector have zero values, except for the one corresponding to the specific word in the dictionary, which has value "one". Figure 1 shows an example of one-hot encoding, where each column corresponds to one word in the dictionary and each row to the specific word in the text. For instance, each time the word "Rome" appears in the corpus, this kind of representation associates the sparse vector corresponding to the word "Rome" to that particular word in the text. Sentences are represented as sequences of sparse vectors. The main drawbacks of one-hot encod-



```
Rome   = [1, 0, 0, 0, 0, 0, …, 0]
Paris  = [0, 1, 0, 0, 0, 0, …, 0]
Italy  = [0, 0, 1, 0, 0, 0, …, 0]
France = [0, 0, 0, 1, 0, 0, …, 0]
```

Figure 1: One-hot encoding.
Source: Medium. "Vector Representations of Text for Machine Learning".

ing [64] are connected to the large vocabulary size, the computational problems brought about by sparse vectors and the fact that it does not account for similarities among different words, as every feature is treated independently. *Embedding* [64] solves these problems by representing features, or words, in dense vector form and locating words with similar meaning close to each other in the vector space. Figure 2 shows that words with semantic relationship, such as Countries and their Capitals, are mapped close to each other in the vector space. Embedding can be performed using an *embedding layer* in the training phase of a Machine Learning process in order to adapt the embedding vector to the specific corpus, but it is also possible to use *pre-trained word embeddings*, such as Word2Vec [51] or GloVe [58]. For these reasons, embedding is widely used and, in recent times, it has overcome other methods such as TD-IDF.

Term Frequency-Inverse Document Frequency (TF-IDF) is a variation one-hot encoding. Instead of representing the encoding vector with binary values, and thus giving the same relevance to each feature, TF-IDF assigns a value that depends on the term frequency and the inverse frequency of the specific word in the corpus. To be clearer, each word i included in document j is still represented by a sparse vector, but the value of the element that corresponds to the specific word in the vocabulary is the product of term frequency and inverse document frequency, as in

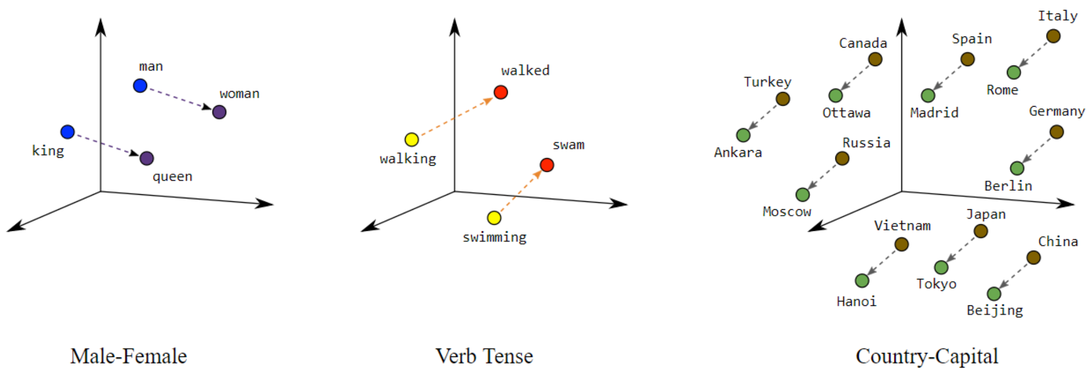


Figure 2: Word embedding.
Source: Google Developers. "Embeddings: Translating to a Lower-Dimensional Space."

Equation (6).

$$\begin{aligned} TF\text{-}IDF_{i,j} &= TF_{i,j} \times IDF_{i,j} \\ &= \frac{Nwords_{i,j}}{Totwords_j} \times log\left(\frac{Totdoc_j}{Ndoc_i}\right) \end{aligned} \tag{6}$$

Term frequency ($TF_{i,j}$) is the ratio between number of times a given word i appears in document j ($Nwords_{i,j}$) and the total number of words in document j ($Totwords_j$), while inverse document frequency ($IDF_{i,j}$) is the logarithm of the ratio between total number of documents ($Totdoc_j$) and number of documents that include the specific word i ($Ndoc_i$). This feature representation method takes into account the relevance of each word, on the basis of how often it appears in a document and how many documents include that word. Indeed, common words that appear very frequently in all documents do not add much information, but words that do not appear so often and are included with high frequency in a small number of documents are likely to be representative of those documents.

1.2.2 Lexicon approach

Lexicon-based approach to sentiment analysis consists in defining a list of words that are associated with a given sentiment state. Medhat, Hassan and Korashy [49] present two lexicon-based approaches, namely *dictionary-based* and *corpus-based* approach. Both of them start with a list of opinion words that is built manually. In dictionary-based approach, this list of words is labeled based on the words meaning and expanded using synonyms and antonyms. Using corpus-based approach, instead, allows to build up a set of words that refer to the specific context of the documents. This is done making use of a wide context-specific corpus to identify new terms associated with the same labels of those of the initial list.

A well known publicly available list of opinion words is *Psychological Harvard IV-4 Dictionary*, which is part of the General Inquirer [69] software for text analysis and classify words in positive or negative. However, Loughran and McDonald [41] note that the poor performance of positive words is probably due to their frequent negation and negative word list should be preferred.

Some authors have developed finance-specific dictionaries that are widely used for research purposes and *Loughran and McDonald* [41] are among them. They analyze SEC's 10-K documents[8] filed between 1994 and 2008 and select, among words that occur in at least 5% of the documents, new terms to improve Harvard Dictionary.

Henry [27] proposes another interesting financial dictionary using words taken from corpus of earnings press releases. To account for the context of each word, she assesses its directional meaning based on the relationship with close terms in the same sentence. In particular, she starts with an initial list of words and examines the three words preceding and following each word in the documents. She evaluates if the word is positive and negative on the basis of its relationship with each of these close terms. Finally, only the words that are labeled as positive (or negative) at least in 80% of the occurrences are included in *Henry's dictionary*.

1.2.3 Supervised Learning approach

Supervised Machine Learning is a technique that is used to learn the function that maps a series of inputs to some outputs. It is based on a labeled data set, where the inputs are the features of that data set and the outputs are the corresponding labels. The supervised Machine Learning algorithm is able to learn from the examples provided by the labeled data set and then apply the same function to unlabeled dataset in order to define the output.

Among the Machine Learning methods, Medhat, Hassan and Korashy [49] present some supervised learning techniques for text classification, such as *Naive Bayes classifier* (NB), which computes the probability that a document belongs to a given class based on the distribution of the words in the document and assumes the independence of all the features. They also mention *Bayesian Network classifier* (BN), which is hardly used because it assumes all features to be fully dependent, instead, and thus requires a complete joint probability distribution to be specified. *Maximum Entropy classifier* is another classification method and, as

[8]10-K document is a form that publicly traded companies submit annually and is required by the U.S. Securities and Exchange Commission (SEC). It summarizes company's financial performance and risk factors and it is necessary to allow investors to make informed buy and sell decisions.

explained by Nigam, Lafferty and McCallum [56], it can be used when a joint probability distribution is unavailable. According to these authors, uniform distribution should be preferred in these cases and updated under the constraints coming from the training data. After selecting some relevant features from the text, they compute their expected value over the training data and set them as constraints on the conditional distribution. They also give an example saying that, if a document has 40% probability of belonging to a given class when it contains the word "professor", uniform probability should be associated with the other classes for the remaining probability: 20% for each of other three classes, for instance. In case the document does not contain the word "professor", for which an expected probability of 40% has been computed, a probability distribution is totally missing and thus the document should be given an equal probability of 25% to belong to each of the four categories.

Medhat, Hassan and Korashy [49] introduce also some *linear classifiers*, such as *Support Vector Machines* (SVMs) and *Artificial Neural Networks* (ANNs). The former separates data into classes, defining a separating hyperplane that maximizes the normal distance of any data points. Normal distance means distance is measured along a line that is orthogonal to the hyperplane. Figure 3 shows an example of classification for a bi-dimensional space, where straight line A is the has *maximum margin of separation.* Artificial Neural Networks, instead, are algorithms that

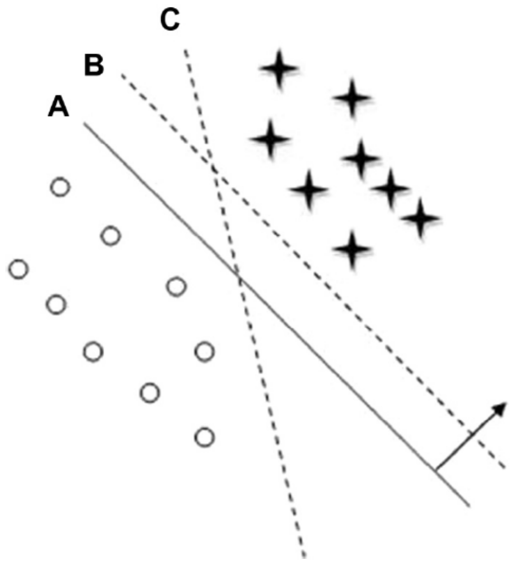


Figure 3: Support Vector Machines.
Source: Walaa Medhat, Ahmed Hassan and Hoda Korashy. "Sentiment analysis algorithms and applications: A survey". In: Ain Shams Engineering Journal 5.4 (2014), pp. 1093-1113.

mimic the functioning of human brain and are made up of units called neurons.

Each neuron receives a vector of inputs $\overline{X_i} = \{a_1, .., a_n\}$, combines them linearly using a vector of weights $A = \{x_1, .., x_n\}$ and transforms these combinations to produce an output $p_i = A \cdot \overline{X_i}$. In the case of text binary classification, the sign of p_i represents the label, positive or negative, to assign to the i-th document.

Other supervised learning algorithms mentioned by Medhat, Hassan and Korashy [49] are *Decision Tree classifiers*, which follow a recursive approach to create sub-partitions of data based on the presence of one of more words, and *Rule-based classifiers*, which divide the data defining sets based on *if-then* rules. *If* some rules are satisfied by a document, *then* that document can be classified with the corresponding class label. Rules are defined during the training phase on the basis of criteria such as *support* and *confidence.* The former counts the number of times a specific rule, for instance the presence of a word in the text, is satisfied in the whole training set, while the latter represents the conditional probability of observing a given label when the rule that is associated with that label is satisfied.

Yadav et al. [73] use a supervised Machine Learning approach to classify real time news headlines. First, they automatically label all headlines based on net buying pressure[9] patterns in S&P NIFTY Index, and then use naive Bayesian classifiers and support vector machines (SVM) for text classification, finding that the best alignment window for Indian futures markets is 5 minutes.

Atkins, Niranjan and Gerding [2] prove that financial news makes a better job in predicting stock market volatility than stock returns. To reduce data dimensionality, they use *Latent Dirichlet Allocation (LDA)*, a generative technique that helps divide each document into a set of topics. Topics are sets of words and they are generated, hence the adjective generative, on the basis of the words they contain. Instead of having documents made up of a large number of words, or possibly n-grams, each document is built up as a set of topics, so that dimensionality is reduced. In particular, Atkins, Niranjan and Gerding [2] create a list of topics for each 60-minute time interval, by assigning a sparse feature vector that counts the number of times a topic appears in that interval. Then they label these feature vectors based on the binary direction of volatility changes during the following time interval. Finally, they perform text classification using Naive Bayesian classifier and they justify this choice for the fact that it has high empirical performance even under the simplistic assumption that each feature is independent of each other.

Also Souma, Vodenska and Aoyama [67] classify financial and economic news using the NLP Machine Learning approach. They train a Recurrent Neural Network (RNN) with LSTM units, which is particularly suitable to capture long term dependencies among words in a text.Before feeding the training algorithm, they perform word embedding using Global Vectors for word representation method (GloVe) to

[9]Net buying pressure is defined as "the difference between the number of buyer-initiated trades and the number of seller-initiated trades calibrated from the bid-ask quotes".

convert words to vectors. Embedding consists in assigning a vector to each word in the dictionary such that words with similar meaning are located close to each other in the vector space. Glove contains pre-trained word vectors based on Wikipedia documents. They assign labels to each news article based on stock return over the next one-minute period and obtain a classification in positive and negative news.

Wang et al. [70] work on sentiment analysis retrieved from two SeekingAlpha articles and StockTwits messages, two social media platforms, and analyze their correlation with S&P500 Index movements, finding the former has better explanatory power, even though sentiment-based investment strategies have generally poor performances. They use a dictionary-based approach for SeekingAlpha analysis, relying on Loughran and McDonald's financial sentiment dictionary [41], and supervised Machine Learning approach for StockTwits, using labels assigned directly by users. Their best model is Supported Vector machine (SVM), a widely used algorithm that detects a separating hyperplane to perform classification.

1.2.4 Unsupervised Learning approach

There are also some authors that have explored alternative solutions to dictionary-based and Supervised Machine Learning methods. While Supervised Machine Learning needs labeled data to train the network and tailor the output to the correct label in the training set, Unsupervised Learning does not require training data to be labeled. Indeed, it can learn patterns in the training set identifying relationships among variables.

Sohangir et al. [65] analyze sentiment of StockTwits using different Unsupervised Deep Learning techniques. *Deep Learning* [36] is defined as a subset of *Machine Learning* and more specifically a type of *Representation Learning* method. Machine Learning, be it Supervised or Unsupervised, requires considerable expertise to accurately pre-process data and extract the features to feed into the algorithm, and feature selection has a high impact on the final result. With Representation Learning, instead, machines can process raw data more easily. Deep Learning is a Representation Learning method that processes data following a hierarchical and multi-level approach and extracts nonlinear features that can eventually be combined for classification purposes. This means it works with multi-layer architectures made up of nonlinear units, where features are extracted with more and more detail as later layers are approached. A common example refers to image recognition, where the first layers detect simple features, such as the edges of an image, and later layers extract more complex features such as faces [65]. Sohangir et al. [65] point out that using Deep Learning for Natural Language Processing can help solve the sparsity problem caused by high-order n-grams, which are extremely useful to keep word order, but appear infrequently in the corpus. Deep learning can help learn abstract features and learn semantic relationship among words. It often uses Artificial Neural Networks architectures with multiple hidden layers and, it these cases, it is referred to as *Deep Neural Networks*.

Deep Learning can be beneficial when applied to Unsupervised Machine Learning tasks, because of its ability to find hidden relationships among variables, and the large availability of unlabeled data makes Unsupervised Learning very attractive for future research.

The interesting fact about StockTwits is that users can write short messages and directly label them as bullish or bearish, but as only 10% of messages in this social network are labeled, Sohangir et al. [65] find that Unsupervised Deep Learning models can be useful for sentiment analysis and can predict stock returns with high accuracy.

Ruan et al. [60] build a new *investor sentiment indicator (ISI)* using Sina Finance blog, a Chinese platform that collects opinions of stock analysts. They use Baidu AI platform[10] to automatically conduct sentiment analysis through pre-developed Deep Learning tools that assign a given value to each word in the text and then combine the value of all words in the document, adjusting the overall sentiment value with respect to the specific scenario. They find that their indicator performs better than other widespread indicators, and correlation with macroeconomic predictors is low.

[10]Baidu AI platform is the largest open platform for artificial intelligent analysis in China.

2 Modeling Sentiment with LSTM networks

2.1 Introduction to ANNs and LSTM networks

Artificial Neural Networks (ANNs) are systems of interconnected nodes that take inspiration from biological neural networks. In the human brain (Figure 4), electric signals are transmitted from one nerve cell, named *neuron*, to another through some connections named *synapses*. These sensory inputs enter the neuron through some fibres known as *dendrites* and are processed by the cell body, called *soma*. Then, the output flows through the *axon*, a long tube that branches into multiple dendrites, through which information reaches the *synapses* and it is passed on to other neurons [25] [43].

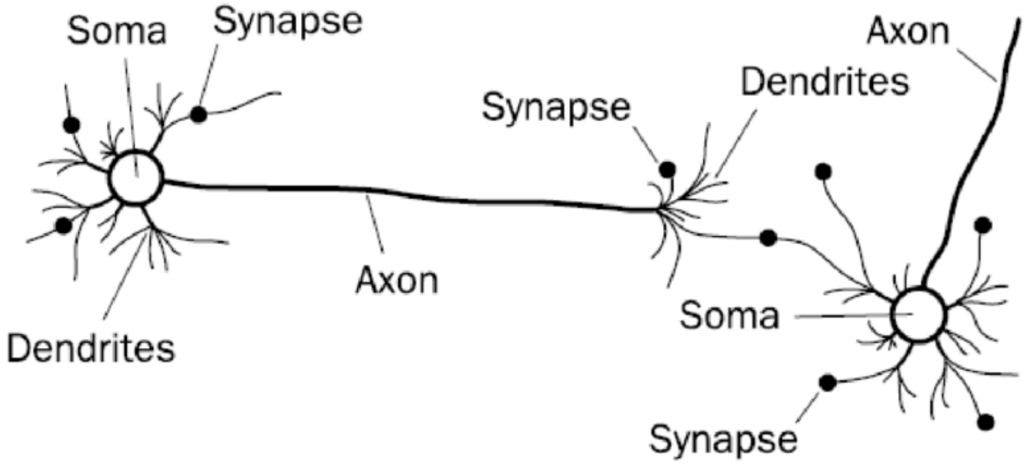
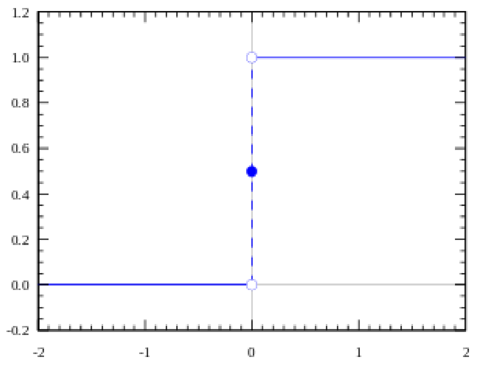


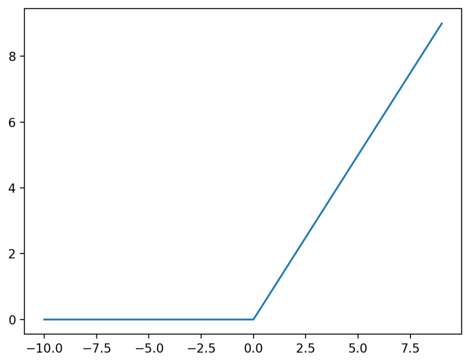
Figure 4: Biological Neural Network.
Source: F. A Makinde et al. "Prediction of crude oil viscosity using feed-forward backpropagation neural network (FFBPNN)". In: Petroleum and Coal 54 (2012), pp. 120-131. Scientific Figure on ResearchGate.

Similarly, in Artificial Neural Networks a set of inputs $X = [x_1, x_2, ..., x_n]$ is provided to each node, or unit, as presented by Lek and Park [39]. The node assigns a set of weights $W = [w_1, w_2, ..., w_n]$, one for each input, to determine the intensity of each input signal. Then, the weighted inputs are summed up and the result are fed into an *activation function* in order to generate an output. An activation function is a linear or non-linear function that is applied to the weighted sum of inputs and it decides how much the output should be transferred to the following node.

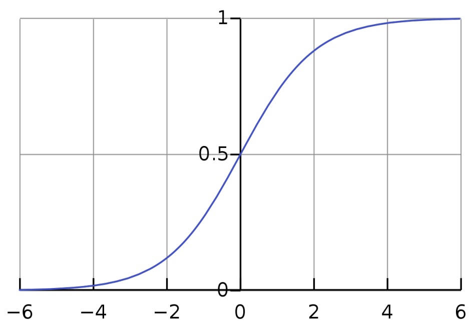
Figure 5 shows some of the most common activation functions [50], namely the *step function*, the ReLU function, the *Sigmoid function* and the *Hyperbolic Tangent function*.



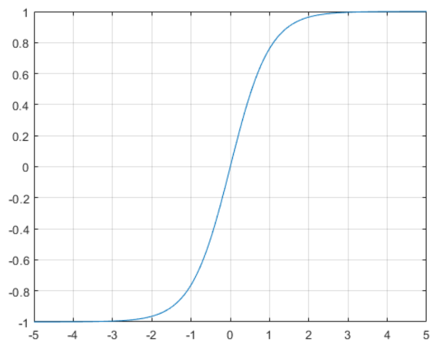
(a) *Step function.*
Source: Wikipedia. "Step function".



(b) *ReLU function.*
Source: Medium. "Why is ReLU non-linear?".



(c) *Sigmoid function.*
Source: Wikipedia. "Sigmoid function".



(d) *Hyperbolic Tangent function.*
Source: MathWorks. "tanh".

Figure 5: Types of activation functions used in Artificial Neural Networks.

Step function is a threshold-based activation function, meaning the weighted sum of the inputs is transferred to the next step (*fired*) only if its value is greater than, or equal to, a given threshold. Step function can be described by Equation

(7):

$$f(x) = \begin{cases} 1, & \text{if } x \geq \alpha \\ 0, & \text{if } otherwise \end{cases} \tag{7}$$

ReLU function, in Equation (8), is a non-linear function[11] that is linear only for positive values of x. Given its non-linearity, it allows to create artificial neural networks with multiple layers. Indeed, if all layers have linear activation functions, the final activation function is nothing more than a linear combination of the previous ones. Differently from the step function, it does not have a binary output.

$$f(x) = max\{0, x\} = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{if } otherwise \end{cases} \tag{8}$$

Sigmoid function is particularly suitable for classification tasks as it tends to be steeper in its central part and thus it pushes output towards its tails. It is non-linear and it has the great advantage of returning values between 0 and 1. It is described by Equation (9).

$$f(x) = \frac{1}{1 + e^{-x}} \tag{9}$$

Hyperbolic Tangent function is similar to sigmoid. It returns values between -1 and 1, as visible in Figure (10).

$$f(x) = tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{10}$$

As Gurney points out, ANNs can be designed following different architectures. In its simplest configuration with one single node, the *single-layer perceptron*, an input layer is followed by an output layer that produces the final result. In the opposite case, instead, a large number of nodes can be fully interconnected and form a *Fully Connected Neural Network*.

Lek and Park [39] distinguish between two main categories of ANNs, that are supervised and unsupervised learning. While the former needs labeled data to learn from the examples provided, the latter can independently find hidden relationships in the data. One common scheme of supervised learning is *Feed-Forward Network*, also called *multi-layer perceptron*, which is an evolution of the single-layer perceptron, and it is made up of at least three layers, where information is passed on from the input layer through one or more hidden layers to the final output layer. All nodes of one layer are connected to all nodes of the next layer and the previous layer, but there are no connections between nodes of the same layers or non-adjacent layers.

[11]ReLU function is not a linear function because it does not satisfy one of the two properties that define linearity, namely additivity. Additivity implies that $f(x+y) = f(x)+f(y)$. The other property is homogeneity, that is $f(\alpha x) = \alpha f(x)$ for all α.

2.1.1 Delta Rule and Backpropagation algorithm

The way connection weights are adjusted in Feed-Forward Networks is known as *Backpropagation Algorithm*. The first part of this algorithm is called *Forward Propagation* and works in the following way. First, weights are randomly determined and weighted inputs summed up and fed into the activation function. Then, the value of the activation function is passed on to the following layer and, when the last layer is reached, an output is produced. The output is compared with the desired output and the difference δ, named *error*, is used to update the initial weights so that the process can be run again and the error minimized.

In single-layer Perceptron, the weight-correction procedure [39] works multiplying the difference between target and output value by the derivative of the activation function at point a_k to obtain Equation (11).

$$\delta_k = (t_k - x_k) f'(a_k), \tag{11}$$

$(t_k - x_k)$ is the difference between target and output value of k-th output layer, and $f'(a_k)$ is the derivative of the weighted sum of inputs of k-th output layer, a_k. $f'(a_k)$ is useful to increase the correction step when the weighted sum of inputs is close to the central part of the non-linear activation function, for instance the sigmoid. The correction works as in Equation (12), where Δw_{kj} is the variation in the weights that connect unit j to unit k, δ_k represents the error of unit k as in Equation (11), x_j is the output value of the previous unit and η is the *learning rate*, which is set by the user and determines the learning speed of the network. It usually takes values between 0 and 1.

$$\Delta w_{kj} = \eta \delta_k x_j \tag{12}$$

This Equation represents a slight modification of the so called *Delta Rule*, because it introduces a non-linear activation function, instead of a linear one, to speed the correction up. The derivation of the Delta rule comes from a general algorithm called *gradient descent*, which is an iterative procedure that adapts a vector of parameters in order to minimize an objective function. Gradient descent is an iterative technique that uses the gradient of a cost function to find its local minimum. The gradient of a multivariate function with k variables, $\nabla f(x_1, x_2, ..., x_k)$ [71] [72], is a vector whose components are the partial derivatives of f at point $x = (x_1, x_2, ..., x_k)$ and it summarizes the direction and rate of fastest increase of f. Gradient descent method exploits this definition by taking small steps towards the direction of the negative gradient. The result is a sequence of vectors $(x_n)_{n=0}^{\infty}$ that moves towards the local minimum. Each element of the sequence is described by Equation (13), where x_n and x_{n+1} are the n-th and $(n+1)$-th elements of the sequence, $\nabla f(x_n)$ is the gradient of f at point x_n, and $\gamma \in R$ is the step size. Figure 6 shows how the sequence behaves on a series of level sets.

$$x_{n+1} = x_n - \gamma \nabla f(x_n), n \geq 0 \tag{13}$$

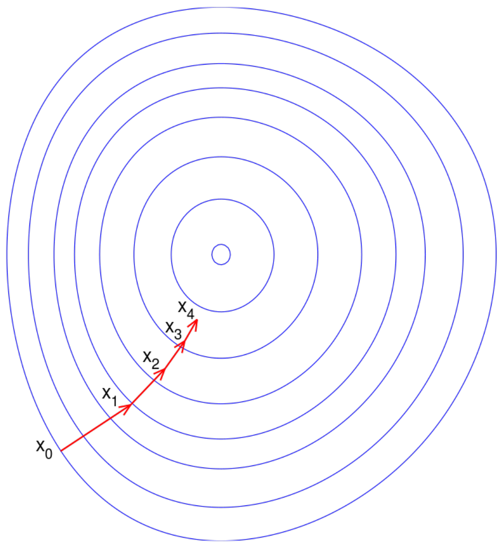


Figure 6: Gradient descent on a series of level sets.
Source: Wikipedia. "Illustration of gradient descent on a series of level sets".

Delta rule was introduced for a single layer network, but it can be generalized and applied to multiple layers networks, using the Backpropagation Algorithm. The first part of this algorithm has already been presented. The second part deals with the actual *Backpropagation*. Backpropagation [74] is the method used to adjust the weights given the loss function produced in the Forward propagation. Weights are modified computing the gradient of the parameters starting from the output layer and going backward to the input layer, following the chain rule[12]. Nielsen [55] explains how the Backpropagation phase works and start with two main assumptions. The first one regards the functional form of the cost function, which should be expressed as an average $C = \frac{1}{n}\sum_x C_x$ of the cost functions computed over different training sets x. The second assumption is that the cost function should be described as a function of the final output of the ANN, a_j^L. Both conditions are satisfied by the quadratic cost function in Equation (14).

$$C(a_j^L) = \frac{1}{2}\sum_j (y_j - a_j^L)^2, \tag{14}$$

where a_j^L is the result of the activation function of the j-th neuron of the output layer L and y_j is the target value of the j-th neuron of the output layer.

[12]The chain rule helps compute the derivatives of composite functions. Given two differentiable functions f and g, $\frac{d}{dx}f(g(x)) = f'(g(x)) \cdot g'(x)$.

In order to adjust weights following the backpropagation technique, one needs to compute the partial derivatives of the cost function with respect to the weights and the bias $\frac{\partial C}{\partial w_{jk}^l}$ and $\frac{\partial C}{\partial b_j^l}$. To familiarize with the notation used by Nielsen, w_{jk}^l represents the weight (Figure 7) that connects the k-th neuron of layer $l-1$ to j-th neuron of layer l, while b_j^l is the bias of the j-th neuron of layer l. The bias is a

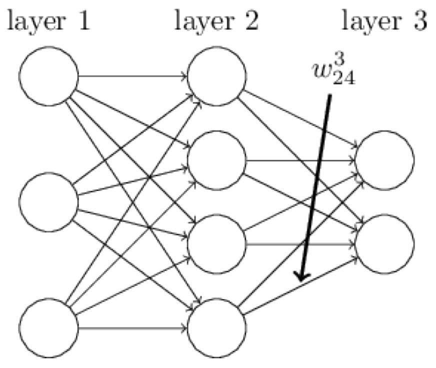


Figure 7: Notation used by Nielsen.
Source: Michael Nielsen. "How the backpropagation algorithm works". In: Neural Networks and Deep Learning. 2015. Chap. 2, pp. 39-58.

term that is added to the weighted sum of inputs of each j-th neuron of layer l, as shown in Equation (15)

$$z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l, \tag{15}$$

where the inputs a_k^{l-1} correspond the output of the previous layer. This weighted sum is then fed into a sigmoid activation function, leading to Equation (16), which represents the output of the j-th neuron of layer l.

$$a_j^l = \sigma(z_j^l) \tag{16}$$

To compute $\frac{\partial C}{\partial w_{jk}^l}$ and $\frac{\partial C}{\partial b_j^l}$, Nielsen introduces an intermediate variable δ_j^l, representing the *error* of the j-th layer. As the actual output and the target output of hidden layers are unknown, Nielsen finds that Equation (17) can be a useful proxy.

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l} \tag{17}$$

The reason for applying this proxy comes from the fact that, in order to adjust weights, one should focus on choosing the sum of weights z_j^l that minimizes the cost function C. This is done by taking small steps towards the opposite direction of $\frac{\partial C}{\partial z_j^l}$ and therefore this measure can approximate the error for layer j-th.

The error $\delta_j^l \equiv \frac{\partial C}{\partial z_j^l}$ can be computed through the chain rule, as in Equation (18). Indeed, the cost function C, expressed by Equation (14), is a function of a_j, which is itself a function of z_j^l, as shown by (16).

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l} = \frac{\partial C}{\partial a_j^l}\frac{\partial a_j^l}{\partial z_j^l} = \frac{\partial C}{\partial a_j^l}\sigma'(z_j^l) \tag{18}$$

The last term is straightforwardly computed plugging $\frac{\partial a_j^l}{\partial z_j^l} = \sigma'(z_j^l)$ into the equation. This means the error δ_j^L of the output layer L can be computed from this equation, because the final output a_j^L and the functional form of the activation function are known. To actually determine the value of δ_j^l for all layers, Backpropagation method is used. Starting from the output layer it is possible to go backward through the network and retrieve the errors of the l-th layer, given the error of the $(l+1)$-th layer. Equation (19) represents δ_j^l in terms of δ_j^{l+1} and is computed, once again, using the chain rule. The last line of this equation substitutes $\frac{\partial C}{\partial z_k^{l+1}}$ with δ_k^{l+1}, as in Equation (17).

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l} = \frac{\partial C}{\partial z_k^{l+1}}\frac{\partial z_k^{l+1}}{\partial z_j^l} = \delta_k^{l+1}\frac{\partial z_k^{l+1}}{\partial z_j^l} \tag{19}$$

Chain rule can be applied because C is a function of z_k^{l+1}, which is itself a function of z_j^l. Indeed, plugging Equation (16) into the cost function C, expressed by Equation (14), and fixing $L = l+1$, one obtains Equation (20). It is easy to note that $C(a_k^{l+1})$ is a function of z_k^{l+1} through a_k^{l+1}.

$$C(a_j^{l+1}) = \frac{1}{2}\sum_j (y_j - \sigma(z_j^{l+1}))^2 \tag{20}$$

Plugging Equation (15) into Equation (20), it is straightforward that z_j^{l+1} is also function of outputs from layer l, a_k^l, and consequently z_k^l.

Solving $\frac{\partial z_k^{l+1}}{\partial z_j^l}$ in Equation (19), using Equations (15) and (16), leads to Equation (21).

$$\delta_j^l = \delta_k^{l+1}\frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_k \delta_k^{l+1} w_{kj}^{l+1}\sigma'(z_j^l) \tag{21}$$

Finally, using Equations (18) and (21), it is possible to compute $\frac{\partial C}{\partial w_{jk}^l}$ and $\frac{\partial C}{\partial b_j^l}$. The partial derivative of the cost function with respect to the bias is simply equal to δ_j^l for the j-th neuron in the l-th layer, as shown in Equation (22).

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{22}$$

The partial derivative of the cost function with respect to the weights can be expressed in terms of δ_j^l, as Equation (23) shows.

$$\frac{\partial C}{\partial w_{jk}^l} = a_l^{l-1}\delta_j^l \tag{23}$$

2.1.2 Recurrent Neural Networks

A *Recurrent Neural Network* (RNN) [63] is a type of Artificial Neural Network (ANN) that processes sequences of inputs to obtain one or more final outputs and it is widely used in speech recognition, image captioning[13] and next word prediction. RNN architectures can be *one-to-many*, with one input and many outputs, *many-to-one*, with many inputs and one output, or *many-to-many*, with a sequence of many inputs and a sequence of many outputs. In this last case, number of output and inputs can be the same or different. Unlike ANNs, RNNs have an internal memory that keeps past observations so that the output depends not only on the current input, but also on past information. In addition, they work well with sequences of variable length.

Figure 8 shows an common example of many-to-many RNN, with number of inputs equal to number of outputs [1] [66], where a sequence of vectors x_t is fed into the machine and produces a sequence of outputs $\hat{y}_t$. For each time step t, x_t,

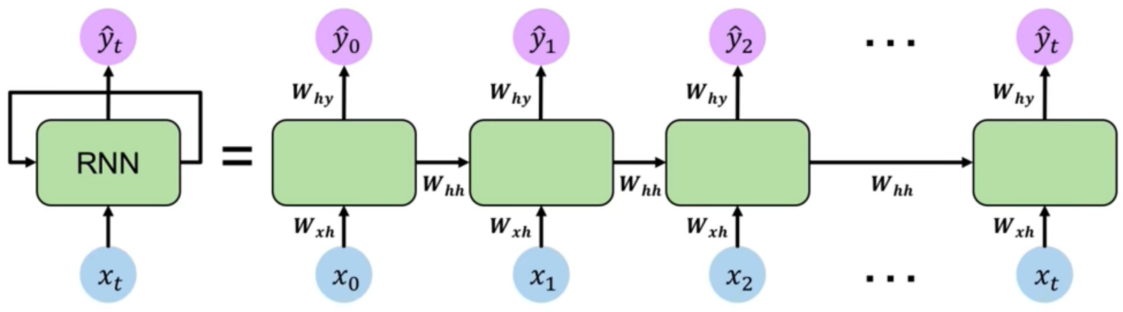


Figure 8: Recurrent Neural Network.
Source: Massachusetts Institute of Technology. "MIT 6.S191 Introduction to Deep Learning. Deep Sequence Modeling with Recurrent Neural Networks".

which represents the input vector, is pre-multiplied by a weight matrix W_{xh}, as in traditional ANNs. In RNNs, however, past activations h_{t-1}, named hidden states, are used as additional inputs for the network, which assigns them a weight matrix W_{hh}. The activation for each time step is represented by Equation (24) and consists

[13]Image captioning consists in generating a textual description from an image.

in applying a function g_1 to the weighted sum of inputs and past activations with the addition of the constant vector b_h, named *bias*.

$$h_t = g_1(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \tag{24}$$

The hidden state h_t is both used to compute the output $\hat{y}_t$ and passed on to the next time step. For each time step t, output $\hat{y}_t$ is computed similarly to activation for a common ANN. The weighted sum of current activation h_t is computed using weight matrix and a constant term is b_y added to the calculation. Then a function g_2 is used to transform this weighted sum, as in Equation (25).

$$\hat{y}_t = g_2(W_{hy}h_t + b_y) \tag{25}$$

In Supervised Learning, the true sequence of outputs y_t of the training set is known and a Loss function $L(y_t, \hat{y}_t)$ is computed for each time step t. The final total Loss function $L(y, \hat{y})$ is then obtained summing up these intermediate loss functions, as in Equation 26.

$$L(y, \hat{y}) = \sum_{t=1}^{T_y} L(y_t, \hat{y}_t) \tag{26}$$

RNNs are trained using a technique known as *Backpropagation Through Time* (BPTT), which is an application of the Backpropagation Algorithm presented in the previous paragraph. Like traditional Backpropagation, BBTT uses the gradient of the Loss function to minimize $L(y, \hat{y})$ and find the optimal weights. To do so, it is necessary to retrieve the partial derivative of the Loss function with respect to each weight in the network. In traditional Backpropagation, partial derivatives are computed starting from the output layer and going backward through the hidden layers to the input layer. In Backpropagation Through Time the network is unfolded[14] as in Figure 8 and, therefore, an additional variable, time t, must be considered. For each time instant t, the partial derivatives of the total Loss function $L(y, \hat{y})$ with respect to the weights and biases, W_{hy}, W_{xh}, b_y and b_h, are computed backward through the layers using the chain rule, as in common Backpropagation. But in this case, Backpropagation also acts through time, meaning gradient is back-propagated across time steps, as shown by Figure 9. This means the chain rule computes the partial derivative of $L(y, \hat{y})$ with respect to all weights at each time step t passing through all layers of the individual network and then going backward in time.

In principle, RNNs would be suitable to learn long-term dependencies, but in practice they suffer from *vanishing and exploding gradient* problems [21] [66]. Recurrent Neural Networks can have arbitrary length and applying the chain rule

[14]An unfolded representation of Recurrent Neural Network is represented in Figure 8, where the same RNN network is repeated many times and the activations of each time step are fed into the next time step.

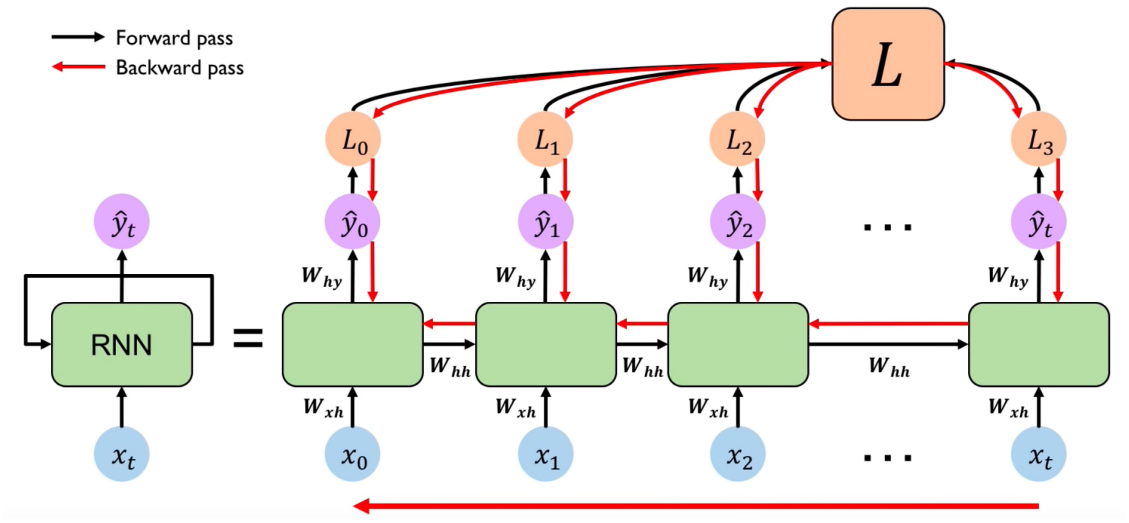


Figure 9: Backpropagation Through Time.
Source: Massachusetts Institute of Technology. "MIT 6.S191 Introduction to Deep Learning. Deep Sequence Modeling with Recurrent Neural Networks".

backward in time implies computing a long series of weight matrix multiplications, as well as multiplication of derivatives of the activation functions. When many multiplication terms are greater than one, their product can explode. When terms are close to zero, instead, their product can vanish. This means the network might meet some difficulties in computing the gradient for early time steps, not being able to learn long-term dependencies. For instance, when adopting Sigmoid or Hyperbolic Tangent activation functions (Figure 5), the derivative of the activation, which is included in the formula of the chain rule used in the Backpropagation phase, as shown by Equation (21), tends to vanish if the input of the function is very high or very low. This is because derivative of both functions is zero at the tails (Figure 10).

Some solutions may include changing the functional form of the activation function, initializing the weight matrix with reasonable values, or modifying the structure of the network. ReLU function (Figure 5), for instance, has constant derivative for values higher than zero, but the problem remains for values lower than zero. Gradient Clipping, instead, is a technique that involves fixing the maximum value of the gradient of the loss function to an arbitrary value and can be adopted for exploding gradient problems [1]. However, the best results have been obtained using particular types of Recurrent Neural Network architectures that have been designed to deal with both exploding and vanishing problems and the most renowned one is *Long Short Term Memory networks* (LSTMs).

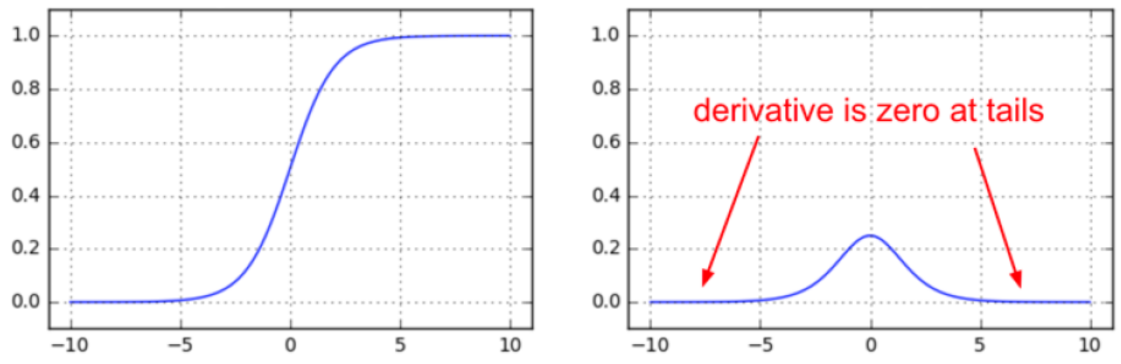


Figure 10: Derivative of Sigmoid function.
Source: Massachusetts Institute of Technology. "MIT 6.S094: Deep Learning for Self Driving Cars. Recurrent Neural Networks for Steering Through Time".

2.1.3 LSTM networks

Long Short Term Memory networks (LSTMs) [57] are a particular type of Recurrent Neural Network, specifically designed to solve vanishing and exploding gradient problems. Therefore, it achieves high performance in learning long-term dependencies.

Figure 11 shows an unfolded representation of a LSTM network, where each module represents an individual LSTM module at time step t. A standard Recurrent Neural Network has one hidden layer.LSTMs are designed to have four hidden layers that interact with each other, instead, represented by the yellow boxes in Figure 11.

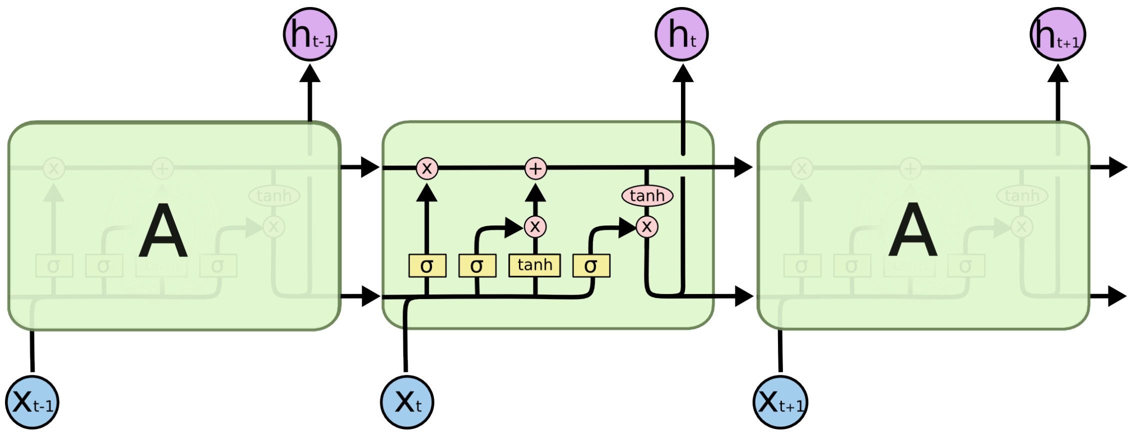


Figure 11: LSTM network.
Source: Christopher Olah's blog. "Understanding LSTM Networks".

To keep track of relevant long-term dependencies, LSTM model contains a *cell* *state* C_t, which memorizes the information about the training set that is learned after the module processes the available information at time t. The cell state C_t substitutes the hidden state h_t of standard Recurrent Neural Networks. In LSTM networks, the notation h_t is used to define the output, instead. The current cell state is computed starting from the previous cell state C_{t-1} and updating it on the basis of the new input at time t. Information contained in cell state C_{t-1} flows through the so called *conveyor belt*, as in Figure 12. Before passing information on to next step, the conveyor belt undergoes only two pointwise operations, which introduce some corrections to the cell state. Therefore, information can flow quite easily.

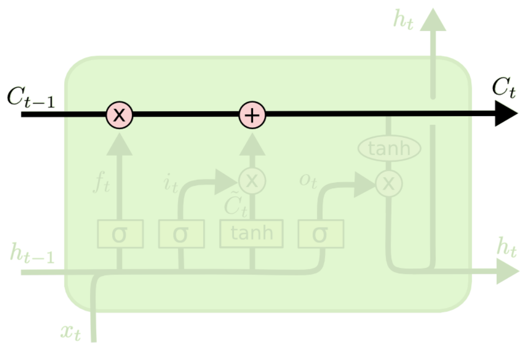


Figure 12: Conveyor belt in a LSTM network.
Source: Christopher Olah's blog. "Understanding LSTM Networks".

A system of *gates* determines what part of information is added and what part is removed. Gates are made up of a sigmoid layer, which forces the input of the gate to be between zero and one, and a pointwise operation that updates the cell state. Each module receives an input x_t which, together with the output of the previous module h_{t-1}, passes through a *forget gate* (Figure 11 Part (a)). The forget gate f_t, represented by Equation (27), applies a Sigmoid function to the weighted sum of current input and previous output, so that all values of the cell state are assigned a number between zero and one. In this way, the gate assesses how much of the information contained in the cell state should be forgotten.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{27}$$

The second and third layers decide how to update the cell state with new information (Figure 11 Part (b)). In particular, the *input gate* i_t, which behaves similarly to the forget gate, decides which elements of the cell state will be updated. Another layer, $\tilde{C}_t$, applies a Hyperbolic Tangent function to the same variables used by f_t and i_t and it determines the new candidate values for C_t. Both layers are represented by Equations (28).

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \end{aligned} \tag{28}$$

Equations (28) and Equation (27) are combined to compute the actual cell state C_t and Equation (29) describes how update is executed.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{29}$$

In the first term, the forget gate f_t is multiplied by the previous cell state C_{t-1}, so that unnecessary information is erased from the cell state. In the second term, instead, the input gate i_t, which decides what elements to update and how much to update them, is multiplied by the candidate values $\tilde{C}_t$. Finally, summing up the two terms, the cell state is updated, dropping useless information and including new significant information. Figure 11 Part (c) graphically shows how Equation (29) update the cell state, starting from the outputs produced by the first three layers.

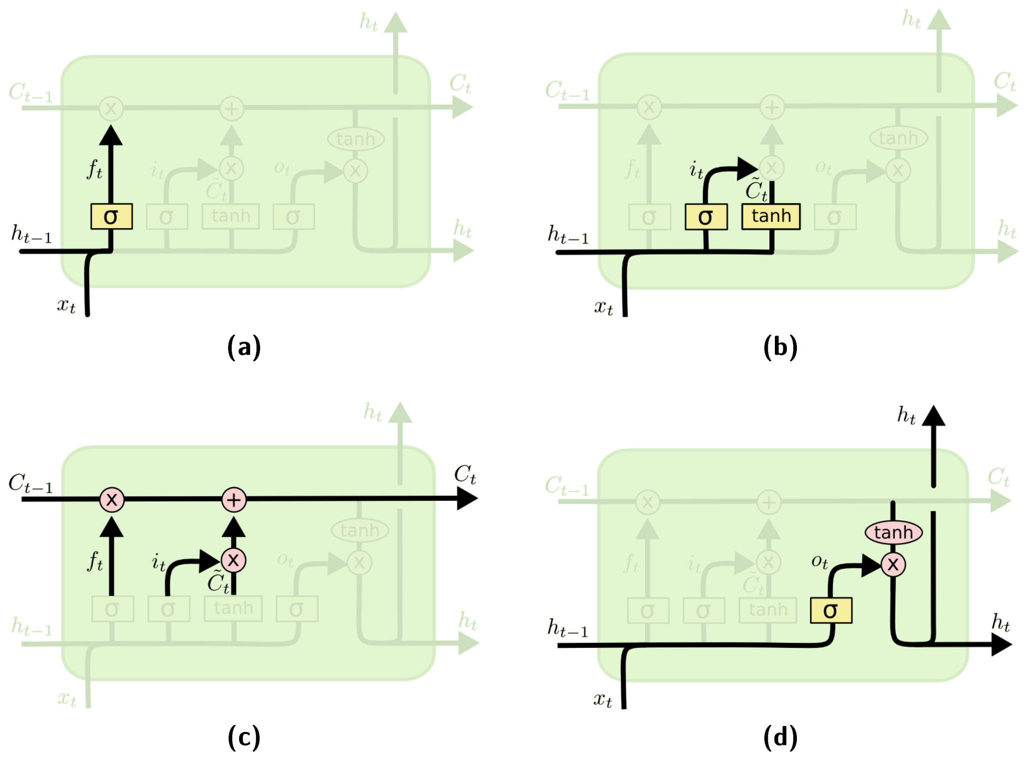


Figure 13: Broken down LSTM Network.
Source: Christopher Olah's blog. "Understanding LSTM Networks".

The last layer (Figure 11 Part (d)) is another gate that applies a sigmoid function to the weighted sum of previous output h_{t-1} and current input x_t and it is called *output gate*. The output gate decides what part of the updated cell state to output. Before multiplying the result of the output gate by the cell state, the values of the cell state are forced to be between minus one and plus one, using a Hyperbolic

Tangent function. The result of the output gate o_t and the final output h_t are described by Equations (30).

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * tanh(C_t) \end{aligned} \tag{30}$$

The real advantage of this model is that it completely solves vanishing and exploding gradient problem [66]. Indeed, backpropagation only acts among cell states C_t and h_t is not involved in this. Equation (29) clearly shows that C_t only involves pointwise operations and, thus, matrix multiplications are not necessary when applying the chain rule.

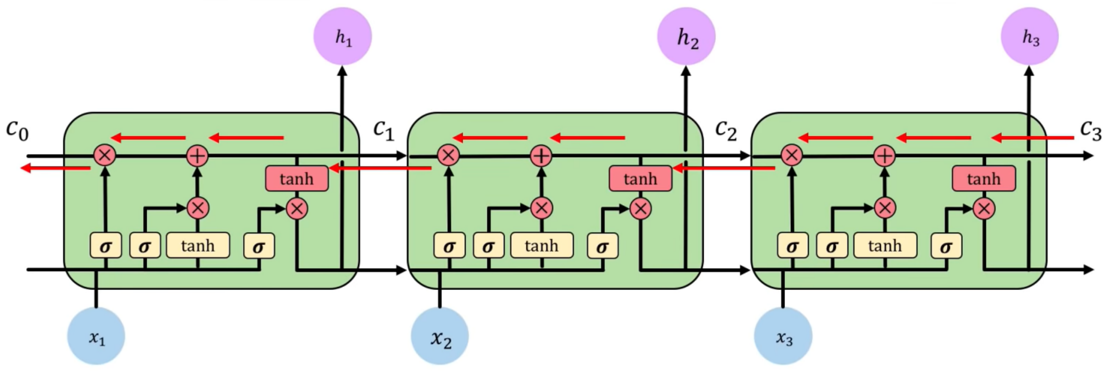


Figure 14: Backpropagation Through Time in a LSTM network.
Source: Massachusetts Institute of Technology. MIT 6.S191 Introduction to Deep Learning. Deep Sequence Modeling with Recurrent Neural Networks".

2.2 The model

In this section, I present the main findings of the model I developed. The aim of this work is to define a reliable proxy for market sentiment using financial news sources and assess if it is correlated with S&P 500 Index returns or volatility.

For this aim, publicly available financial news articles related to S&P 500 Index were scraped from Reuters.com website. Web scraping was done using *Rvest* and *Rselenium*, two open source packages available for the software environment *R*. As the web page is a dynamic page, it was necessary to use a dynamic approach to retrieve the html code. Using Rselenium, I simulated browser activity to load web pages that contained older news articles and with Rvest I extracted the html code that included article headlines and links to these articles. Then, using those links, I extracted the remaining information.

Scraping techniques allowed me to download 20,799 news articles, from March 20 to November 7, 2019. The output was an excel file, which did not include any html tags, where each article was cataloged by headline, body, date and time.

Article Headlines and bodies were concatenated in order to have a unique text source for each article.

As news articles scraped from Reuters are not labeled, I introduced some assumptions to automatically label them, following two main approaches. The first approach defines labels based on S&P 500 Index returns, while the second one on S&P 500 Implied Volatility Index (VIX) log variations.

First, I downloaded intraday S&P 500 Index prices from Bloomberg Terminal, choosing 20-minute time intervals, as suggested by Gidofalvi and Elkan [23]. Then, I computed log returns using the close price of each time interval t as in Equation (31), except for the first 20-minute interval of each day, where log returns were computed using the open price and close price of that time window.

$$R_t = log\left(\frac{Close_t}{Close_{t-1}}\right) \tag{31}$$

All news articles were classified into three categories on the basis of the time interval they belong to. Articles published during time intervals that registered a positive return, higher than the 65th percentile, were labeled as *positive*. Those included in intervals that scored negative, meaning returns were lower than the 35th percentile, were classified as *negative*. With returns between the 35th and 65th percentile, articles were labeled as *neutral*. The choice of using these two percentiles derives from the attempt of capturing only relevant price variations, which are not due to normal price fluctuations. In addition, these percentile levels allows positive, neutral and negative news to be almost relatively equally labeled on the data set.

Of course, this labeling approach, based on fixed time intervals, has the limit that articles published at the beginning of each time window have enough time to impact stock index returns, meaning they belong to that specific time interval and they are labeled on the basis of price movements during that time interval, while articles published at later times within the time window are labeled with price movements of that time window even if they don't impact them. I also tried to apply a lagged labeling technique, meaning each article was labeled following index returns over the next time window, but no significant improvement was registered. An alternative solution might be observing index returns over the exact 20-minute period following each news.

During this matching procedure, all articles published before 9:30 AM and after 4:00 PM were removed, as it was not possible to match them with any label. In addition, Bloomberg Terminal allowed me to download only data from May 14 to December 17, 2019. Therefore, news articles available for sentiment analysis were 3506 (2628 when considering lagged labeling).

The same procedure was followed for VIX Index. Intraday VIX data were downloaded from Bloomberg Terminal, and I performed automatic labeling considering VIX Index log variations, instead of index returns, because VIX can be considered as proxy for investor sentiment [8]. In this case, negative log variations in the VIX index were associated with more optimistic sentiment and positive variations to more pessimism among investors. The reason is that volatility is asymmetric, meaning it is higher in downtrends. It was possible to download VIX data between June 5 and December 17, 2019, therefore news articles available for sentiment analysis at the end of this matching procedure were 3090 (2719 when considering lagged labeling).

After running the first codes, I tried to modify some parameters looking for improvements. Indeed, I tried with both 20-minute and 10-minute time intervals, as Gidofalvi and Elkan [23] empirically prove that increasing time interval size, from the optimal 20-minute one, worsen stock return prediction. Furthermore, in an effort to obtain better results, I changed the percentiles used to define positive, negative and neutral news. Besides 35th and 65th percentiles, I also tried with 45th and 55th.

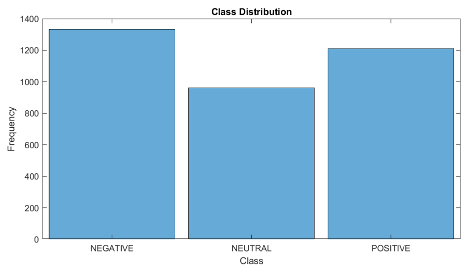
2.2.1 Data, preprocessing and labeling

Text data were pre-processed using Matlab, which gives access to many toolboxes for text analysis and deep learning.

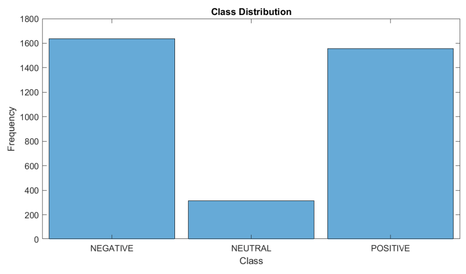
The excel file containing the corpus was imported into Matlab environment. Figure 15 presents the histograms of the class distribution of news articles labeled using S&P 500 Index 20-minute log returns. In Part (a) the percentiles used to define positive and negative news are the 35th and 65th percentiles, and out of 3506 articles, 1333 were classified as negative, 962 as neutral and 1211 as positive. When using the 45th and 55th percentiles, instead, out of 3506 articles, 1636 were classified as negative, 315 as neutral and 1555 as positive. Thus, in Figure 15 Part (b), the number of news classified as neutral decreases.

In Figure 16, similar histograms are presented, where articles are labeled using VIX Index 20-minute log variations. In Part (a), the 35th and 65th percentiles were used and out of 3090 articles, 1099 were classified as negative, 342 as neutral and 1649 as positive. When using the 45th and 55th percentiles, as in Part (b), 1364 articles were labeled as negative, 293 as neutral and 1433 as positive. Using 35th and 65th percentiles, leads to a more uniform classification, at least in the case of log returns, but results are unsatisfactory. However, the fact that the number of negative and positive labels is almost identical is a good starting point for training the network.

Before training the network, the first step is dividing data into three sets, namely Training, Validation and Testing sets. The Training set is used to find the optimal

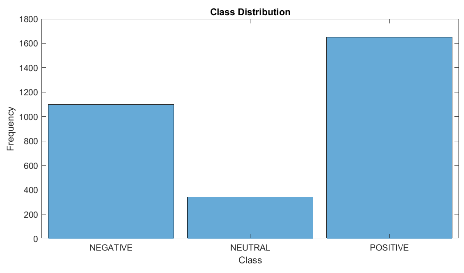


(a) *20-minute log returns with 35th-65th percentiles.*

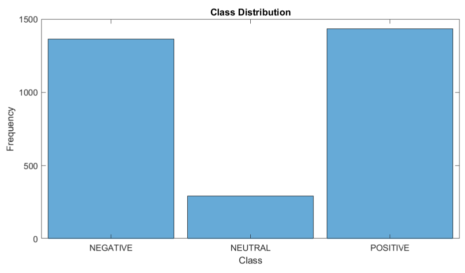


(b) *20-minute log returns with 45th-55th percentiles.*

Figure 15: Financial news labels based on S&P 500 Index 20-minute log returns. *Source: Matlab.*



(a) *20-minute VIX log variations with 35th-65th percentiles.*



(b) *20-minute VIX log variations with 45th-55th percentiles.*

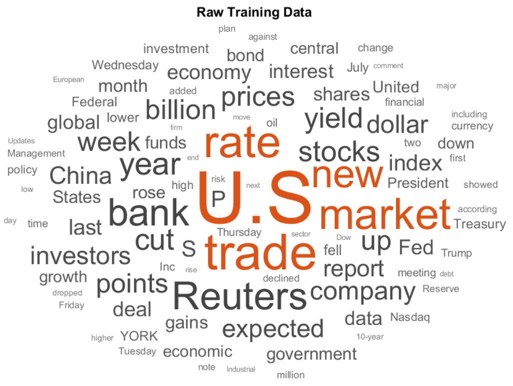
Figure 16: Financial news labels based on VIX Index 20-minute log variations. *Source: Matlab.*

weights through Backpropagation. Many algorithms, like the one I explain in Section 2.2.3, divide the Training set into subsets of observations named *mini-batches*. The network computes the total loss function at each mini-batch and updates the parameters, which can be used by the following mini-batch. The Validation set, which is itself divided into the same number of mini-batches, is used to test the model with the parameters optimized by the network at each mini-batch. The Testing set, instead, is used to test the fully trained network, when the learning phase is completed.

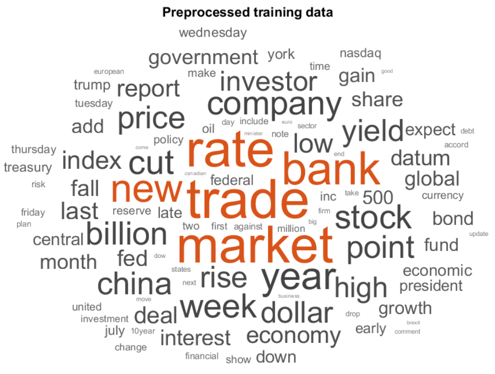
I used Holdout cross validation technique to divide data into Training, Validation and Testing sets. After fixing the percentage of data to allocate to each of these subsets, this technique allows to randomly select observations from the data set. I used 70% of the articles for Training and 15% each for Testing and Validation set.

Figure 17 Part (a) shows the words that appear most frequently in the Training set, before the preprocessing phase. Words like *Reuters* appear very often in the

text but are unlikely to have strong semantic relevance for the article. Fortunately, punctuation is not included among the most frequent features even before preprocessing the data. Preprocessing is an important phase and can heavily influence the



(a) *Raw word cloud.*



(b) *Processed word cloud.*

Figure 17: Word cloud representation of raw Training Data and Preprocessed Training Data.
Source: Matlab.

accuracy of text classification. First, documents were tokenized and punctuation and stop words were erased. Then, all words were converted to lower case, in order to avoid duplication.

Another useful technique to reduce dimensionality is lemmatization, which consists in keeping only roots of each words, as in stemming. The difference is that stemming uses general rules to cut off the final part of a word, for instance it cuts suffixes like "ing" or "ed", while lemmatization is a more complex method that is able to turn each word into its dictionary form [62]. Words like "are" and "is" are transformed into "be", for instance. Matlab provides pre-trained lemmatization tools based on widely available English dictionaries. To apply lemmatization, *part of speech* (POS) information is needed. This means words are categorized in nouns, verbs, adjectives, adverbs and so on.

In addition, words with less than two characters or more than 14 characters were removed. Lastly, I removed "reuters" from the corpus, as it unlikely to be informative.

After preprocessing the Training set and doing the same with validation and testing sets, the word cloud representation in Figure 17 Part (b) shows that some words do not appear anymore, even though a lot of them are still present.

Tokens went through encoding, which transformed them into numerical indexes, expressed by a sparse vector with all zero values but the one corresponding to the specific word in the vocabulary. In this way, tokens can be recognized by the machine.

Another important aspect to consider is document size. Each document is an article that contains an arbitrary number of words. However, best results can be achieved when the Training set contains documents with similar length, because infrequent long articles might bias the learning phase, and, therefore, their length should be reduced. Figure 18 represents length of documents that belong to the Training set using a histogram. Most documents have length that does not exceed 360, so this value can be used as an appropriate threshold. All articles that exceed this limit were truncated, while shorter articles were padded, meaning they were filled in with zero vectors.

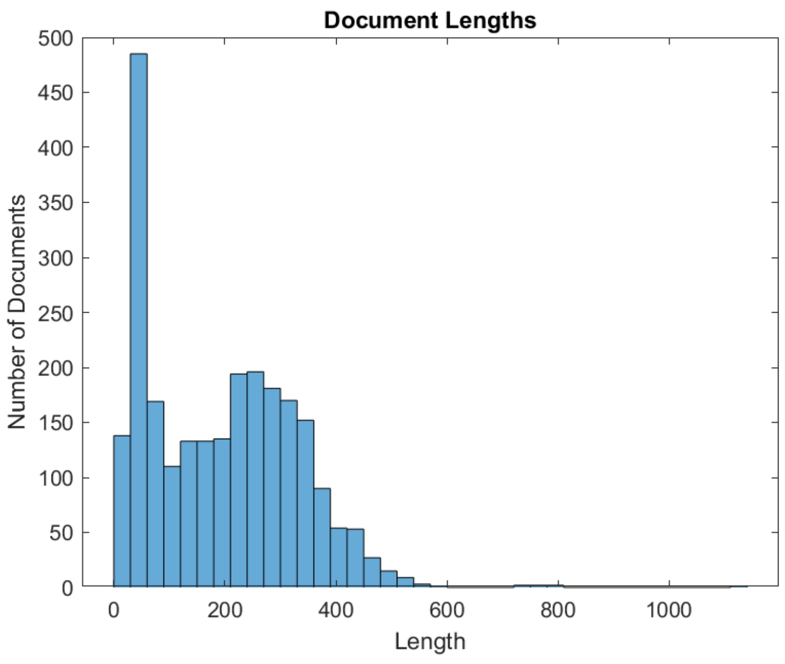


Figure 18: Length of news articles included in the Training set.
Source: Matlab

2.2.2 The LSTM architecture

In this section, I describe the Artificial Neural Network architecture used to analyze sentiment and divide financial news in three classes: positive, negative and neutral.

Before training the network, a common practice is performing word embedding. Embedding allows for dimensionality reduction and it can be useful to capture the semantic of words, mapping words into real vectors, as explained in Chapter 1. Among the different alternatives, I decided to include an embedding layer in the Artificial Neural Networks used to train the model. A different solution might be using pre-trained word embedding.

The first step of the ANN is feeding the network with sequences of words. Therefore, the first layer is a one-dimensional *Input layer* that takes each individual word of the sequence and passes it on to an *Embedding layer*, as shown by Figure 19. The Embedding layer maps each word to a d-dimensional dense vector and adapts it to the corpus during the learning phase. The weighting matrix computed by the network for this layer is a $d \times k$ matrix, where d represents the embedding dimension and k the dictionary size, meaning the number of words in the dictionary.

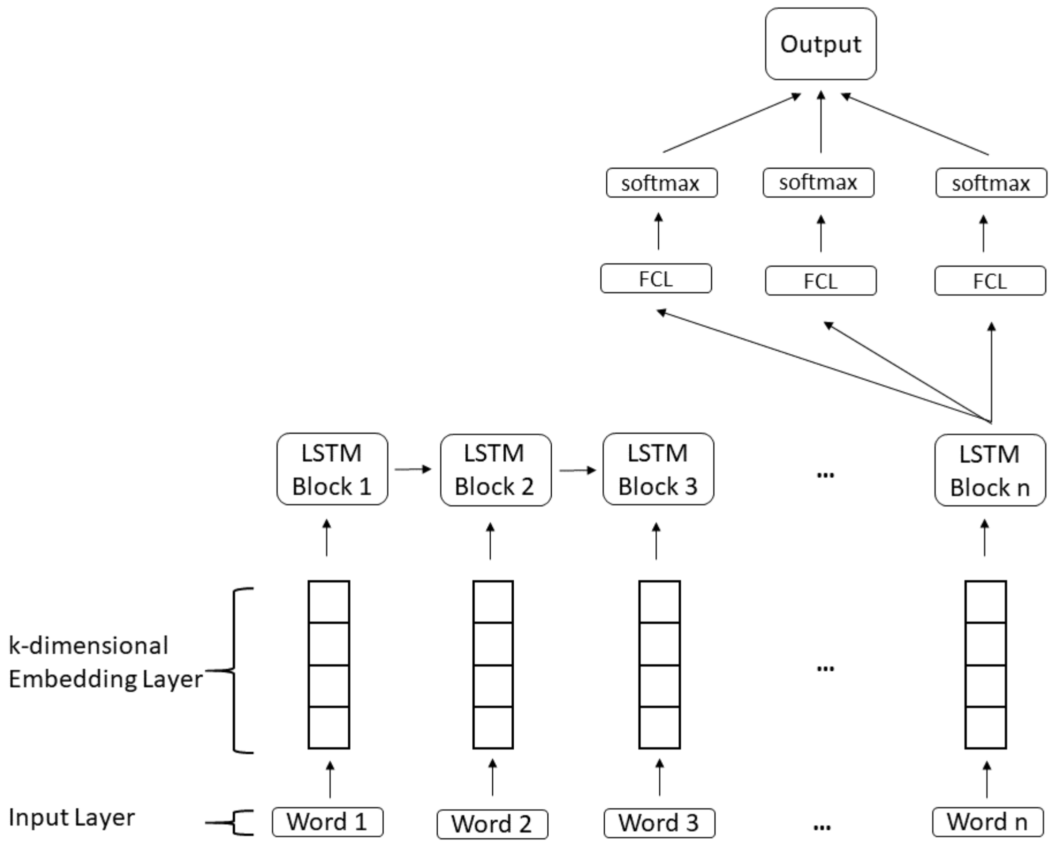


Figure 19: ANN Architecture with LSTM layer.

After the Embedding layer, vectors are transferred to the *Long Short Term Memory layer*, which is made up of an arbitrary number of LSTM blocks. Each vector is associated with a LSTM block, which includes multiple LSTM units, as shown in Figure 20. Units are essentially represented by the LSTM modules in Figure 11. According to the Matlab documentation [46], the number of hidden units determines how much information is remembered between time steps. The number of hidden units should be fixed to an appropriate integer value, in order not to overfit the training data.

For each LSTM block, the input weights to compute are defined by a $d \times 4u$ matrix, where d is again the embedding dimension and $4u$ the number of units of each LSTM block u multiplied by four, recalling that a LSTM unit has four layers (Forget Gate, Input Gate, Candidate Cell and Output Gate). In addition, each LSTM block has a $4u \times u$ recurrent weight matrix that must be computed by the network at each time step. Finally, the layer learns also a $4u$-dimensional vector of biases.

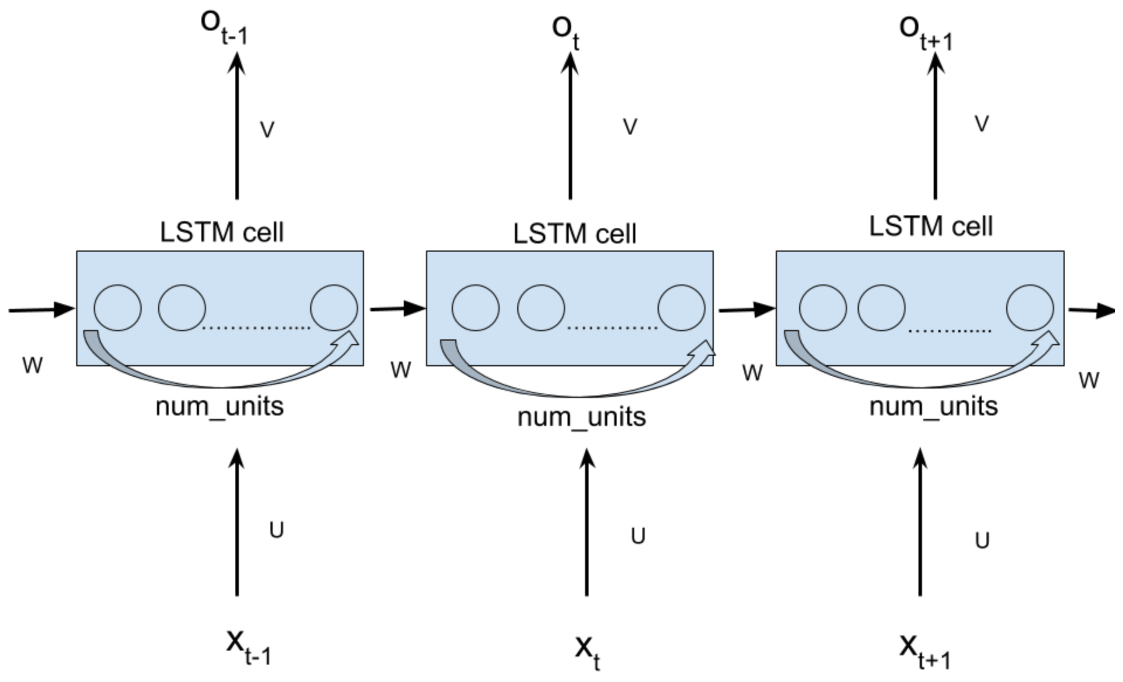


Figure 20: Interpreting LSTM units.
Source: jasdeep06's blog. Understanding LSTM in Tensorflow (MNIST dataset)

When the last final LSTM block is reached, outputs are passed on to a *Fully Connected layer* with three activations, like the number of classes (positive, negative and neutral). The number of weights among these two layers are equal to the product of the three activations and the number of units of the LSTM block u. This Fully Connected layer receives as inputs the result of the activations of the LSTM layer, which learns long-term dependencies in the documents, and compute three weighted sums of these inputs, adjusting the weights together with all the others in the network.

When classification involves only two classes, only one activation is necessary in this layer and a Hyperbolic Tangent Function (or a Sigmoid) pushes its output between -1 and 1. The more the value of this function is close to -1, the more the document is associated with one class. The opposite holds true when the value approaches 1. In a three-class classification, instead, three activations are needed, and their output is passed to a Softmax function, which generalizes the Sigmoid function for multi-class classification [47]. The Softmax function [14] computes the conditional probability of a class x_i with $i = (1,2,...,k)$ over all possible classes, as shown by Equation 32.

$$Softmax(x) = \frac{\exp(x_i)}{\sum_{j=1}^{k} \exp(x_j)} \tag{32}$$

It is also called Normalized Exponential function, because it normalizes its input vector, meaning its norm[15] is equal to one. The Softmax function takes values between 0 and 1 (Figure 21), and the sum of the probabilities of all classes is one, $\sum_{j=1}^{k} Softmax(x_i) = 1$.

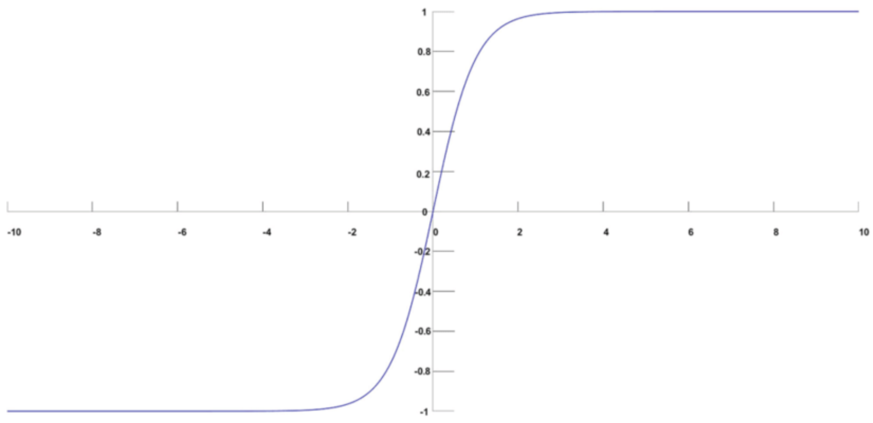


Figure 21: Softmax function.
Source: Leixian et al. "Fall Detection System Based on Deep Learning and Image Processing in Cloud Environment". In: Complex, Intelligent, and Software Intensive Systems. 2019, pp. 590-598. Scientific Figure on ResearchGate.

The final layer is the *Classification Output layer* [45] [35], which takes the probabilities provided by the Softmax layer and computes a particular type of loss function, called *Cross-Entropy Loss function*, for all documents i.

Cross-Entropy Loss function is used for classification tasks and it is described

[15]The norm of a vector $||x||$, with $x = (x_1, x_2, ..., x_n)$, denotes the magnitude of the vector and is represented by a scalar. In its Euclidean form $||x|| := \sqrt{\sum_{i=1}^{n} x_i^2}$.

by Equation 33.

$$Loss(y_i, \hat{y}_i) = -\sum_{i=1}^{n}\sum_{j=1}^{k} y_{ij} ln(\hat{y}_{ij}), \tag{33}$$

where y_{ij} represents the true class distribution of document i and is an indicator function that takes value one when the processed document i belongs to class j and zero otherwise. The output of the Softmax layer is $\hat{y}_{ij}$ and it represents the estimated probability of document i to belong to class j. The indexes n and k denote the number of documents in the Training set and the number of classes, which is equal to three (positive, negative and neutral).

Equation 34 shows an example of the Cross-Entropy Loss function computed in vector form for the individual document i, assuming document i belongs to class $j = 3$.

$$Loss(y_i, \hat{y}_i) = -\begin{bmatrix}0\\0\\1\end{bmatrix}\begin{bmatrix}ln(\hat{y}_{i1}) & ln(\hat{y}_{i2}) & ln(\hat{y}_{i3})\end{bmatrix} = -ln(\hat{y}_{i3}) \tag{34}$$

In this example, the sparse vector $y = [y_1, y_2, y_3] = [0, 0, 1]$ is multiplied by the logarithm of the transpose of the dense vector $\hat{y}$, $\hat{y}^T = [\hat{y}_1, \hat{y}_2, \hat{y}_3]^T$. As the three classes are mutually exclusive, only one element of the vector y_i has non-zero value. Therefore, the result of the equation is simply represented by the last line of Equation 34.

The network selects the weights that minimize the loss function for each individual document i and this implies minimizing the negative log likelihood of the true class j for training observation i, $Loss(y_i, \hat{y}_i) = -ln(\hat{y}_{ij})$, which can be interpreted as performing Maximum Likelihood Estimation (MLE)[16] to estimate the parameters [35].

2.2.3 LSTM network for text classification

Before presenting the different parameters used to train the network, I present the algorithm that performs the text classification.

The algorithm I used is *Adam* [48], which is widely adopted by practitioners and derives from Stochastic Gradient Descent method. Stochastic Gradient Descent method is a modification of Gradient Descent presented in Equation 13. Stochastic Gradient Descent (SGD) applies Gradient Descents method to subsets of the training data, called *mini batches*, instead of the whole Training set. Oftentimes, a momentum term is added to Stochastic Gradient Descent, as in Equation 35, in

[16]Maximum Likelihood Estimation (MLE) is a method used to estimate the parameters of a model. It does so by maximizing the logarithm of the Likelihood function, which represents the probability that the available sample data set was produced by the underlying model.

order to reduce the oscillation of the algorithm. The coefficient δ represents the effect of the previous step $(x_n - x_{n-1})$ on the current step, where x_n represents the vector of parameters at step n.

$$x_{n+1} = x_n - \gamma \nabla f(x_n) + \delta(x_n - x_{n-1}), n \geq 0 \tag{35}$$

Adam (Adaptive Moment Estimation), like SGD, adapts the parameters dividing the Training set into many subsets called mini-batches. A mini-batch corresponds to a fix number of instances in the Training set and parameters are adapted at each mini-batch. The number of mini-batches needed by the network to analyze the entire Training set represents an *epoch*. It uses two main terms, m_n and v_n, as shown in Equation 36. The former is useful as a momentum term and it represents the moving average of the gradient and substitutes the gradient itself in Adam algorithm. The latter includes a moving average of the element-wise squares of the gradient, where β_2 (the default value is $\beta_2 = 0.999$) is the decay rate of the moving average. The learning rate γ is divided by the square root of v_n, plus an additional term ϵ that avoid division by zero. In this way, the learning rate becomes variable and it adapts to the loss function.

$$\begin{gathered} x_{n+1} = x_n - \frac{\gamma m_n}{\sqrt{v_n} + \epsilon}, n \geq 0 \\ m_n = \beta_1 m_{n-1} + (1 - \beta_1)\nabla f(x_n) \\ v_n = \beta_2 v_{n-1} + (1 - \beta_2)[\nabla f(x_n)]^2 \end{gathered} \tag{36}$$

To train the network in Matlab, 192 different combinations of parameters for both labeling types were used, and the main parameters are presented as follows.

- Time window size;
- percentiles used to define classes;
- embedding dimension;
- number of hidden units of the LSTM blocks;
- gradient threshold;
- number of epochs;
- initial Learn Rate;
- validation patience.

As already anticipated, I used 10-minute and 20-minute *time windows* and I classified articles on the basis on returns and volatility log variations corresponding to the specific time window, or lagged time window. I used both 35th-65th and 45th-55th *percentiles*.

For the *embedding dimension*, which represents the length of the d-dimensional dense vector used by the Embedding layer, three values were used: 10, 50 and 100. For the number of hidden units used by the LSTM blocks, the values were 50 and 100. *Gradient threshold* is a fixed value that the gradient is not allowed to overcome. Indeed, using gradient threshold method, every time the Euclidean norm[17] of gradient becomes greater than this threshold, the algorithm clips it and brings it back to the threshold. This is to avoid gradient to diverge. The values I used as gradient threshold are 1 and 0.1.

The *initial learning rate* represents the step size used by the learning algorithm at each iteration to move towards the minimum of the loss function. First, I tried different values of initial learning rate, but then I decided to use only 0.0001, as I noticed that with larger values, the algorithm tends to run quickly and leads to unsatisfactory results.

After trying with 100, I fixed the *number of epochs* to 150, but the use of an *early stopping* method makes this number variable. Early stopping is a technique that avoids network over-training. During the learning phase, the network adjusts its parameters in order to learn relationship in the training data. However, while this iterative procedure decreases the error in the Training set, it might increase that of the Validation set if the network excessively tailors the parameters to the specific data set. As shown by Figure 22, in a perfect word, the learning phase should be stopped when the error function of the Validation set reaches its minimum. After this point, the network starts learning also the individual biases of the Training set, and the out of sample performance worsens. The *Validation Patience* is an early stopping technique that is useful to stop the algorithm before approaching the maximum number of epochs. The Validation Patience fix the number of times the loss function computed on the Validation set can be larger than the previous smallest loss [48]. This means the loss function on the Validation set must decrease towards zero and, if it increases too much, the training algorithm stops. After a few attempts with 10, 15, 20, 30 and 50, I decided to fix this value to 30, that is not too low, in order to allow the algorithm to run for enough time.

In addition to these variables, I decided to set two additional variables to their default values. The *mini-batch size* represents the number of training instances that are analyzed by the algorithm before updating the parameters, and I set this value to the default value of 128. The *Validation frequency*, instead, represents the number of instances used by the network to validate the training parameters on

[17]The Euclidean norm is $||x|| := \sqrt{\sum_{i=1}^{n} x_i^2}$.

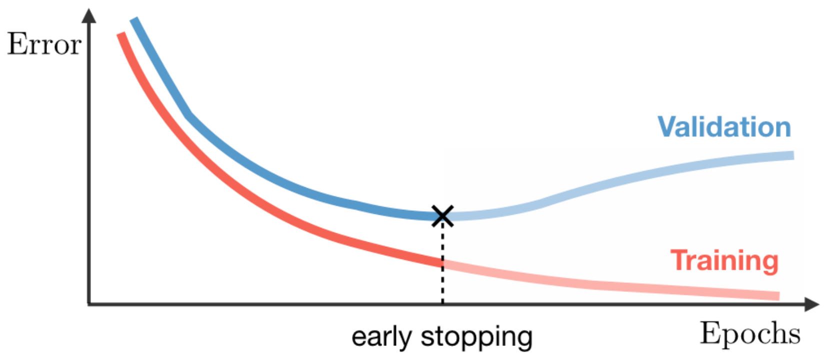


Figure 22: Early stopping.
Source: Afshine Amidi and Shervine Amidi. Deep Learning Tips and Tricks cheatsheet. Stanford University.

the Validation set. Its default value is 50. This means the network assesses the accuracy of the model once every 50 instances.

Table 1 shows the main results obtained by the network. The columns represent the accuracy of the model, the labeling technique used, the time window, the percentile used to define positive or negative variations of returns or volatility log variations, the embedding dimension, the number of units in each LSTM block of the LSTM layer and the Gradient Threshold.

Accuracy	Label	Time Window	Percentile	Lag	Embedding Dim.	N. units	GT
0.51619	Returns	20 min	45th-55th	no	100	100	0.10
0.50095	Returns	20 min	45th-55th	no	100	100	0.10
0.5054	Δ % Vix	20 min	45th-55th	no	100	100	0.10
0.52268	Δ % Vix	20 min	45th-55th	no	100	100	0.10
0.55411	Δ % Vix	10 min	45th-55th	no	100	100	0.10
0.51515	Δ % Vix	10 min	45th-55th	no	100	100	0.10
0.55399	Δ % Vix	10 min	45th-55th	yes	100	100	0.10
0.5681	Δ % Vix	10 min	45th-55th	yes	100	100	1

Table 1: Results of LSTM network applied on articles labeled with S&P 500 Index log returns and VIX Index log variations.
Source: Matlab

The table shows the best results obtained by this study, which were mainly achieved with VIX labeling. In all these cases, accuracy is higher than the random guess, 50%. For returns-based labeling, however, achieving this accuracy level was more difficult.

It is interesting to note that for VIX labeling, the highest accuracy values were achieved with 10-minute time windows, proving that more efficient markets, like the American one, discount information quickly. For the same reason, when using lagged observations, good accuracy was achieved only with 10-minute time windows.

This partially confirms the empirical findings about the time window size presented by Gidofalvi and Elkan [23], which proved that 20-minute time windows are optimal to capture market reactions to new information, and higher time windows tend to make results worse.

In all the attempts, 45th and 55th percentiles led to good accuracy results, while using 35th and 65th percentiles was not successful. In addition, the best results in terms of accuracy were produced using 100-dimensional embedding vectors and 100 hidden units in the LSTM blocks. These results show that using higher embedding dimension is beneficial in terms of accuracy, as it can better capture semantic relationships among words. Similarly, a high number of hidden units allows the LSTM nodes to learn more complex representations of the training data.

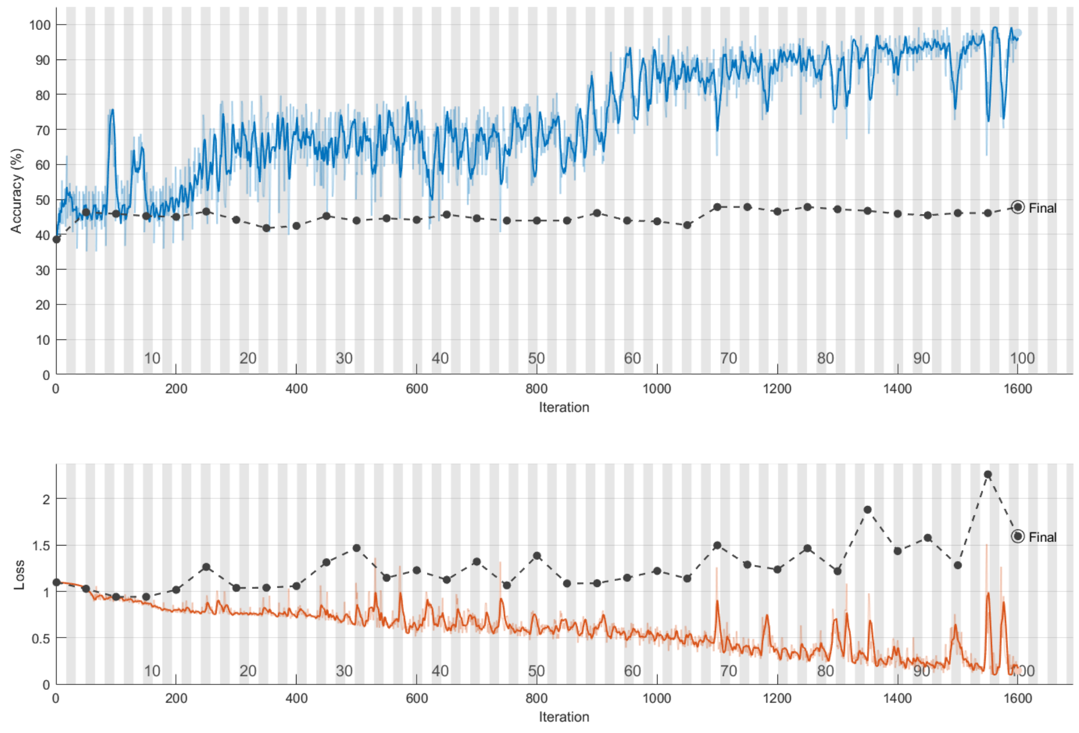


Figure 23: Accuracy and Loss function of the Training Process of the LSTM network. *Source: Matlab*

Figure 23 shows the training process of the network that produced an accuracy of 0.52268 on the Testing set. Accuracy is defined as the number of labels predicted correctly, over the total number of predicted labels. The blue line represents the accuracy of the model on the Training set, given the parameters fixed during each iteration. The dashed black line in the upper graph, instead, indicates the accuracy on the Validation set, computed at each mini-batch. The orange line represents the loss function, updated at every iteration, while the dashed black line indicates the loss function computed on the Validation set, at each mini-batch. In this case a Validation Patience was fixed to 30 and the algorithm stopped after 100 epochs, instead of reaching 150. The dashed line on the upper graph shows the final accuracy achieved on the Validation set, which is slightly lower than 50%. The accuracy on the Testing set was computed using the final parameters obtained by the model. The blue line increases an converges towards high levels and the loss function has opposite behavior. However, validation is not able to increase above 50%, meaning the process is not able to learn too much information. Using an early stopping technique avoided overfitting, which would have worsened the results on the Validation and Testing sets.

2.2.4 Dictionary-based approach to text classification

In addition to automatic labeling that uses returns and changes in the volatility of the S&P 500 Index, I decided to adopt another approach. Indeed, I tried to label each article following a dictionary-based approach. I decided to use three well known dictionaries, already presented in Chapter 2.

- Psychological Harvard IV-4 General Inquirer dictionary (GI);
- Loughran and McDonald's dictionary (LM);
- Henry's Dictionary (HE).

These three dictionaries provide a classification of positive and negative words, which can be used to assess if a document is positive or negative using a simple formula. Each article was labeled on the basis of how many positive and negative words it contains, and I repeated the process for each of the three dictionaries.

Equation 37 shows the formula used to define each label. A "positive" class was attributed to article i if the difference between the number of positive and negative words it contains is greater than zero. If this difference is negative the assigned class is "negative". If, instead, this formula returns zero, then the class of article i is "neutral".

$$Class_i = p\_words_i - n\_words_i \tag{37}$$

Table 2 shows some of the positive and negative words included in the three Dictionaries.

After labeling all articles according to the three dictionaries, I used the same LSTM network presented in the previous paragraph and I trained the model to recognize these words and predict the class of each article. I divided the data set into Training, Validation and Testing sets and I used the parameters listed in Table 3, which shows the result of the training process for the three dictionary-based labeling techniques. The columns represent the accuracy of the model, the type of dictionary used for labeling, the embedding dimension, the number of units in each

LSTM block, the Gradient Threshold, the number of epochs, the initial learning rate and the validation patience used as early stopping technique.

Henry's Dictionary		Harvard IV-4 General Inquirer Dictionary		Loughran and McDonald's Dictionary	
POSITIVE	NEGATIVE	POSITIVE	NEGATIVE	POSITIVE	NEGATIVE
above	below	abide	abandon	able	abandon
accomplish	challenge	ability	abandonment	abundance	abandoned
accomplished	challenged	able	abate	abundant	abandoning
accomplishes	challenges	abound	abdicate	acclaimed	abandonment
accomplishing	challenging	absolve	abhor	accomplish	abandonments
accomplishment	decline	absorbent	abject	accomplished	abandons

Table 2: Main words contained in Henry's, Loughran and McDonald's, and Harvard IV-4 Dictionaries.
Source: CRAN. Package 'SentimentAnalysis'

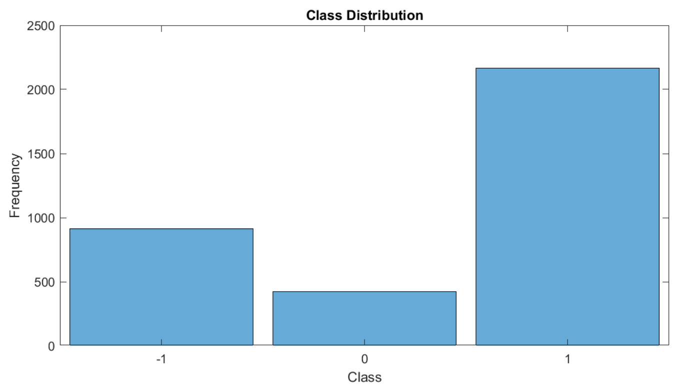
Accuracy	Label	Emb. Dim.	N. units	GT	N. epochs	Learn Rate	Val. Patience
0.94476	LM Dictionary	100	200	0.1	100	0.001	5
0.84571	GI Dictionary	100	200	0.1	100	0.001	10
0.6190	HE Dictionary	100	200	0.1	100	0.001	10

Table 3: Results of LSTM network applied on articles labeled with Dictionary-based techniques.
Source: Matlab

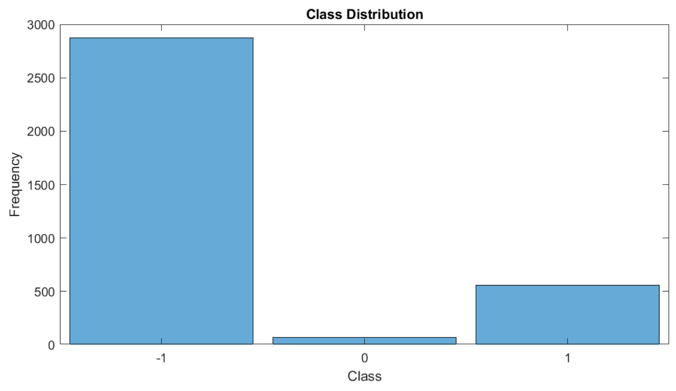
The best result was obtained for Loughran and McDonald's finance-specific Dictionary and Figure 24 explains why this result is not surprising. Indeed, Loughran and McDonald's Dictionary is the one that contains the higher relative number of negative words. Positive words lead to poor performances, as Loughran and McDonald [41] note, and the reason is that they are often followed or preceded by negations, which change the meaning of the word, biasing the results.

Figure 25 shows the training process run on the articles labeled with Loughran and McDonald's Dictionary. As explained in the previous paragraph, the graph on the top represents the accuracy of the model along the mini-batches. The blue line computes the accuracy on the Training set, while the dashed black line the accuracy on the Validation set. The graph on the bottom, instead, represents the loss function computed at each mini-batch. The accuracy of the Training set rapidly approached high values and the loss function, shown in orange, goes in the opposite direction, decreasing sharply and converging towards zero. Both graphs clearly show that the training process stopped before reaching the maximum number of epochs, as a result of the early stopping technique that was adopted. The dashed line on the top follows the blue line quite well, the one on the bottom decreases together with the orange line, and then increases a bit, activating early stopping.

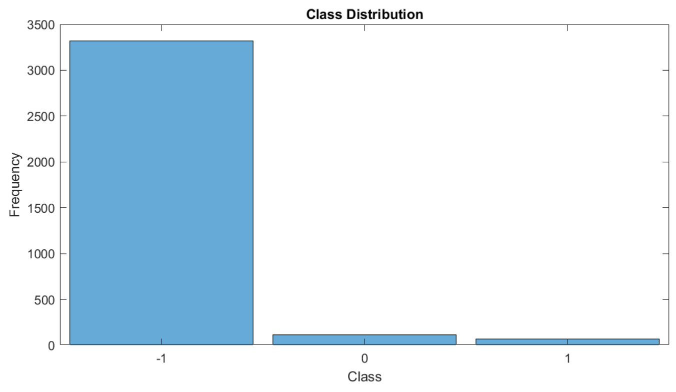
The network was able to learn LM Dictionary classification with high accuracy. When it comes to GI and HE Dictionaries, accuracy decreases to 0.84571 and



(a) *HE Dictionary.*



(b) *GI Dictionary.*



(c) *LM Dictionary.*

Figure 24: Histogram representation of article classes based on HE, GI and LM dictionaries.
Source: Matlab.

0.6190, both of them better than the random guess 0.50. At the end of the process the network reached an accuracy on the out of sample Testing set of about 94%, which is the percentage of articles labeled correctly.

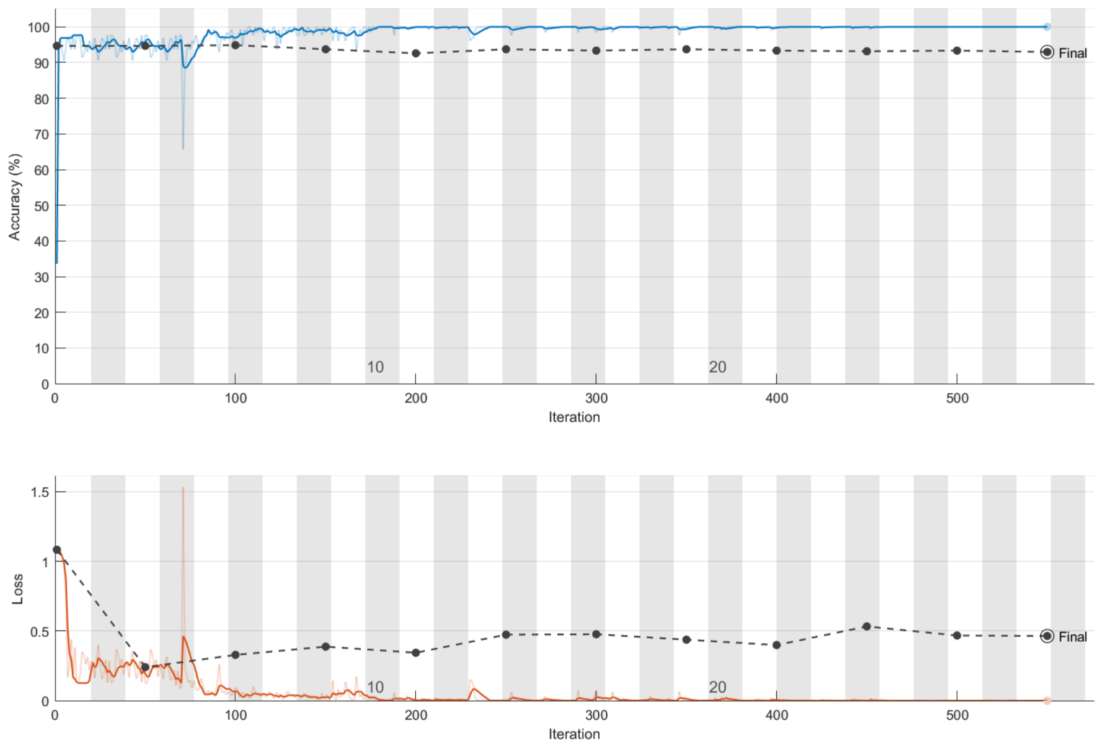


Figure 25: Accuracy and Loss function of the Training Process applied on articles labeled with LM Dictionary.
Source: Matlab

3 Investor Sentiment contribution to stock index returns and volatility variations

3.1 Introduction to the problem

Previous chapters mentioned the importance of alternative sources of information, such as social media contents and news articles, in many fields and some papers have proved they can be beneficial to investment decisions.

The previous chapter already proved in same sense the existence of a relationship between VIX log variations and text-based investor sentiment, even though VIX Index considers implied volatility registered on the options market and not on the stock market. Some kind of relationship was shown also with respect to stock index log returns, as it was possible to classify news articles with fair accuracy. In both cases accuracy was higher than the random guess, but results were far from 100%. Chapter 2 focuses on classifying news articles and defining what can be seen as proxies for investor sentiment. However, it is interesting to assess the actual explanatory power of these sentiment indicators.

In this chapter, I investigate whether the sentiment variables presented so far can explain stock index returns or volatility. While returns can be easily computed from stock prices, volatility is not directly measurable, exactly like market sentiment. Therefore, univariate volatility needs to be estimated.

For this aim I used S&P 500 Index close prices to compute log returns, and I estimated volatility with a type of Generalized Autoregressive Conditional Heteroskedasticity model. I followed the same approach of Rupande et al. [61], who investigate the explanatory power of their market-based investor sentiment proxy introducing it into the mean and variance equation of a GARCH model applied on the South African stock index. Unlike Rupande et al., I did not use any market-based proxy of investor sentiment for this analysis and I dealt with a much more efficient market than the South African one.

In the next paragraphs, I briefly introduce GARCH models and then I present the main findings of my work.

3.1.1 A brief introduction to GARCH models

Generalized Autoregressive Conditional Heteroskedasticity models are among the techniques used to estimate volatility. One of the biggest challenges in volatility modeling is the presence of volatility clusters, which suggests that recent observations should be given a higher weight [13]. The first models address this issue by considering simple or exponential moving averages of past squared returns over an estimation window.

Engle [16] in 1982 introduced the first GARCH type model, the Autoregressive Conditional Heteroskedasticity model, ARCH(p), described by Equation 38.

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i Y_{t-i}^2 \tag{38}$$

Conditional variance is represented by a constant, ω, plus a weighted sum of past squared returns Y_{t-i}^2, with p indicating the number of lags considered by the ARCH model [13]. Returns Y_t are expressed as $Y_t = \sigma_t Z_t$, where σ_t is the conditional volatility and Z_t is a white noise standardized random variable with zero mean and unit variance, often assumed to be Normally or Student-t distributed.

ARCH models have zero conditional and unconditional mean and they have constant unconditional variance and time varying conditional variance. For ARCH(1) model, which is the simplest ARCH model, for instance, unconditional variance is defined as in Equation 39:

$$\sigma^2 = \frac{\omega}{1-\alpha} \tag{39}$$

In ARCH models [15], Y_t is serially autocorrelated, but not independent, which means Y_t^2 is autocorrelated. ARCH models are particularly suitable for modeling conditional variance of stock returns, whose distribution is often known to be fat tailed. Indeed, even when Z_t is assumed to be normally distributed, conditional returns are normal, but unconditional distribution of returns is fat tailed.

The main drawback of this model is its complexity. Indeed, it is often the case that many lags are necessary to appropriately model volatility. Besides the fact that a high number of lags makes estimation particularly hard, it also requires a longer sample time series for modeling purposes. The solution to this issue comes from the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models developed by Bollerslev in 1986 [10], which generalize the ARCH model introduced by Engle.

GARCH models [13] include one or more lagged volatility terms in the equation for conditional volatility, as in Equation 40:

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i Y_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2 \tag{40}$$

The first part of the equation is exactly like Equation 38, while the second part represents the weighted sum of past variances. The parameter p indicates the number of lags related to past squared returns, q the number of lags related to past variances, and the coefficients β_j represent the decaying effect of past variances on current conditional variance.

GARCH(p,q) models share the same properties of ARCH models, but they are much easier to estimate. The simplest example of GARCH model is GARCH(1,1), which can capture the long-term dependencies of high-order ARCH processes. The unconditional variance of a GARCH(1,1) is described by Equation 41.

$$\sigma^2 = \frac{\omega}{1-\alpha-\beta} \tag{41}$$

To ensure variance to be positive and to assume second order stationarity[18], ARCH and GARCH models are subject to some constraints.

The first and more important constraint regards non-negativity of the conditional volatility and implies that all parameters in Equation 40, ω, α_i and β_j, must be strictly positive. Equation 42 represents these constraints for all $i > 0$ and $j > 0$.

$$\omega, \alpha_i, \beta_j > 0 \tag{42}$$

In addition, to ensure second order stationarity, the sum of α_i and β_j coefficients must be lower than 1.

In a GARCH(1,1) process, this constraint is represented by Equation 43:

$$\alpha + \beta < 1 \tag{43}$$

If $\alpha + \beta = 1$, unconditional variance expressed by Equation 41 explodes towards infinite, while if $\alpha + \beta > 1$, it is not defined.

Generalized Autoregressive Conditional Heteroskedasticity models represent an important innovation in volatility modeling and some academicians have developed alternative models that are based on GARCH.

Nelson in 1991 [54] introduced the Exponential GARCH model, which takes into consideration the asymmetric effect of returns on volatility. Stock returns are often negatively correlated with volatility, because price drops deteriorate the market price of equity and, thus, increase leverage, making the company riskier [15]. In addition, risk aversion makes investors more sensitive to losses than gains [9] and creates panic selling when market prices decrease.

[18] A stochastic process is *Second Order Stationary* if it has constant mean, constant variance and its autocovariance and autocorrelation functions, which define the covariance between the same random variable at different time instants, depend only on the distance between time instants.

Exponential GARCH [13] [22], or E-GARCH, defines the logarithm of the conditional volatility as in Equation 44:

$$log(\sigma_t^2) = \omega + \sum_{j=1}^{p}[\alpha_j(Z_{t-j}) + \delta_j(|Z_{t-j}| - E(|Z_{t-j}|))] + \sum_{j=1}^{q}\beta_j log(\sigma_{t-j}^2) \tag{44}$$

The parameters δ_j and α_j represent the leverage effect. The former captures the size or magnitude effect, while the latter the sign effect. This model uses past errors Z_{t-j}, instead of past squared returns, and past log variances σ_{t-j} instead of variances.

One of the advantages of this model is that it does not require any non-negativity condition, because conditional volatility, expressed in exponential form, is always positive. It still requires some conditions to be second order stationary. For E-GARCH(1,1) model, to ensure unconditional variance does not explode towards infinite, the parameter β_1 must be lower than one in absolute value. Therefore, the necessary and sufficient condition to ensure second order stationarity in the E-GARCH(1,1) is $|\beta_1| < 1$ [54] [26].

3.2 GARCH models and sentiment variables: empirical evidence

In this section I present the main empirical findings of the analysis of the sentiment variables obtained through supervised Machine Learning techniques.

To assess the explanatory power of these sentiment variables on stock market returns, I used S&P 500 Index intraday log returns computed from May 14 to November 7, 2019, using both 10-minute and 20-minute time intervals. Figure 26 shows these two time series with respectively 4983 and 2619 observations.

First, I modeled the conditional volatility of stock index returns using a GARCH type model and then I introduced the sentiment variables into the mean and variance equations of the GARCH, looking for improvements in model fitting. All variables underwent unit root tests in order to assess their stationarity. S&P 500 Index 10-minute returns were modeled using an ARCH(1) model with the addition of Autoregressive model of order 1, AR(1). The conditional distribution that was used is a Skew Student's t-distribution, because of the presence of left skewness and fat tails in log returns distribution. Fitting this model leads to uncorrelated residuals and squared residuals, as shown by the estimated Autocorrelation functions in Figure 27. Ljung-Box and McLeod Li tests confirm this result, not rejecting the null hypothesis of no serial correlation for all lags. Table 4 presents the model, where all coefficients are significantly different from zero, except for the intercept, and standard errors are the robust ones based on the method of White [22]. Skewness

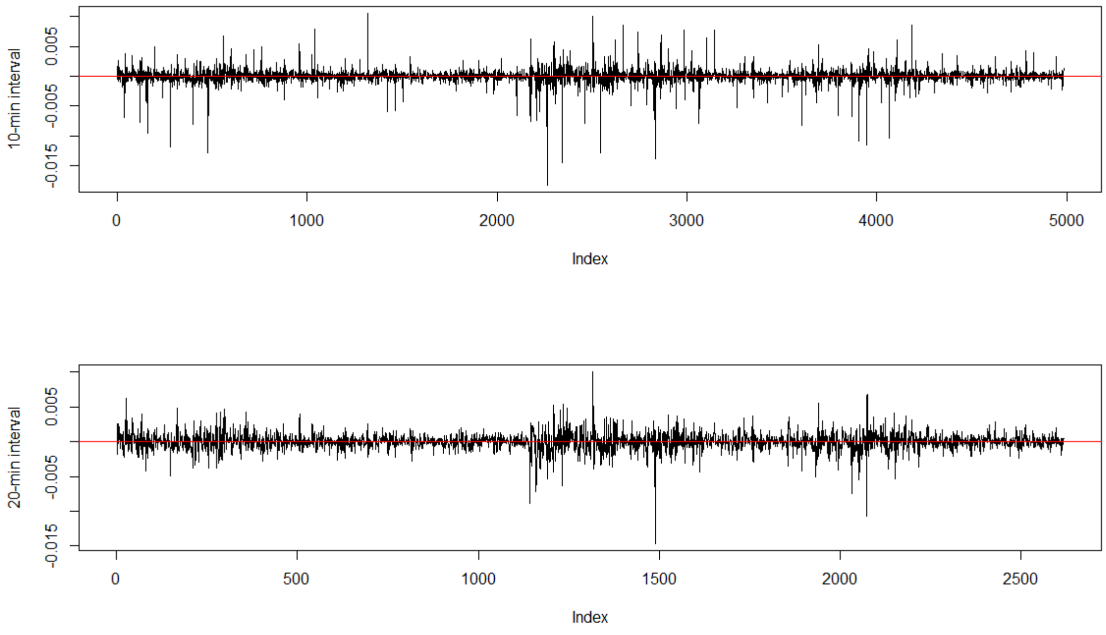
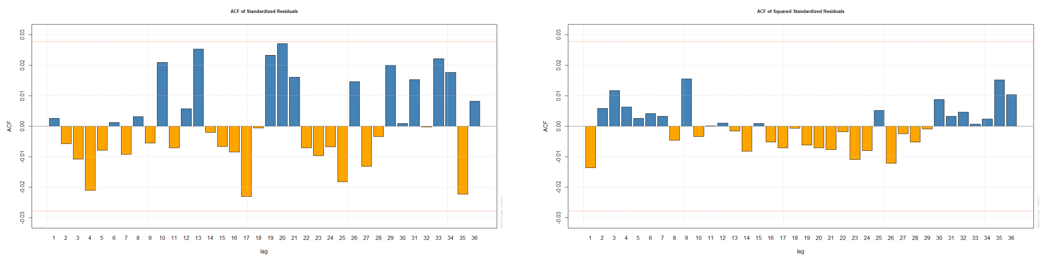


Figure 26: Plot of S&P 500 Index log returns with 10-minute and 20-minute time intervals from May 14 to November 7, 2019.
Source: R.



(a) *Estimated ACF of residuals.* **(b)** *Estimated ACF of squared residuals.*

Figure 27: Estimated Autocorrelation function of residuals and squared residuals of ARCH(1) and AR(1) models applied on S&P 500 10-minute log returns.
Source: R, Rugarch package.

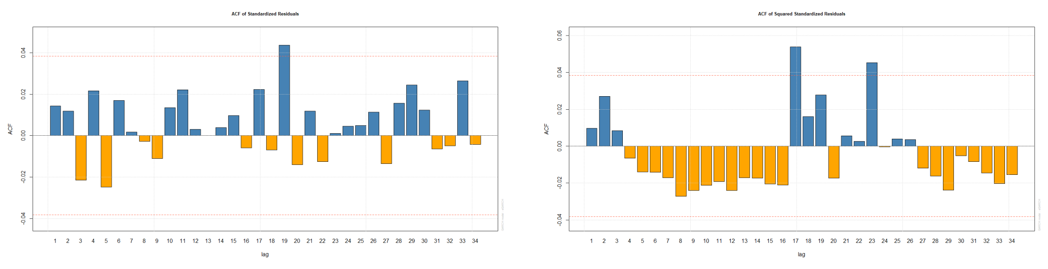
is positive as expected and shape parameter is quite low, signaling the presence of fat tails.

When dealing with 20-minute returns, instead, an E-GARCH(1,1) model was adopted, again with the assumption of Skew Student-t conditional distribution. Figure 28 shows the estimated Autocorrelation function of residuals and squared residuals of this model, where uncorrelation holds for all reasonable lags. This is also confirmed by Ljung-Box and McLeod Li tests. Table 5 presents the estimates

	Estimate	Std. Error	t-value	p-value	
mu	3.966450e-06	1.505715e-05	0.2634262	7.922221e-01	
ar1	1.370350e-01	1.715969e-02	7.9858656	1.332268e-15	***
omega	1.703900e-06	1.124324e-07	15.1548802	0.000000e+00	***
alpha1	9.983071e-01	1.404554e-01	7.1076471	1.180389e-12	***
skew	9.531758e-01	1.646085e-02	57.9056120	0.000000e+00	***
shape	2.236005e+00	1.276375e-02	175.1840535	0.000000e+00	***

Table 4: ARCH(1) and AR(1) models fitted on S&P 500 Index intraday log returns with 10-minute time intervals.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

of the parameters of this model and uses Heteroskedasticity-robust standard errors. Again, skew and shape parameters behave as expected.



(a) *Estimated ACF of residuals.* **(b)** *Estimated ACF of squared residuals.*

Figure 28: Estimated Autocorrelation function of residuals and squared residuals of E-GARCH(1,1) model applied on S&P 500 20-minute log returns.
Source: R, Rugarch package.

3.2.1 Explanatory power of automatic labeling-based sentiment variables

After having specified a suitable model to fit the S&P 500 Index log returns, I introduced the sentiment variables presented in previous chapters into the mean and variance equations of these models, depending on their specific time window (10 or 20 minutes). All the sentiment variables are summarized in Table 6, which recalls the results of the learning processes presented in Table 1. The columns refer to the sentiment variable name, the accuracy of the model computed on the Testing set, the label type, the time window and the presence of lags. Figure 29 shows time

	Estimate	Std. Error	t-value	p-value	
mu	7.442975e-06	2.171365e-05	0.3427787	7.317650e-01	
omega	-3.774281e-01	9.315179e-03	-40.5175371	0.000000e+00	***
alpha1	-6.092010e-02	1.456591e-02	-4.1823746	2.884801e-05	***
beta1	9.720799e-01	6.486486e-04	1498.6231387	0.000000e+00	***
gamma1	2.081576e-01	6.261149e-03	33.2459123	0.000000e+00	***
skew	9.848583e-01	2.712338e-02	36.3103061	0.000000e+00	***
shape	4.064593e+00	3.239540e-01	12.5468191	0.000000e+00	***

Table 5: E-GARCH(1,1) model fitted on S&P 500 Index intraday log returns with 20-minute time intervals.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

Variable name	Accuracy on Testing set	Label	Time Window	Lag
S_1	0.51619	Returns	20 min	no
S_2	0.50095	Returns	20 min	no
S_3	0.5054	Δ % Vix	20 min	no
S_4	0.52268	Δ % Vix	20 min	no
S_5	0.55411	Δ % Vix	10 min	no
S_6	0.51515	Δ % Vix	10 min	no
S_7	0.55399	Δ % Vix	10 min	yes
S_8	0.5681	Δ % Vix	10 min	yes

Table 6: Sentiment variables to introduce into mean and variance equations of GARCH type models.
Source: Matlab.

series of sentiment variables computed over 20-minute time intervals, while Figure 30 shows those computed over 10-minute time windows.

It is worth mentioning that there is a small difference between 20-minute time intervals used for computing S&P 500 log returns and those used for VIX log variations. The format of returns-based 20-minute time windows represents observation as *hh*:00, *hh*:20 and *hh*:40, where *hh* is the hour in a 24-hours time system and 00, 20, and 40 are minutes. The format of VIX-based 20-minute time intervals, instead, given prices available on Bloomberg Terminal, represents observation as *hh*:15, *hh*:35 and *hh*:55. Therefore, when performing regression between S&P log returns and news sentiment based on VIX log variations, it is important to remember that sentiment is slightly lagged by 5 minutes. The same problem holds true when considering 10-minute time intervals.

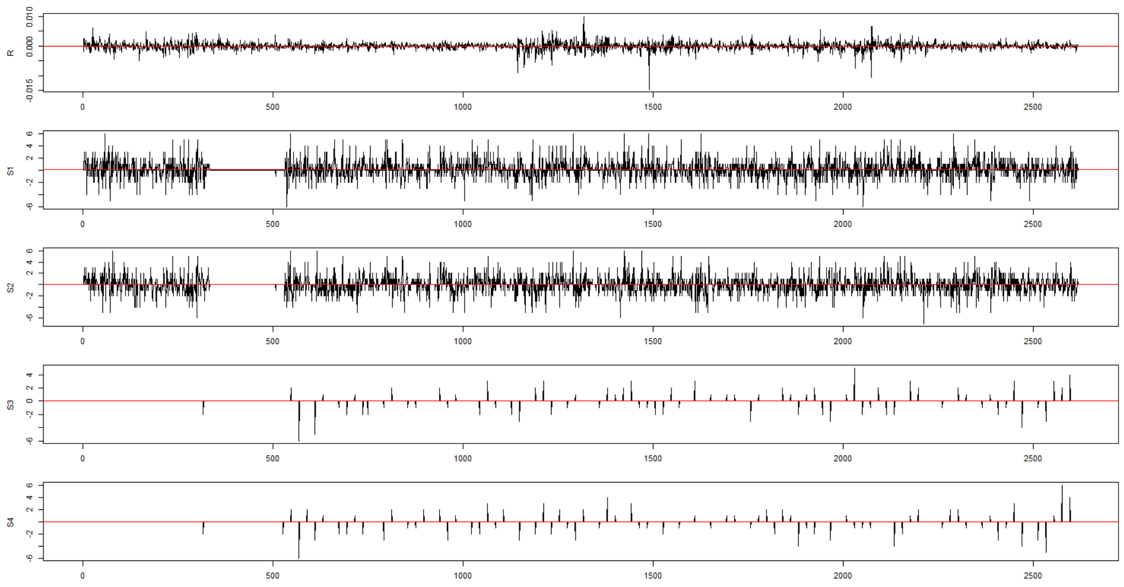


Figure 29: Log returns and Sentiment variables computed over 20-minute time intervals. *Source: R.*

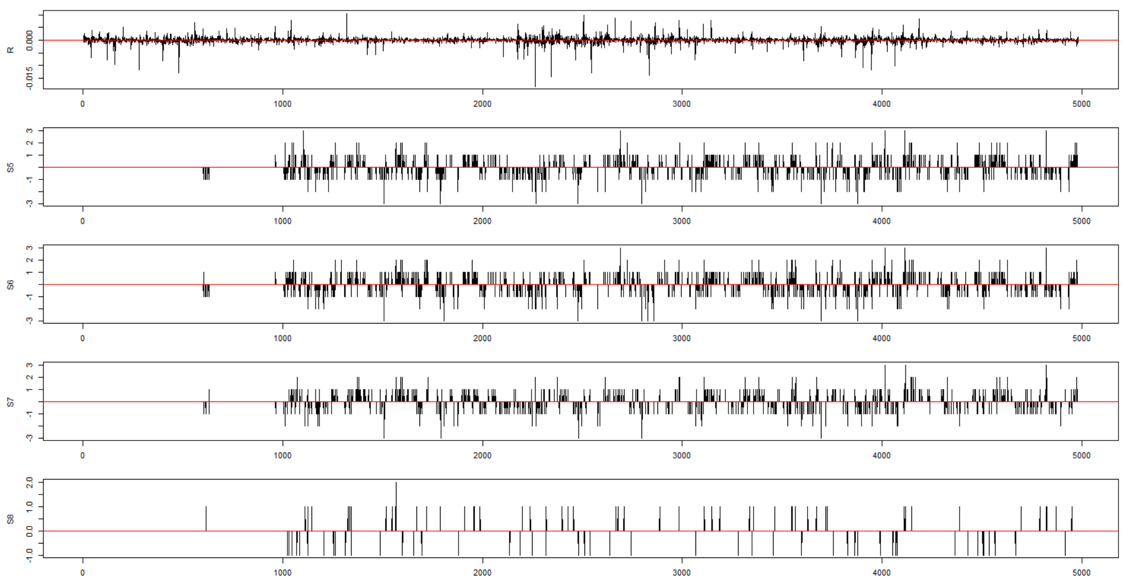


Figure 30: Log returns and Sentiment variables computed over 10-minute time intervals. *Source: R.*

In previous chapters I explain that the whole data set was divided into Training, Validation and Testing set before running the learning algorithm. These sets were built through random sampling and, thus, they do not contain only consecutive observations, but they can be represented as a sparse time series with many zero values and some observations (about 15% of the total number of news articles). To make this concept clearer, Figure 31 shows the sparse time series of Sentiment values computed for news articles belonging to the Testing set, thus not related to the learning process. The plot on the top shows the time series of the sentiment variable S_1, computed by applying the learned model to the articles included in the whole data set. The plot on the bottom, instead, represents the sparse time series of sentiment S_1 computed over articles that belong to the Testing set only. The former is itself a sparse time series, but it is less sparse. Articles were classified on the basis

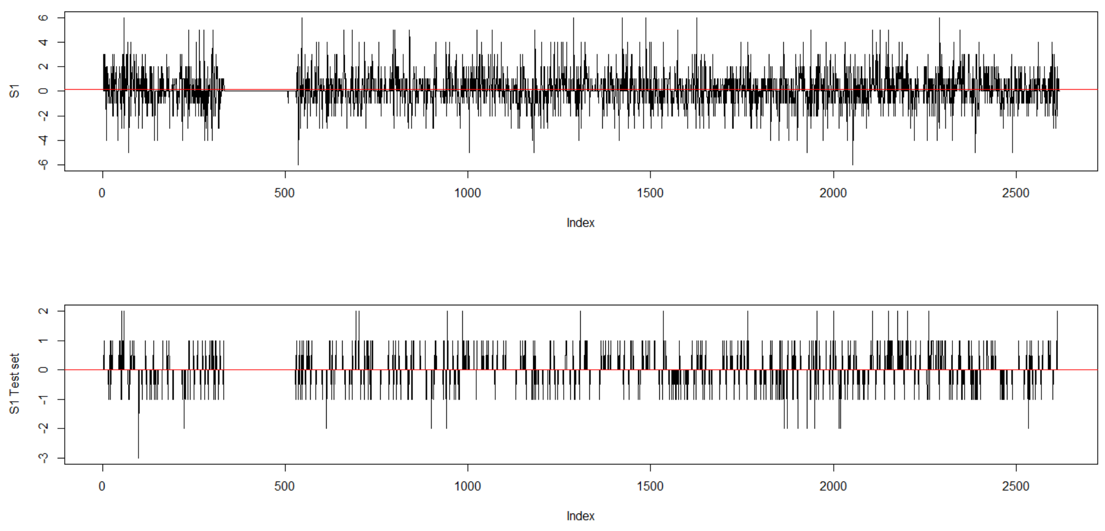


Figure 31: Sentiment variable S_1 computed using the whole data set and the Testing set, respectively.
Source: R.

of the parameters obtained through the learning process and they were assigned value 1 if they are positive, -1 if negative, 0 if neutral. However, as some time intervals include more than one article, the values assigned to articles belonging to the same time window are algebraically summed up. For this reason, the value of some of the observations in the sentiment time series presented in Figure 31 is not included in the interval $[-1, +1]$.

As sparser time series may show less empirical significance, I used sentiment values computed on the whole data set. However, I analyzed also the Testing set and, in some cases, sentiment variables are still significant. The sentiment variable S_1, obtained classifying articles using the model that associates them with S&P 500 Index 20-minute log returns, was introduced into the mean and variance equations of the E-GARCH(1,1) model. Table 7 shows the results for the sentiment variable S_1 representing the classification of the entire data set of articles and it confirms its explanatory power in both equations, where $mxreg1$ indicates the sentiment variable in the mean equation and $vxreg1$ in the variance equation. Significance is at the 99% for $mxreg1$ and 95% for $vxreg1$, and both coefficients have the expected sign. Indeed, sentiment S_1 was built on the basis of log returns and positive sentiment should be associated with positive variations in the S&P 500 Index. With regard to the variance equation, instead, $vxreg1$ represents the negative relation between sentiment and volatility. Positive sentiment (associated with positive log returns) is linked to decreasing volatility, because of the asymmetric effect of stock returns on volatility. As investors are risk averse and give more weight to losses than gains [9], downtrends can push investors to sell and this creates larger price movements which are reflected in higher volatility. This is particularly true in case of panic selling. When using the sparser time series, the one that considers only

	Estimate	Std. Error	t-value	p-value	
mu	-2.980005e-05	1.877663e-05	-1.587082	0.1124941629	
mxreg1	2.546154e-04	1.141802e-05	22.299446	0.0000000000	***
omega	-1.198721e-01	3.059420e-03	-39.181316	0.0000000000	***
alpha1	-4.670968e-02	1.272715e-02	-3.670082	0.0002424724	***
beta1	9.908866e-01	1.067440e-04	9282.830110	0.0000000000	***
gamma1	1.167086e-01	9.070969e-03	12.866162	0.0000000000	***
vxreg1	-2.093745e-02	9.274082e-03	-2.257630	0.0239687224	**
skew	1.000324e+00	2.198744e-02	45.495255	0.0000000000	***
shape	3.127035e+00	2.109837e-01	14.821216	0.0000000000	***

Table 7: E-GARCH(1,1) model fitted on S&P 500 Index intraday log returns with 20-minute time intervals with Sentiment S_1 as exogenous variable in the mean and variance equations.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

article in the Testing set, the coefficient of the sentiment variable in the mean equation is still significantly different from zero, even though this does not hold for the variance equation.

The same model applies for the sentiment variable S_2, which was obtained using the same time window, but produced a slightly lower accuracy. Here too, the estimate of the coefficient in the mean equation is significantly different from zero (Table 8), but the one in the variance equation is not significant when it is introduced. Again, when using the sparse representation of S_2, the one based on the Testing set only, sentiment is still significant in the mean equation.

The same procedure was followed for variables S_3 and S_4, which use 20-minute time intervals, too, but they are based on VIX log variations instead of Index log returns. As observations for VIX Index are available only from June 5 to November

	Estimate	Std. Error	t-value	p-value	
mu	8.670035e-06	1.628008e-05	0.5325547	5.943419e-01	
mxreg1	2.548016e-04	1.231658e-05	20.6876954	0.000000e+00	***
omega	-9.030236e-02	2.633125e-03	-34.2947542	0.000000e+00	***
alpha1	-4.996589e-02	1.164639e-02	-4.2902481	1.784736e-05	***
beta1	9.933408e-01	1.252367e-04	7931.7094487	0.000000e+00	***
gamma1	1.027694e-01	9.074579e-03	11.3249779	0.000000e+00	***
skew	9.652446e-01	1.970511e-02	48.9844924	0.000000e+00	***
shape	3.145021e+00	2.255473e-01	13.9439552	0.000000e+00	***

Table 8: E-GARCH(1,1) model fitted on S&P 500 Index intraday log returns with 20-minute time intervals with Sentiment S_2 as exogenous variable in the mean equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

7, 2019, I reduced the number of observations of the S&P 500 Index log returns to match them with this new time window. The model applied on this new subset of data is still E-GARCH(1,1), so nothing changes. Table 9 and Table 10 show the most relevant results. While the coefficient of S_3 is significantly different from zero when introduced in the mean equation $mxreg1$ of E-GARCH(1,1) model, the one of S_4 is significantly different from zero in the variance equation $vxreg1$. Coefficient have again their expected sign, with volatility negatively influenced by sentiment S_4. In this case sentiment was built on the basis of VIX Index and positive variation of the index were associated with negative sentiment, because of the asymmetric effect of volatility. Therefore, $vxreg1$ reflects the predictable positive relation between volatility computed on S&P 500 Index log returns and VIX Index itself.

When considering 10-minute time intervals, no statistical significance was found with respect to the variance equation, but interesting results were achieved for the mean equation. Log returns over 10-minute time intervals were modeled using an ARCH(1) model with the addition of an autoregressive component of order 1, AR(1), as shown before. Even when reducing the length of the time series of the S&P 500 Index log returns to adapt it to VIX-based sentiment (from June 5 to November 7, 2019), the model remains the same. Sentiment variables S_5 and S_6 were introduced into the mean equation of the ARCH(1) model and in both cases, coefficients are significant, even with the addition of the AR(1) model. In the case of S_6, however, the coefficient for α is equal to 1, violating the constraint for second order stationarity. When imposing the constraint, the coefficient for S_6 remains significant. Results are shown in Table 11 and Table 12.

The last two sentiment variables, S_7 and S_8, are the result of the learning process based on automatic labeling technique, where articles belonging to a given

	Estimate	Std. Error	t-value	p-value	
mu	8.362319e-06	0.000023135	0.3614575	0.7177574879	
mxreg1	2.871462e-04	0.000116819	2.4580437	0.0139696185	**
omega	-4.988590e-01	0.023114450	-21.5821281	0.0000000000	***
alpha1	-5.794299e-02	0.016738266	-3.4617082	0.0005367588	***
beta1	9.632287e-01	0.001685761	571.3910110	0.0000000000	***
gamma1	2.421911e-01	0.009252674	26.1752523	0.0000000000	***
skew	9.766263e-01	0.029313134	33.3170201	0.0000000000	***
shape	3.912570e+00	0.329895116	11.8600414	0.0000000000	***

Table 9: E-GARCH(1,1) model fitted on S&P 500 Index intraday log returns with 20-minute time intervals with Sentiment S_3 as exogenous variable in the mean equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

	Estimate	Std. Error	t-value	p-value	
mu	7.806203e-06	2.269457e-05	0.3439679	0.7308704262	
omega	-4.038238e-01	1.219396e-02	-33.1167124	0.0000000000	***
alpha1	-6.019315e-02	1.563671e-02	-3.8494765	0.0001183705	***
beta1	9.702983e-01	8.401103e-04	1154.9653134	0.0000000000	***
gamma1	2.111217e-01	7.193224e-03	29.3500840	0.0000000000	***
vxreg1	-8.167910e-02	4.014461e-02	-2.0346219	0.0418889311	**
skew	9.754886e-01	2.898713e-02	33.6524664	0.0000000000	***
shape	3.936890e+00	3.309971e-01	11.8940325	0.0000000000	***

Table 10: E-GARCH(1,1) model fitted on S&P 500 Index intraday log returns with 20-minute time intervals with Sentiment S_4 as exogenous variable in the variance equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

time interval were classified as positive if the subsequent time interval registered a negative VIX Index log variation. Therefore, they were introduced in the model as lagged variables. Table 13 and Table 14 show the result, where $mxreg1$ represents the positive effect of past sentiment on current Index log returns.

S_7 is statistically significant at the 99% level, while S_8 at the 95% level. The significance of S_8 is even more relevant considering that it contains only articles that belongs to the Testing set. In addition, the fact that this sentiment variable explains subsequent log returns in the S&P Index makes it particularly interesting for prediction purposes. The case of S_8 is similar to that of S_6, with the coefficient for α that is equal to 1 and violates the constraint for second order stationarity. Here too, when imposing the constraint, the coefficient for S_8 remains significant.

	Estimate	Std. Error	t-value	p-value	
mu	7.042988e-06	1.530042e-05	0.4603134	6.452913e-01	
ar1	1.273821e-01	1.964287e-02	6.4849006	8.879031e-11	***
mxreg1	4.592985e-05	2.058660e-05	2.2310553	2.567746e-02	**
omega	1.553271e-06	1.079462e-07	14.3893094	0.000000e+00	***
alpha1	9.989864e-01	1.494660e-01	6.6837044	2.329759e-11	***
skew	9.482215e-01	1.761249e-02	53.8380207	0.000000e+00	***
shape	2.240371e+00	1.650998e-02	135.6979724	0.000000e+00	***

Table 11: ARCH(1) and AR(1) models fitted on S&P 500 Index intraday log returns with 10-minute time intervals with Sentiment S_5 as exogenous variable in the mean equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

	Estimate	Std. Error	t-value	p-value	
mu	7.939078e-06	1.532829e-05	0.5179363	6.045027e-01	
ar1	1.260838e-01	1.735062e-02	7.2668175	3.681500e-13	***
mxreg1	5.325421e-05	2.082264e-05	2.5575151	1.054230e-02	**
omega	1.614932e-06	8.998988e-08	17.9457102	0.000000e+00	***
alpha1	1.000000e+00	1.402630e-01	7.1294640	1.007638e-12	***
skew	9.497868e-01	1.766126e-02	53.7779674	0.000000e+00	***
shape	2.231162e+00	1.661229e-02	134.3079258	0.000000e+00	***

Table 12: ARCH(1) and AR(1) models fitted on S&P 500 Index intraday log returns with 10-minute time intervals with Sentiment S_6 as exogenous variable in the mean equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

	Estimate	Std. Error	t-value	p-value	
mu	7.733672e-06	1.588303e-05	0.486914	6.263192e-01	
ar1	1.270495e-01	1.865763e-02	6.809518	9.792611e-12	***
mxreg1	5.232186e-05	2.246800e-05	2.328728	1.987346e-02	**
omega	1.571057e-06	1.414926e-07	11.103455	0.000000e+00	***
alpha1	9.989993e-01	1.814273e-01	5.506335	3.663812e-08	***
skew	9.486055e-01	1.776720e-02	53.390831	0.000000e+00	***
shape	2.237188e+00	1.391400e-02	160.786907	0.000000e+00	***

Table 13: ARCH(1) and AR(1) models fitted on S&P 500 Index intraday log returns with 10-minute time intervals with lagged Sentiment S_7 as exogenous variable in the mean equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

	Estimate	Std. Error	t-value	p-value	
mu	7.616759e-06	1.473716e-05	0.5168402	6.052677e-01	
ar1	1.290559e-01	1.943411e-02	6.6406928	3.122125e-11	***
mxreg1	1.578665e-04	6.061979e-05	2.6042077	9.208695e-03	***
omega	1.595593e-06	1.273945e-07	12.5248142	0.000000e+00	***
alpha1	1.000000e+00	1.621201e-01	6.1682662	6.904286e-10	***
skew	9.488065e-01	1.727392e-02	54.9271032	0.000000e+00	***
shape	2.234101e+00	1.308588e-02	170.7260739	0.000000e+00	***

Table 14: ARCH(1) and AR(1) models fitted on S&P 500 Index intraday log returns with 10-minute time intervals with lagged Sentiment S_8 as exogenous variable in the mean equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

3.2.2 Explanatory power of dictionary-based sentiment variables

After proving that the sentiment variables obtained through the classification that associates news with S&P 500 Index log returns and VIX log variations contribute to the explanatory power of GARCH models, it is interesting to assess also the impact of dictionary-based sentiment variables. Therefore, in this paragraph I present the analysis of the three sentiment variables based on positive and negative words of Harvard IV-4 (GI), Loughran and McDonald's (LM) and Henry's (HE) dictionaries. For each dictionary, I used the model with the parameters learned during the training phase to classify the news contained in the whole data set. The LSTM network proved to perform this classification task with high accuracy and provided a time series of observations (with value −1, 0 or +1) that represent the class of each articles.

To assess the relationship between these sentiment variables and S&P 500 Index, articles with their corresponding class value were divided into 20-minute time intervals, depending on the time they had been published. When more than one article belonged to the same time interval, their value was algebraically summed up to one single value. The resulting time series represents the dictionary-based sentiment variable used for the analysis of the relationship with 20-minute log returns from May 14 to November 7, 2019. Depending on the dictionary used, the three sentiment variables are represented by Figure 32.

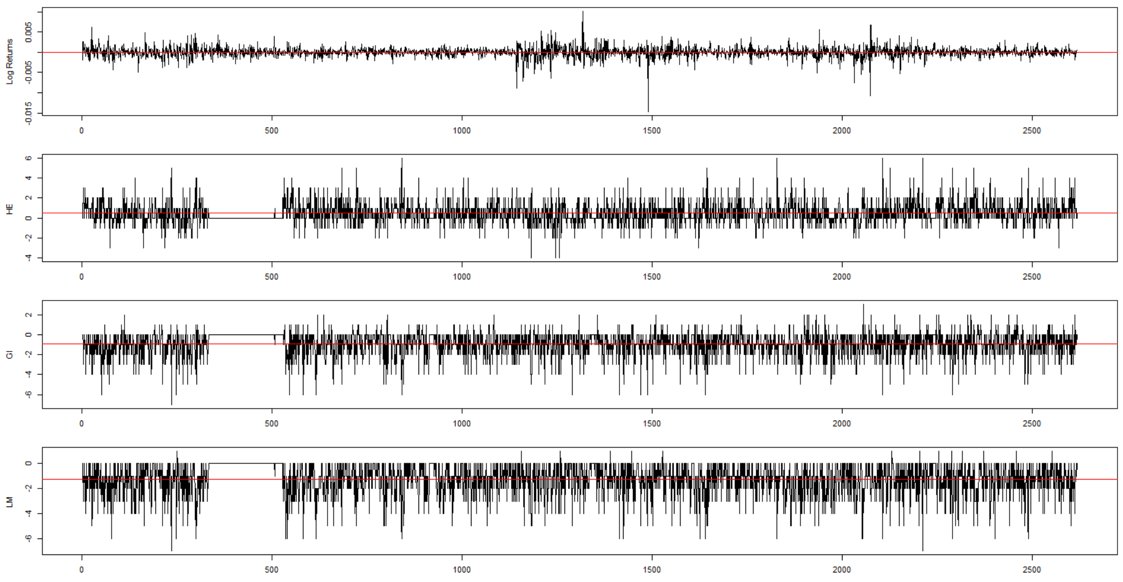


Figure 32: S&P 500 Index 20-minute log returns and dictionary-based sentiment variables.
Source: R.

The S&P 500 Index was modeled with an E-GARCH(1,1) as in Table 5 and the dictionary-based sentiment variables were introduced into its mean and variance equations.

Table 15 and Table 16 show the results for Loughran and McDonald's (LM) Dictionary-based sentiment in the mean ($mxreg1$) and variance ($vxreg1$) equations respectively.

In both cases, the coefficient is significantly different from zero at the 90% confidence level, but their sign is different from the expectation. Previous results

	Estimate	Std. Error	t-value	p-value	
mu	-1.525793e-05	2.285015e-05	-0.6677388	5.043003e-01	
mxreg1	-1.987881e-05	1.191402e-05	-1.6685224	9.521206e-02	*
omega	-3.754431e-01	9.404097e-03	-39.9233589	0.000000e+00	***
alpha1	-6.031359e-02	1.458086e-02	-4.1364903	3.526582e-05	***
beta1	9.722401e-01	6.592614e-04	1474.7414777	0.000000e+00	***
gamma1	2.073228e-01	6.171541e-03	33.5933591	0.000000e+00	***
skew	9.890137e-01	2.702973e-02	36.5898519	0.000000e+00	***
shape	4.074326e+00	3.258646e-01	12.5031261	0.000000e+00	***

Table 15: E-GARCH(1,1) model fitted on S&P 500 Index intraday log returns with 20-minute time intervals with Sentiment obtained through *LM* Dictionary as exogenous variable in the mean equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

	Estimate	Std. Error	t-value	p-value	
mu	9.562106e-06	2.194618e-05	0.4357071	0.6630492760	
omega	-4.907180e-01	2.712740e-02	-18.0893878	0.0000000000	***
alpha1	-6.369098e-02	1.661897e-02	-3.8324257	0.0001268859	***
beta1	9.624042e-01	1.797018e-03	535.5561654	0.0000000000	***
gamma1	2.475420e-01	1.035877e-02	23.8968588	0.0000000000	***
vxreg1	1.360451e-02	8.014069e-03	1.6975789	0.0895872682	*
skew	9.868604e-01	2.750760e-02	35.8759201	0.0000000000	***
shape	4.042029e+00	3.169450e-01	12.7530936	0.0000000000	***

Table 16: E-GARCH(1,1) model fitted on S&P 500 Index intraday log returns with 20-minute time intervals with Sentiment obtained through *LM* Dictionary as exogenous variable in the variance equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

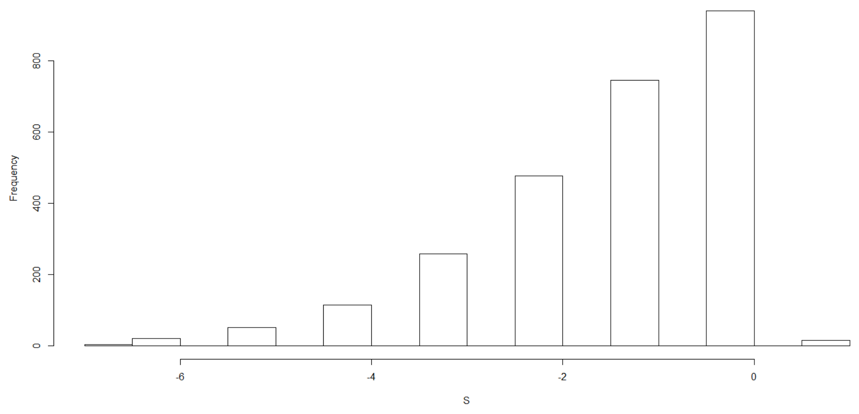
indicate a negative relationship between sentiment and volatility, explained by the fact that high sentiment is beneficial for positive market returns, which create low volatility conditions because of the asymmetric behavior of volatility. In this case, $mxreg1$ has negative sign and a possible explanation might be that, out of 2618 time intervals, only 14 contains positive articles (value 1) according to this model, and 939 intervals have value zero. Time intervals can have value zero if they contain neutral articles, if the number of positive and negative articles contained in those time windows is the same, or if they do not contain any articles. All other time intervals are associated with negative sentiment, as shown by the histogram in Figure 33 Part (a). As most of the articles are associated with negative sentiment, this sentiment variable might erroneously capture the number of articles published within time intervals, instead of the optimism or pessimism expressed by the text. When this sentiment variable is highly negative, it means a lot of articles are published during the corresponding time interval. Therefore, the negative relationship between sentiment and log returns might imply that the presence of numerous articles leads or follows market price increases in the same time window, which in turn decrease volatility, because of its asymmetric behavior.

When considering sentiment generated by Harvard IV General Inquirer Dictionary, the estimate of the coefficient in the E-GARCH(1,1) specification of log returns, $mxreg1$, is significantly different from zero at the 90% confidence level. However, the sign of the relationship, shown in Table 17, is exactly like the one provided by Loughran and McDonald's Dictionary-based sentiment and might be explained by the same reasons.

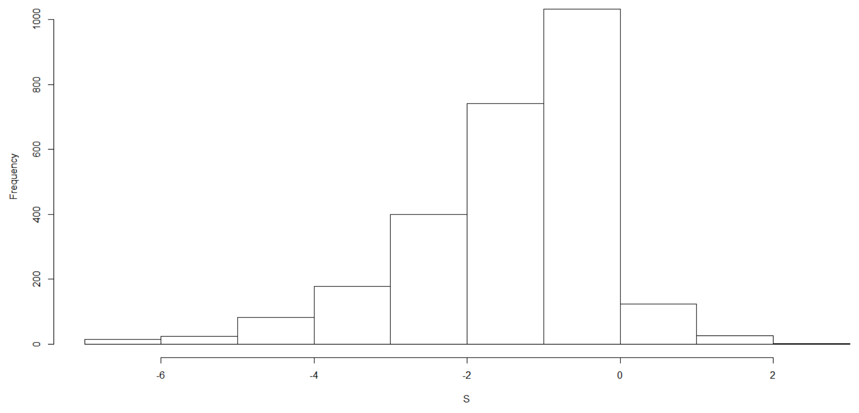
	Estimate	Std. Error	t-value	p-value	
mu	-1.425324e-05	2.499963e-05	-0.5701382	0.5685839697	
mxreg1	-2.459831e-05	1.320323e-05	-1.8630520	0.0624549495	*
omega	-3.579121e-01	8.570550e-03	-41.7606948	0.0000000000	***
alpha1	-6.102813e-02	1.441544e-02	-4.2335256	0.0000230056	***
beta1	9.735393e-01	5.919666e-04	1644.5848129	0.0000000000	***
gamma1	2.014105e-01	6.089790e-03	33.0734714	0.0000000000	***
skew	9.878571e-01	2.640676e-02	37.4092569	0.0000000000	***
shape	4.082704e+00	3.277803e-01	12.4556116	0.0000000000	***

Table 17: E-GARCH(1,1) model fitted on S&P 500 Index intraday log returns with 20-minute time intervals with Sentiment obtained through *GI* Dictionary as exogenous variable in the mean equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

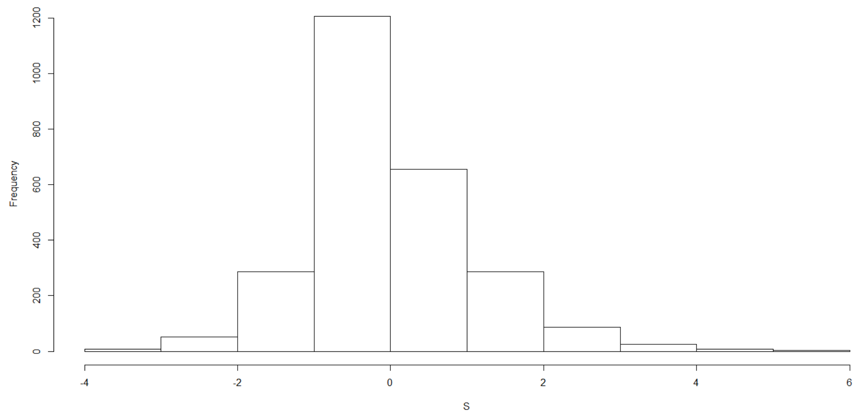
Results produced by sentiment based on Henry's Dictionary, instead, are much more intuitive and interesting. This sentiment variable was introduced in the variance equation of the E-GARCH(1,1) model applied on S&P 500 Index log returns and Table 18 shows that the corresponding coefficient $vxreg1$ is significantly different from zero at the 99% confidence level. In addition, the sign of the relationship is the expected one, implying a negative relationship between sentiment variable and



(a) *Loughran and McDonald's Dictionary.*



(b) *Harvard IV-4 General Inquirer Dictionary*



(c) *Henry's Dictionary*

Figure 33: Histogram of class distribution of dictionary-based sentiment variables. *Source: R.*

volatility. Indeed, negative sentiment represented by pessimistic news articles leads or follows increases in volatility within the same 20-minute time interval. It is not clear whether news articles published within a time interval causes price changes, or whether the opposite holds true. However, an interesting insight into this question is provided by the next analysis, which involves the use of a lagged version of this sentiment variable in the variance equation of the E-GARCH(1,1) model. Table 19 shows the results, where *vxreg*1 represents the estimate of the coefficient and it is significantly different from zero at the 99% confidence level. Again, the estimate has the expected negative sign. This result implies sentiment registered within one time window is able to explain volatility variations over the following time window.

	Estimate	Std. Error	t-value	p-value	
mu	1.251185e-05	2.216413e-05	0.5645092	5.724076e-01	
omega	-8.545816e-01	7.446377e-02	-11.4764760	0.000000e+00	***
alpha1	-5.982697e-02	1.936310e-02	-3.0897412	2.003310e-03	***
beta1	9.342788e-01	5.769767e-03	161.9265911	0.000000e+00	***
gamma1	3.050301e-01	4.102904e-02	7.4344924	1.050271e-13	***
vxreg1	-6.887654e-02	1.650614e-02	-4.1727826	3.009020e-05	***
skew	9.867904e-01	2.794138e-02	35.3164566	0.000000e+00	***
shape	4.078447e+00	3.162936e-01	12.8945000	0.000000e+00	***

Table 18: E-GARCH(1,1) model fitted on S&P 500 Index intraday log returns with 20-minute time intervals with Sentiment obtained through *HE* Dictionary as exogenous variable in the variance equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

The analysis of the explanatory power of these three dictionary-based sentiment variables is particularly interesting because the way they were built is not related to market information, but only to semantic of words in the dictionaries.

Finance-specific Henry's dictionary is the one that produced the most relevant results in this sense. However, it is also the one that the LSTM network learned with lowest accuracy on the Testing set, 0.6190 compared to 0.84571 of General Inquirer and 0.94476 of the Loughran and McDonald's Dictionary. Therefore, this result might be biased by the fact that the Machine Learning algorithm was not able to fully learn the relationship between features contained in the corpus and words in the dictionaries. To solve this issue, I used the labels originally assigned to the articles on the basis of dictionary words, as in Equation 37, and I divided all articles in 20-minute time intervals following the same procedure presented at the beginning of this paragraph. Figure 34 shows the resulting sentiment variable, which was introduced into mean and variance equation of the E-GARCH(1,1) model.

	Estimate	Std. Error	t-value	p-value	
mu	9.541357e-06	2.170781e-05	0.4395357	6.602734e-01	
omega	-6.161600e-01	1.324292e-01	-4.6527507	3.275361e-06	***
alpha1	-6.222232e-02	1.687603e-02	-3.6870235	2.268924e-04	***
beta1	9.531277e-01	1.013690e-02	94.0255892	0.000000e+00	***
gamma1	2.572402e-01	6.304220e-02	4.0804450	4.494957e-05	***
vxreg1	-3.520249e-02	1.340897e-02	-2.6252944	8.657408e-03	***
skew	9.850433e-01	2.747244e-02	35.8556965	0.000000e+00	***
shape	4.057888e+00	3.158441e-01	12.8477558	0.000000e+00	***

Table 19: E-GARCH(1,1) model fitted on S&P 500 Index intraday log returns with 20-minute time intervals with lagged Sentiment obtained through *HE* Dictionary as exogenous variable in the variance equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

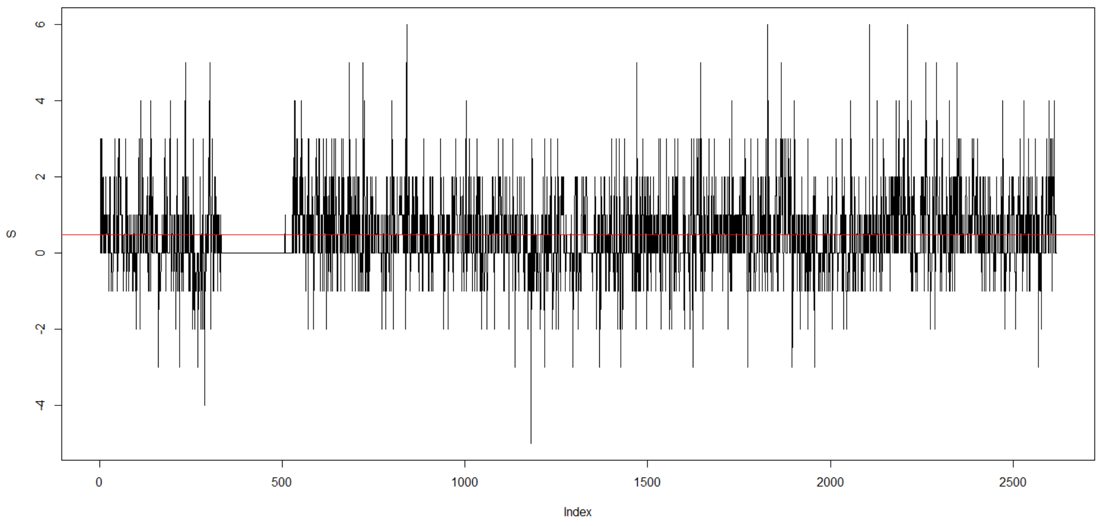


Figure 34: Henry's Dictionary-based Sentiment variable computed over 20-minute time intervals and using the original labels.
Source: R.

This analysis not only confirms previous results with respect to significance of sentiment and lagged sentiment in the variance equations, but it also reports a significant estimate of the coefficient for the sentiment introduced in the mean equation.

In both Table 20 and Table 21 the coefficient $vxreg1$, which represents sentiment and lagged sentiment respectively, is significantly different from zero at the 99% confidence level and the sign is negative, as expected. Table 22, instead, shows a significant positive relationship between S&P 500 log returns and Henry's Dictionary-based Sentiment index. Negative sentiment expressed by pessimistic news articles is correlated with negative log returns and, as a consequence, with higher volatility. In addition, the relationship between sentiment and volatility holds also when introducing a lagged version of the sentiment index, even though the effect on log returns is not significantly different from zero in this case. This analysis proves that, among the three selected dictionaries, Henry's finance-specific Dictionary is the one that better captures the relationship between words in financial news articles and market prices. In contrast to Loughran and McDonald's view [41] that positive words have poor performance in text classification because of their frequent negation, Henry's Dictionary is the one that contains the highest relative number of positive words (Figure 33) and obtained the best results in this finance-specific context.

	Estimate	Std. Error	t-value	p-value	
mu	1.492872e-05	2.250041e-05	0.6634867	5.070189e-01	
omega	-8.323309e-01	7.725308e-02	-10.7740811	0.000000e+00	***
alpha1	-5.558335e-02	2.350810e-02	-2.3644343	1.805763e-02	**
beta1	9.359287e-01	5.862907e-03	159.6356109	0.000000e+00	***
gamma1	2.951145e-01	4.628470e-02	6.3760687	1.816916e-10	***
vxreg1	-6.929969e-02	1.376328e-02	-5.0351147	4.775628e-07	***
skew	9.895577e-01	2.798567e-02	35.3594486	0.000000e+00	***
shape	4.120365e+00	3.262731e-01	12.6285747	0.000000e+00	***

Table 20: E-GARCH(1,1) model fitted on S&P 500 Index intraday log returns with 20-minute time intervals with original Sentiment obtained through HE Dictionary as exogenous variable in the variance equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

	Estimate	Std. Error	t-value	p-value	
mu	1.101497e-05	0.0000239968	0.4590184	6.462210e-01	
omega	-6.816847e-01	0.1022128222	-6.6692675	2.570832e-11	***
alpha1	-5.905524e-02	0.0187535380	-3.1490187	1.638197e-03	***
beta1	9.478881e-01	0.0078794982	120.2980309	0.000000e+00	***
gamma1	2.640949e-01	0.0696391282	3.7923355	1.492371e-04	***
vxreg1	-4.635476e-02	0.0140226639	-3.3057028	9.473852e-04	***
skew	9.859291e-01	0.0284708561	34.6294168	0.000000e+00	***
shape	4.091208e+00	0.3227999508	12.6741284	0.000000e+00	***

Table 21: E-GARCH(1,1) model fitted on S&P 500 Index intraday log returns with 20-minute time intervals with lagged original Sentiment obtained through *HE* Dictionary as exogenous variable in the variance equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

	Estimate	Std. Error	t-value	p-value	
mu	-2.667267e-06	2.300738e-05	-0.1159309	9.077073e-01	
mxreg1	1.968617e-05	5.201425e-06	3.7847644	1.538543e-04	***
omega	-3.553421e-01	8.124346e-03	-43.7379299	0.000000e+00	***
alpha1	-5.966158e-02	1.440262e-02	-4.1424124	3.436716e-05	***
beta1	9.737226e-01	5.524124e-04	1762.6733781	0.000000e+00	***
gamma1	2.024959e-01	5.990988e-03	33.8000888	0.000000e+00	***
skew	9.886369e-01	2.739394e-02	36.0896144	0.000000e+00	***
shape	4.072977e+00	3.259451e-01	12.4958996	0.000000e+00	***

Table 22: E-GARCH(1,1) model fitted on S&P 500 Index intraday log returns with 20-minute time intervals with original Sentiment obtained through *HE* Dictionary as exogenous variable in the variance equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

Conclusion

This thesis analyzes the alternative proxies used by academicians to estimate investor sentiment, which is not directly measurable. Among the various sentiment variables, those based on financial news and social media contents have become particularly interesting in recent years, due to improvements in Machine Learning and text-based sentiment analysis. The continue development in Internet of Things (IoT) and the growing interaction of humans with devices connected on the internet allows the collection of larger and larger amounts of unstructured data, which can now be processed and used for business and research purposes. The sentiment indicators based on financial news and Twitter contents that are provided by Bloomberg Terminal represent interesting examples of how qualitative information can be transformed into quantitative variables that are easier to manage for making investment decisions. An analysis of these sentiment variables is presented in the Appendix.

This thesis proves that Long Short Term Memory networks can be used to perform supervised text classification with fair performance in terms of accuracy, even if documents are not manually labeled, but are assigned a label on the basis of automatic techniques. One of the two automatic labeling techniques matches financial news articles that belong to a particular time window with the log returns of S&P 500 Index over the corresponding time window. The other technique applies the same method with VIX Index logarithmic variations. In all the cases presented in Chapter 2, accuracy computed on the Testing set was higher then random guess, indicating that the network is able to learn the features presented by the examples and apply its knowledge on new unlabeled data sets.

In addition, Chapter 3 proves that the sentiment variables obtained by this text classification improve the explanatory power of GARCH models. They explain S&P 500 Index log returns and, in some cases, they also have some predictive power when introduced as lagged variables. In a few cases the sentiment variables explain S&P 500 Index volatility, computed using GARCH models.

This thesis also analyzes an alternative text classification method, which is based on positive and negative words contained in three well known dictionaries available in the financial literature. Articles were classified as *positive* if they contains more positive than negative words, and *negative* in the opposite case. Using this classification as labels, the LSTM network was applied also on this supervised Machine Learning problem and showed high accuracy on the Testing set, being able to learn this classification technique. Three new sentiment indicators were based on this classification method and, in one case, the sentiment indicator improves the explanatory power of the E-GARCH model fitted on S&P 500 log returns, demonstrating the goodness of Henry's finance-specific dictionary. This variable is able to explain both mean and variance of S&P 500 Index log returns and it proves to have an impact on the variance even when introduced as a lagged term.

Future research might focus on the analysis of the financial news-based and Twitter-based daily sentiment Indicators provided by Bloomberg. The analysis presented in the Appendix has room for improvements. Principal Component Analysis might be a better solution to capture the common sentiment among indicators that refer to individual stocks. In addition, although this thesis focuses on the analysis of the overall market sentiment, it might be interesting to assess the predictive power of these sentiment variables with respect to individual stocks, with particular emphasis on low-volume stocks, which might be impacted by positive or negative news or tweets.

The sentiment variables produced by the LSTM network and those based on dictionaries, especially Henry's dictionary, might be used to built an intraday trading system. A trading system might be particularly interesting when considering the explanatory power of the sentiment variables expressed as lagged terms.

Of course, there is room for improvement in these sentiment variables and future research might work on perfecting the parameters used to train the LSTM models. To improve results, instead of considering fixed time intervals, articles might be labeled using Index log returns (or VIX log variations) computed over the 20-minute (or 10-minute) time interval that starts from the exact moment the article is published. In this way the effect of each article on index prices is registered over the same exact time interval. Indeed, the approach I used has the limit that articles published at the end of each time window are labeled with price movements of that time window even if they don't have the time to impact them, assuming the causality follows this direction. If, instead, the causality is inverse and articles are published on the basis of price movements that have been observed on the market, the problem concerns articles published at the beginning of the time window, because they don't have enough time to observe the pattern of that time window. Another way to try to improve the results might be using different word embedding techniques and, for instance, using pretrained word embedding to initialize the weights of the embedding layer. In addition, improvements might come from the use of different financial news sources.

With respect to dictionary-based sentiment, more complex equations than that presented in Equation 37 can be used in order to improve the results based on the three dictionaries. Even if the dictionaries used in this analysis are widely used in financial literature, other dictionaries can be used.

Bibliography

[1] Afshine Amidi and Shervine Amidi. *CS 230 — Deep Learning. Recurrent Neural Networks cheatsheet.* Stanford University. 2019. URL: `https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks`.

[2] Adam Atkins, Mahesan Niranjan, and Enrico Gerding. "Financial news predicts stock market volatility better than close price". In: *The Journal of Finance and Data Science* 4.2 (2018), pp. 120–137.

[3] Malcolm Baker and Jeremy C. Stein. "Market Liquidity as a Sentiment Indicator". In: *Journal of Financial Markets* 7.3 (2004), pp. 271–299.

[4] Malcolm Baker and Jeffrey Wurgler. "A Catering Theory of Dividends". In: *The Journal of Finance* LIX.3 (2004), pp. 1125–1165.

[5] Malcolm Baker and Jeffrey Wurgler. "The Equity Share in New Issues and Aggregate Stock Returns". In: *The Journal of Finance* LV.5 (2000), pp. 2219–2257.

[6] Malcom Baker and Jeffrey Wurgler. "Investor Sentiment and the Cross-Section of Stock Returns". In: *The Journal of Finance* LXI.4 (2006), pp. 1645–1680.

[7] Malcom Baker and Jeffrey Wurgler. "Investor Sentiment in the Stock Market". In: *Journal of Economic Perspectives* 21.2 (2007), pp. 129–152.

[8] Arindam Bandopadhyaya and Anne Leah Jones. "Measures Of Investor Sentiment: A Comparative Analysis Put-Call Ratio Vs. Volatility Index". In: *Journal of Business & Economics Research* 6.8 (2008), pp. 27–34.

[9] Nicholas Barberis and Richard Thaler. "A Survey of Behavioral Finance". In: *Handbook of the Economics of Finance.* Ed. by M. Harris G.M. Constantinides and R. Stulz. Vol. 1.B. Elsevier Science B.V., 2003, pp. 1053–1128.

[10] Tim Bollerslev. "Generalized autoregressive conditional heteroskedasticity". In: *Journal of Econometrics* 31.3 (1986), pp. 307–327.

[11] Stephen J Brown et al. *Investor Sentiment in Japanese and U.S. Daily Mutual Fund Flows.* Working Paper 9470. National Bureau of Economic Research, 2002.

[12] Wu-Jen Chuang, Liang-Yuh Ouyang, and Wen-Chen Lo. “The Impact of Investor Sentiment on Excess Returns: A Taiwan Stock Market Case”. In: *International Journal of Information and Management Sciences* 21.1 (2010), pp. 13–28.

[13] Jon Danielsson. “Univariate volatility modeling”. In: *Financial risk forecasting*. 2011. Chap. 2, pp. 31–56.

[14] Dataaspirant. *Difference between Softmax function and Sigmoid function*. 2017. URL: `https://dataaspirant.com/2017/03/07/difference-between-softmax-function-and-sigmoid-function/`.

[15] Walter Enders. “Modeling Volatility”. In: *Applied Econometric Time Series*. 2014. Chap. 3, pp. 118–180.

[16] Robert F. Engle. “Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation”. In: *Econometrica* 50.4 (1982), pp. 987–1007.

[17] Eugene Fama. “Efficient Capital Markets: A Review of Theory and Empirical Work”. In: *The Journal of Finance* 25.2 (1970), pp. 383–417.

[18] Kenneth L. Fisher and Meir Statman. “Investor Sentiment and Stock Returns”. In: *Financial Analysts Journal* 56.2 (2000), pp. 16–23.

[19] Forbes. *What’s The Difference Between Structured, Semi-Structured And Unstructured Data?* 2019. URL: `https://www.forbes.com/sites/bernardmarr/2019/10/18/whats-the-difference-between-structured-semi-structured-and-unstructured-data/`.

[20] Andrea Frazzini and Owen A. Lamont. *Dumb Money: Mutual fund flows and the cross-section of stock returns*. Working Paper 11526. National Bureau of Economic Research, 2005.

[21] Lex Fridman. *MIT Deep Learning. Recurrent Neural Networks*. Massachusetts Institute of Technology. 2017. URL: `https://deeplearning.mit.edu/`.

[22] Alexios Ghalanos. *Introduction to the rugarch package*. 2017. URL: `https://cran.r-project.org/web/packages/rugarch/vignettes/Introduction_to_the_rugarch_package.pdf`.

[23] Gyozo Gidofalvi and Charles Elkan. *Using news articles to predict stock price movements*. Department of Computer Science and Engineering, University of California, San Diego. 2001.

[24] John R. Graham and Campbell R. Jarvey. “The theory and practice of corporate finance: evidence from the field”. In: *Journal of Financial Economics* 60 (2001), pp. 187–243.

[25] Kevin Gurney. “Neural networks - an overview”. In: *An Introduction to Neural Networks*. 1997. Chap. 1, pp. 1–4.

[26] Changli He, Timo Terasvirta, and Hans Malmsten. “Moment Structure of a Family of First-Order Exponential GARCH Models”. In: *Econometric Theory* 18.4 (2002), pp. 868–885.

[27] Elaine Henry. “Are investors influenced by how earnings press releases are written?” In: *Journal of Business Communication* 45.4 (2008), pp. 363–407.

[28] Dashan Huang et al. “Investor Sentiment Aligned: A Powerful Predictor of Stock Returns”. In: *The Review of Financial Studies* 28.3 (2015), pp. 791–837.

[29] American Association of Individual Investors. *AAII Investor Sentiment Survey.* 1987. URL: `https://www.aaii.com/sentimentsurvey`.

[30] Investopedia. *Deep Learning.* 2019. URL: `https://www.investopedia.com/terms/d/deep-learning.asp`.

[31] Investopedia. *Prime Rate.* 2019. URL: `https://www.investopedia.com/terms/p/primerate.asp`.

[32] Investopedia. *Repurchase Agreement.* 2019. URL: `https://www.investopedia.com/terms/r/repurchaseagreement.asp`.

[33] Investopedia. *What Is the Investors Intelligence Sentiment Index?* 2019. URL: `https://www.investopedia.com/articles/trading/03/100103.asp`.

[34] Ian T. Jolliffe and Jorge Cadima. “Principal component analysis: a review and recent developments”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065, 20150202 (2016).

[35] Andrej Karpathy. *CS231n Convolutional Neural Networks for Visual Recognition. "Linear classification: Support Vector Machine, Softmax".* Stanford University. 2015. URL: `http://cs231n.github.io/linear-classify/#softmax`.

[36] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–44.

[37] Charles M. C. Lee, Andrei Shleifer, and Richard H. Thaler. “Investor Sentiment and the Closed-End Fund Puzzle”. In: *The Journal of Finance* 46.1 (1991), pp. 75–109.

[38] Charles M. C. Lee and Bhaskaran Swaminathan. “Price Momentum and Trading Volume”. In: *The Journal of Finance* LV.5 (2000), pp. 2017–2069.

[39] S. Lek and Y. S. Park. “Artificial Neural Networks”. In: *Encyclopedia of Ecology.* 2008, pp. 237–245.

[40] Shuming Liu. “Investor Sentiment and Stock Market Liquidity”. In: *Journal of Behavioral Finance* 16.1 (2015), pp. 51–67.

[41] Tim Loughran and Bill McDonald. “When is a Liability not a Liability? Textual Analysis, Dictionaries, and 10-Ks”. In: *The Journal of Finance* LXVI.1 (2011), pp. 35–65.

[42] Michelle Lowry. “Why does IPO volume fluctuate so much?” In: *Journal of Financial Economics* 67.1 (2003), pp. 3–40.

[43] F. A. Makinde et al. “Prediction of crude oil viscosity using feed-forward back-propagation neural network (FFBPNN)”. In: *Petroleum and Coal* 54 (2012), pp. 120–131.

[44] Yale School of Management. *Stock Market Confidence Indices*. 1989. URL: `https://som.yale.edu/faculty-research/our-centers-initiatives/international-center-finance/data/stock-market-confidence`.

[45] MathWorks. *classificationLayer*. 2019. URL: `https://it.mathworks.com/help/deeplearning/ref/classificationlayer.html`.

[46] MathWorks. *lstmLayer*. 2019. URL: `https://it.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.lstmlayer.html`.

[47] MathWorks. *softmaxLayer*. 2016. URL: `https://it.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.softmaxlayer.html#mw_a09d3c68-d062-4692-a950-9a7fea5c40c3.bvk87b1`.

[48] MathWorks. *trainingOptions*. 2019. URL: `https://it.mathworks.com/help/deeplearning/ref/trainingoptions.html`.

[49] Walaa Medhat, Ahmed Hassan, and Hoda Korashy. “Sentiment analysis algorithms and applications: A survey”. In: *Ain Shams Engineering Journal* 5.4 (2014), pp. 1093–1113.

[50] Medium. *Understanding Activation Functions in Neural Networks*. 2017. URL: `https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0`.

[51] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. International Conference on Learning Representations, 2013. URL: `http://arxiv.org/abs/1301.3781`.

[52] Arman Khadjeh Nassirtoussi et al. “Text mining for market prediction: A systematic review”. In: *Expert Systems with Applications* 41.16 (2014), pp. 7653–7670.

[53] Robert Neal and Simon M. Wheatley. “Do Measures of Investor Sentiment Predict Returns?” In: *The Journal of Financial and Quantitative Analysis* 33.4 (1998), pp. 523–547.

[54] Daniel B. Nelson. “Conditional Heteroskedasticity in Asset Returns: A New Approach”. In: *Econometrica* 59.2 (1991), pp. 347–370.

[55] Michael Nielsen. “How the backpropagation algorithm works”. In: *Neural Networks and Deep Learning*. 2015. Chap. 2, pp. 39–58.

[56] Kamal Nigam, John Lafferty, and Andrew McCallum. “Using Maximum Entropy for Text Classification”. In: *IJCAI-99 Workshop on Machine Learning for Information Filtering* 1.1 (1999), pp. 61–67.

[57] Christopher Olah. *Understanding LSTM Networks.* 2015. URL: `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[58] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. *GloVe: Global Vectors for Word Representation.* 2013. URL: `https://nlp.stanford.edu/projects/glove/`.

[59] Roper Center for Public Opinion Research. *UBS Index of Investor Optimism, 1996-2007 blog.* 2015. URL: `https://ropercenter.cornell.edu/blog/ubs-index-investor-optimism-1996-2007-blog`.

[60] Qingsong Ruan et al. “A new Investor Sentiment Indicator (ISI) Based on Artificial Intelligence: A Powerful Return Predictor in China”. In: *Economic Modelling* (2019), pp. 1–54.

[61] Lorraine Rupande, Hilary Tinotenda Muguto, and Paul-Francois Muzindutsi. “Investor sentiment and stock return volatility: Evidence from the Johannesburg Stock Exchange”. In: *Cogent Economics & Finance* 7.1, 1600233 (2019), pp. 1–16.

[62] Towards Data Science. *Stemming? Lemmatization? What?* 2018. URL: `https://towardsdatascience.com/stemming-lemmatization-what-ba782b7c0bd8`.

[63] Towards Data Science. *Understanding RNN and LSTM.* 2019. URL: `https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e`.

[64] Towards Data Science. *Why do we use word embeddings in NLP?* 2019. URL: `https://towardsdatascience.com/why-do-we-use-embeddings-in-nlp-2f20e1b632d2`.

[65] Sohar Sohangir et al. “Big Data: Deep Learning for financial sentiment analysis”. In: *Journal of Big Data* 5.3 (2018), pp. 1–25.

[66] Ava Soleimany. *MIT 6.S191 Introduction to Deep Learning. Deep Sequence Modeling with Recurrent Neural Networks.* Massachusetts Institute of Technology. 2019. URL: `http://introtodeeplearning.com/`.

[67] Wataru Souma, Irena Vodenska, and Hideaki Aoyama. “Enhanced news sentiment analysis using deep learning methods”. In: *Journal of Computational Social Science* 2.1 (2019), pp. 33–46.

[68] Hans R. Stoll and Anthony J. Curley. “Small Business and the New Issues Market for Equities”. In: *The Journal of Financial and Quantitative Analysis* 5.3 (1970), pp. 309–322.

[69] Harvard University. *General Inquirer.* URL: `http://www.wjh.harvard.edu/~inquirer/`.

[70] Gang Wang et al. “Crowds on Wall Street: Extracting Value from Social Investing Platforms”. In: *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. ACM. 2015, pp. 17–30.

[71] Wikipedia. *Gradient*. 2019. URL: `https://en.wikipedia.org/wiki/Gradient`.

[72] Wikipedia. *Gradient descent*. 2020. URL: `https://en.wikipedia.org/wiki/Gradient_descent`.

[73] Ritu Yadav, A. Vinay Kumar, and Ashwani Kumar. “News-based supervised sentiment analysis for prediction of futures buying behaviour”. In: *IIMB Management Review* 31.2 (2019), pp. 157–166.

[74] Aston Zhang et al. “Forward Propagation, Backward Propagation, and Computational Graphs”. In: *Dive into Deep Learning*. 2019. Chap. 4.7, pp. 169–172.

[75] Cathy Zhang. “Defining, Modeling, and Measuring Investor Sentiment”. Thesis. University of California, Berkeley, 2008.

[76] Martin Zweig. “An Investor Expectations Stock Price Predictive Model Using Closed-End Fund Premiums”. In: *The Journal of Finance* 28.1 (1973), pp. 67–78.

Appendix: News and Twitter Sentiment provided by Bloomberg Terminal

Before approaching Natural Language Processing of financial news, I performed a preliminary analysis to assess the existence of a relationship between this type of text sources and market prices.

Bloomberg Terminal provides some daily indicators, which summarize sentiment towards a particular stock on the basis of various news sources and Twitter social network contents. Their sentiment indicators are built following some proprietary supervised Machine Learning techniques and are released ten minutes before market opening based on articles published within the previous 24 hours.

Among the different indicators provided by Bloomberg, I highlight the ones that were used in this analysis:

- **NEWS SENTIMENT DAILY AVG:** It is an average daily news sentiment indicator towards a stock. It aggregates the sentiment of all news published during the previous 24 hours that refer to a particular stock. It takes values bounded to the interval $[-1{,}1]$.

- **NEWS POS SENTIMENT COUNT:** It represents the number of positive news during the day (24 hours).

- **NEWS NEG SENTIMENT COUNT:** It represents the number of negative news during the day (24 hours).

- **TWITTER SENTIMENT DAILY AVG:** It is the average daily Twitter sentiment indicator towards a stock. It aggregates the sentiment of all Tweets published during the previous 24 hours that refer to a particular stock. It takes values bounded to the interval $[-1{,}1]$.

- **TWITTER POS SENTIMENT COUNT:** It represents the number of positive tweets during the day (24 hours).

- **TWITTER NEG SENTIMENT COUNT:** It represents the number of negative tweets during the day (24 hours).

To analyze the explanatory power of these sentiment variables with respect to S&P 500 Index log returns and volatility, I considered Index prices from January 5, 2015 to October 3, 2019.

As sentiment variables are only available for individual public companies, I downloaded sentiment indicators for all stocks included in S&P 500 Index from Bloomberg Terminal, and, then, I combined them taking weighted average with respect to their weights within the index. The resulting variables were introduced into mean and variance equation of a E-GARCH(1,1) model applied on S&P 500 Index log returns, where the conditional distribution assumed for the model is Skew Student-t.

All variables have some predictive power because, although they are not introduced as a lagged variable, they are built on the basis of news and Twitter contents published during the previous 24 hours and released ten minutes before stock market opening.

Figures 35 and 36 show S&P 500 Index daily log returns, together with the six sentiment variables retrieved from financial News and Twitter posts respectively. Table 23, instead, shows the estimates of the coefficients for the E-GARCH(1,1) model applied on Index log returns.

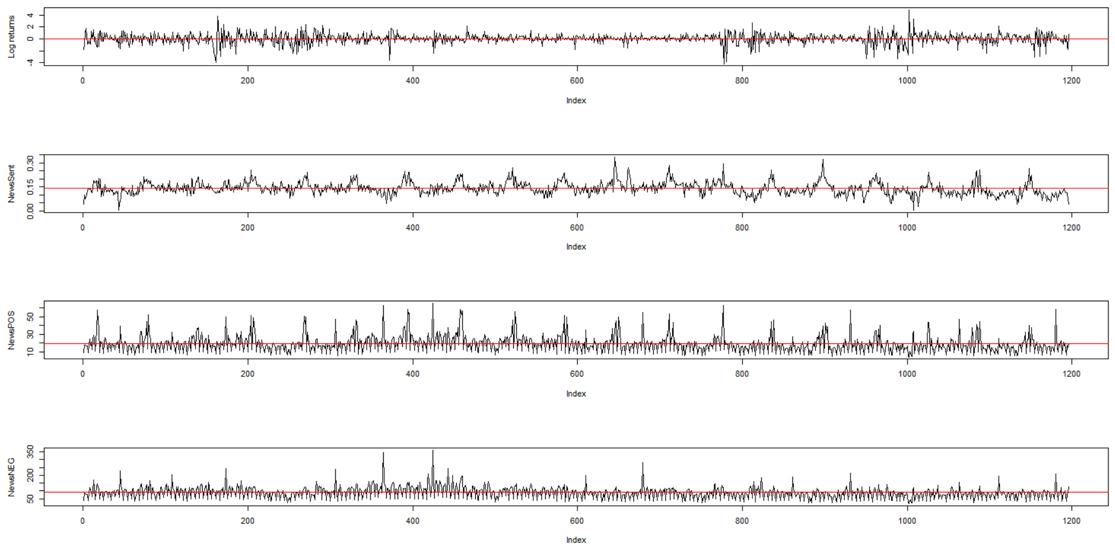


Figure 35: Daily news Sentiment about S&P 500 Index provided by Bloomberg Terminal. *Source: R*

All the following Tables represent the estimates of the coefficients for the E-GARCH(1,1) model with the addition of one of these sentiment variables in the

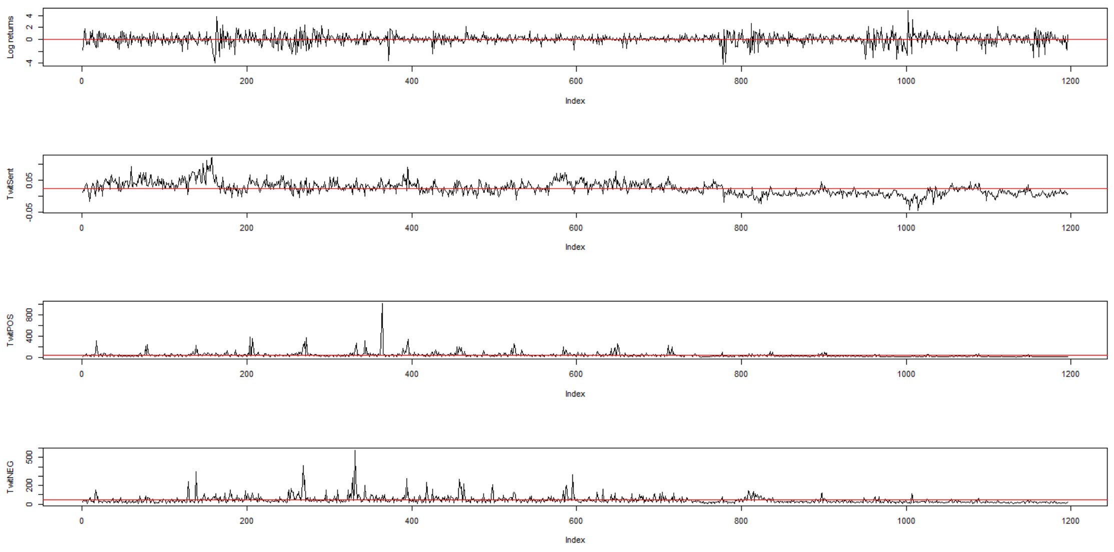


Figure 36: Daily Twitter Sentiment about S&P 500 Index provided by Bloomberg Terminal.
Source: R

	Estimate	Std. Error	t-value	p-value	
mu	0.02030164	0.01155126	1.757526	7.882825e-02	***
omega	-0.03496290	0.01062274	-3.291327	9.971578e-04	***
alpha1	-0.22483024	0.03171206	-7.089740	1.343592e-12	***
beta1	0.94552326	0.01380110	68.510728	0.000000e+00	***
gamma1	0.19266541	0.04410278	4.368555	1.250716e-05	***
skew	0.89011345	0.03258361	27.317832	0.000000e+00	***
shape	5.94559518	1.12847174	5.268714	1.373826e-07	***

Table 23: E-GARCH(1,1) model fitted on S&P 500 Index daily log returns, with Skew Student-t conditional distribution.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

mean or variance equation. In the next rows, I summarize the main results. Then I try to interpret the relationships among variables.

- In Table 24, NEWS SENTIMENT DAILY AVG was introduced into the mean equation and the estimates of all coefficients are significantly different from zero at the 99% level. The sign of $mxreg1$ is negative. A possible reason is covered later on this Appendix and it is connected to price correction following previous day positive (or negative) log returns.

- In Table 25, NEWS POS SENTIMENT COUNT was introduced into the variance equation and the estimate of the coefficient for $vxreg1$ is significantly different from zero at the 90% level. The sign of $vxreg1$ is negative, as expected. Positive sentiment is associated with low volatility levels, because of the asymmetric behavior of volatility. Omega ω, the intercept of the E-GARCH model, is not significantly different from zero, but this is not a problem because the Exponential GARCH model does not require any non-negativity constraint.
- In Table 26, NEWS NEG SENTIMENT COUNT was introduced into the mean equation and the estimates of all coefficients are significantly different from zero at the 99% level. The sign of $mxreg1$ is negative. Negative news sentiment is associated with negative log returns over the following day.
- In Table 27, NEWS NEG SENTIMENT COUNT was introduced into the variance equation and the estimate of the coefficient for $vxreg1$ is significantly different from zero at the 90% level. The sign of $vxreg1$ is negative, implying a negative relationship between number of negative news published during the previous 24 hours and volatility, which is difficult to explain. Here too, omega ω is not significantly different from zero, but this is not a problem because the Exponential GARCH model does not require any non-negativity constraint.
- In Table 28, TWITTER SENTIMENT DAILY AVG was introduced into the variance equation and the estimate of the coefficient for $vxreg1$ is significantly different from zero at the 90% level. The sign of $vxreg1$ is negative, as expected. Positive sentiment is associated with low volatility levels, because of the asymmetric behavior of volatility. Again, omega ω is not significantly different from zero, but this is not a problem because the Exponential GARCH model does not require any non-negativity constraint.
- In Table 29, TWITTER POS SENTIMENT COUNT was introduced into the mean equation and the estimate of the coefficient for $mxreg1$ is significantly different from zero at the 95% level. The sign of $mxreg1$ is negative. This result is difficult to explain, as positive sentiment should be associated with positive returns.
- In Table 30, TWITTER NEG SENTIMENT COUNT was introduced into the mean equation and the estimate of the coefficient for $mxreg1$ is significantly different from zero at the 90% level. The sign of $mxreg1$ is negative. Negative Twitter sentiment is associated with negative log returns on the following day.

As some of these estimates, obtained introducing sentiment variables into mean and variance equations of an E-GARCH(1,1) model applied on S&P 500 Index log returns, have ambiguous signs, I examined further the relationship between log returns and sentiment. In some cases, a reasonable explanation was found, while, in others, the assumption made at the beginning of this Appendix, where S&P 500 Index sentiment was defined as the weighted average of individual stocks sentiment, might have played an important role.

These news and Twitter based sentiment variables can be thought of as lagged variables, because they are released 10 minutes before market opening and are based on news published during the previous 24 hours. Although some of these articles are published when markets are closed, others are published during the previous market session and, thus, they might have an impact on previous day market prices. Therefore, I analyzed also the effect of these variables on lagged log returns. In this way I tried to assess the impact of Sentiment released by Bloomberg at time $t+1$, but representing news published during the previous 24 hours, on log returns at time t. Results are shown in Tables 31 to 36.

- In Table 31, NEWS SENTIMENT DAILY AVG was introduced into the mean equation and the estimates of all coefficients are significantly different from zero at the 99% level. The coefficient for $mxreg1$ is 0.88590361 and this implies that news based positive sentiment leads or follows positive returns. The causality of this relation is uncertain also because the indicator refers both to articles published during open markets and those published after markets close. The sign of this coefficient is in contrast to that of Table 24. In this case, the negative sign of $mxreg1$ can represent a smaller correction (-0.41102860) in market prices that follows the excessive optimism of the previous day.

- In Table 32, NEWS POS SENTIMENT COUNT was introduced into the mean equation and the estimates of all coefficients are significantly different from zero at the 99% level. However, it is not clear why the coefficient for $mxreg1$ is negative. One reason might be the assumption made in order to define Index sentiment based on sentiment of individual stocks.

- In Table 33, NEWS NEG SENTIMENT COUNT was introduced into the mean equation and the estimates of all coefficients are significantly different from zero at the 99% level. The coefficient for $mxreg1$ is negative, as expected, implying a negative relationship between number of negative news and log returns. Again, the direction of this relationship is not clear. Negative news lead or follow negative returns and, the day after, when the indicator is published, prices continue to decrease, even if magnitude is lower (Table 26).

- In Table 34, TWITTER SENTIMENT DAILY COUNT was introduced into the mean equation and the estimates of all coefficients are significantly different from zero at the 99% level. The coefficient for $mxreg1$ is positive, as expected, and the magnitude is quite important (1.48694395). In this case, I interpret the positive relationship between Index log returns and Twitter based sentiment as a result of Tweets being published after stock price variations, instead of prices being influenced by Tweets. However, this issue should be examined further. No significance was registered for the following day, when the sentiment indicator is released by Bloomberg, meaning Tweets of the previous day do not have an impact on Index log returns.

- In Table 35, TWITTER POS SENTIMENT COUNT was introduced into the mean equation and the estimates of all coefficients are significantly different from zero at the 99% level. However, it is not clear why the coefficient for $mxreg1$ is negative and the reason might be again connected to the original assumptions on S&P 500 Index sentiment indicators.

- In Table 36, TWITTER NEG SENTIMENT COUNT was introduced into the mean equation and the estimates of all coefficients are significantly different from zero at the 99% level. The coefficient for $mxreg1$ is -0.001217076, which is negative, as expected, and implies a negative relationship between number of negative news and log returns. Here too, the direction of this relationship is not clear, but a reasonable answer might be that negative Tweets are published following quite relevant drops in stock prices. The day after, prices keep decreasing but magnitude is lower, -0.0006611851 (Table 30).

Of course, these results are influenced by the assumption I made at the beginning of this Appendix, when I defined S&P 500 Index sentiment as the weighted average of individual stocks sentiment. Using better techniques to aggregate sentiment related to individual stocks, such as Principal Analysis Component, might be beneficial to this analysis. In addition, it is also interesting to examine the impact on individual stocks of the sentiment variables provided by Bloomberg Terminal. In particular, low-volume stocks might register an interesting impact on volatility when positive or negative news are published.

	Estimate	Std. Error	t-value	p-value	
mu	0.08099054	0.01559431	5.193597	2.062694e-07	***
mxreg1	-0.41102860	0.07434946	-5.528334	3.232868e-08	***
omega	-0.03672300	0.01158073	-3.171043	1.518929e-03	***
alpha1	-0.22670531	0.03354587	-6.758070	1.398415e-11	***
beta1	0.94359643	0.01458649	64.689753	0.000000e+00	***
gamma1	0.19389934	0.04490938	4.317569	1.577571e-05	***
skew	0.89166585	0.03514590	25.370406	0.000000e+00	***
shape	5.86143125	1.09946693	5.331157	9.758878e-08	***

Table 24: E-GARCH(1,1) model fitted on S&P 500 Index daily log returns, with NEWS SENTIMENT DAILY AVG as exogenous variable in the mean equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

	Estimate	Std. Error	t-value	p-value	
mu	0.018995137	0.006764732	2.8079661	4.985548e-03	***
omega	0.020248448	0.032495496	0.6231155	5.332086e-01	
alpha1	-0.226370514	0.030953348	-7.3132803	2.606804e-13	***
beta1	0.944137067	0.013873992	68.0508569	0.000000e+00	***
gamma1	0.181506351	0.044357203	4.0919250	4.278069e-05	***
vxreg1	-0.002928524	0.001762452	-1.6616189	9.658922e-02	*
skew	0.890056429	0.032050704	27.7702618	0.000000e+00	***
shape	5.822409683	1.078786161	5.3971861	6.769416e-08	***

Table 25: E-GARCH(1,1) model fitted on S&P 500 Index daily log returns, with NEWS POS SENTIMENT COUNT as exogenous variable in the variance equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

	Estimate	Std. Error	t-value	p-value	
mu	0.0528920202	0.0204568727	2.585538	9.722718e-03	***
mxreg1	-0.0003484905	0.0001307312	-2.665702	7.682786e-03	***
omega	-0.0346652487	0.0121662671	-2.849292	4.381663e-03	***
alpha1	-0.2245075473	0.0328599539	-6.832254	8.359091e-12	***
beta1	0.9466783617	0.0142381349	66.488931	0.000000e+00	***
gamma1	0.1879103646	0.0447895308	4.195408	2.723806e-05	***
skew	0.8906283810	0.0369941848	24.074821	0.000000e+00	***
shape	5.8420371202	1.1186923935	5.222202	1.768084e-07	***

Table 26: E-GARCH(1,1) model fitted on S&P 500 Index daily log returns, with NEWS NEG SENTIMENT COUNT as exogenous variable in the mean equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

	Estimate	Std. Error	t-value	p-value	
mu	0.0191557336	0.012256191	1.562943	1.180659e-01	
omega	0.0472815645	0.041175585	1.148291	2.508483e-01	
alpha1	-0.2244970978	0.030163880	-7.442580	9.880985e-14	***
beta1	0.9469326658	0.013497179	70.157823	0.000000e+00	***
gamma1	0.1783175499	0.042941095	4.152608	3.287070e-05	***
vxreg1	-0.0009021238	0.000481875	-1.872112	6.119115e-02	*
skew	0.8892299906	0.033596167	26.468198	0.000000e+00	***
shape	5.7262212314	1.038179002	5.515640	3.475137e-08	***

Table 27: E-GARCH(1,1) model fitted on S&P 500 Index daily log returns, with NEWS NEG SENTIMENT COUNT as exogenous variable in the variance equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

	Estimate	Std. Error	t-value	p-value	
mu	0.01768295	0.02289263	0.7724297	4.398600e-01	
omega	-0.01712149	0.01424370	-1.2020394	2.293483e-01	
alpha1	-0.24063645	0.03477642	-6.9195299	4.531486e-12	***
beta1	0.93722715	0.01510456	62.0492951	0.000000e+00	***
gamma1	0.18729782	0.04555513	4.1114538	3.931755e-05	***
vxreg1	-0.97822029	0.53469956	-1.8294765	6.732825e-02	*
skew	0.88512411	0.03716292	23.8174008	0.000000e+00	***
shape	5.91181773	1.13259330	5.2197181	1.791957e-07	***

Table 28: E-GARCH(1,1) model fitted on S&P 500 Index daily log returns, with TWITTER SENTIMENT DAILY AVG as exogenous variable in the variance equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

	Estimate	Std. Error	t-value	p-value	
mu	0.0469481653	0.0192445540	2.439556	1.470533e-02	**
mxreg1	-0.0005924614	0.0002658582	-2.228486	2.584811e-02	**
omega	-0.0354687544	0.0114032885	-3.110397	1.868360e-03	***
alpha1	-0.2287057792	0.0328486064	-6.962420	3.344880e-12	***
beta1	0.9457045337	0.0143219488	66.031833	0.000000e+00	***
gamma1	0.1838661185	0.0440932886	4.169934	3.046874e-05	***
skew	0.8890999653	0.0360774124	24.644228	0.000000e+00	***
shape	5.8152657911	1.0860257003	5.354630	8.573167e-08	***

Table 29: E-GARCH(1,1) model fitted on S&P 500 Index daily log returns, with TWITTER POS SENTIMENT COUNT as exogenous variable in the mean equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

	Estimate	Std. Error	t-value	p-value	
mu	0.0490358534	0.0236313213	2.075036	3.798316e-02	**
mxreg1	-0.0006611851	0.0003402361	-1.943312	5.197844e-02	*
omega	-0.0351339216	0.0113645314	-3.091542	1.991201e-03	***
alpha1	-0.2296788906	0.0326779075	-7.028568	2.086553e-12	***
beta1	0.9456482026	0.0143243449	66.016855	0.000000e+00	***
gamma1	0.1805632245	0.0441594313	4.088894	4.334352e-05	***
skew	0.8884400942	0.0348441009	25.497575	0.000000e+00	***
shape	5.8759293453	1.1044970503	5.320005	1.037647e-07	***

Table 30: E-GARCH(1,1) model fitted on S&P 500 Index daily log returns, with TWITTER NEG SENTIMENT COUNT as exogenous variable in the mean equation. *Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level. Source: R, Rugarch package.*

	Estimate	Std. Error	t-value	p-value	
mu	-0.10532060	0.009831559	-10.712503	0.000000e+00	***
mxreg1	0.88590361	0.089296951	9.920872	0.000000e+00	***
omega	-0.03260217	0.011166015	-2.919768	3.502920e-03	***
alpha1	-0.21603750	0.032118271	-6.726312	1.740186e-11	***
beta1	0.94953064	0.014055177	67.557359	0.000000e+00	***
gamma1	0.18994354	0.043300396	4.386647	1.151116e-05	***
skew	0.89887611	0.035476811	25.337004	0.000000e+00	***
shape	5.88932600	1.158080462	5.085420	3.668127e-07	***

Table 31: E-GARCH(1,1) model fitted on lagged S&P 500 Index daily log returns, with NEWS SENTIMENT DAILY AVG as exogenous variable in the mean equation. *Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level. Source: R, Rugarch package.*

	Estimate	Std. Error	t-value	p-value	
mu	0.059739540	0.0072824250	8.203248	2.220446e-16	***
mxreg1	-0.002040942	0.0001674266	-12.190069	0.000000e+00	***
omega	-0.034783350	0.0104486078	-3.328994	8.716035e-04	***
alpha1	-0.228196972	0.0316093971	-7.219276	5.226930e-13	***
beta1	0.945519906	0.0138773398	68.134089	0.000000e+00	***
gamma1	0.188610553	0.0446204054	4.227002	2.368253e-05	***
skew	0.887036064	0.0329295661	26.937375	0.000000e+00	***
shape	5.828202229	1.0743486723	5.424870	5.799663e-08	***

Table 32: E-GARCH(1,1) model fitted on lagged S&P 500 Index daily log returns, with NEWS POS SENTIMENT COUNT as exogenous variable in the mean equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

	Estimate	Std. Error	t-value	p-value	
mu	0.1057421719	0.0279274611	3.786315	1.528977e-04	***
mxreg1	-0.0009472503	0.0002688297	-3.523608	4.257136e-04	***
omega	-0.0337155019	0.0110940925	-3.039050	2.373255e-03	***
alpha1	-0.2258707090	0.0308663777	-7.317694	2.522427e-13	***
beta1	0.9475782491	0.0138232170	68.549763	0.000000e+00	***
gamma1	0.1838835689	0.0454212875	4.048401	5.156882e-05	***
skew	0.8880117224	0.0365318350	24.307887	0.000000e+00	***
shape	5.7309444172	1.0480419686	5.468239	4.545276e-08	***

Table 33: E-GARCH(1,1) model fitted on lagged S&P 500 Index daily log returns, with NEWS NEG SENTIMENT COUNT as exogenous variable in the mean equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

	Estimate	Std. Error	t-value	p-value	
mu	-0.01587072	0.01772514	-0.8953793	3.705844e-01	***
mxreg1	1.48694395	0.63990109	2.3237090	2.014109e-02	***
omega	-0.03328891	0.01111803	-2.9941380	2.752215e-03	***
alpha1	-0.21391129	0.03489587	-6.1299892	8.788505e-10	***
beta1	0.94819360	0.01399127	67.7703731	0.000000e+00	***
gamma1	0.20142105	0.04140829	4.8642683	1.148810e-06	***
skew	0.89534647	0.03472181	25.7862821	0.000000e+00	***
shape	5.97845737	1.17076978	5.1064330	3.282970e-07	***

Table 34: E-GARCH(1,1) model fitted on lagged S&P 500 Index daily log returns, with TWITTER SENTIMENT DAILY AVG as exogenous variable in the mean equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

	Estimate	Std. Error	t-value	p-value	
mu	0.0482558764	0.0170420604	2.831575	4.631933e-03	***
mxreg1	-0.0006373508	0.0002602163	-2.449312	1.431295e-02	***
omega	-0.0357235888	0.0114393393	-3.122872	1.790958e-03	***
alpha1	-0.2304154966	0.0328350982	-7.017354	2.261080e-12	***
beta1	0.9448430791	0.0147641429	63.995796	0.000000e+00	***
gamma1	0.1820918201	0.0441608624	4.123376	3.733602e-05	***
skew	0.8886572173	0.0358570972	24.783301	0.000000e+00	***
shape	5.7651943693	1.0809613518	5.333395	9.639307e-08	***

Table 35: E-GARCH(1,1) model fitted on lagged S&P 500 Index daily log returns, with TWITTER POS SENTIMENT COUNT as exogenous variable in the mean equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.

	Estimate	Std. Error	t-value	p-value	
mu	0.074882337	0.0431915226	1.733728	8.296642e-02	***
mxreg1	-0.001217076	0.0005416133	-2.247132	2.463161e-02	***
omega	-0.035645822	0.0126531835	-2.817143	4.845301e-03	***
alpha1	-0.230975792	0.0344500504	-6.704658	2.018785e-11	***
beta1	0.946448743	0.0154795787	61.141764	0.000000e+00	***
gamma1	0.174642860	0.0438347675	3.984117	6.773138e-05	***
skew	0.885748150	0.0374231899	23.668430	0.000000e+00	***
shape	5.859110722	1.1059618579	5.297751	1.172376e-07	***

Table 36: E-GARCH(1,1) model fitted on lagged S&P 500 Index daily log returns, with TWITTER NEG SENTIMENT COUNT as exogenous variable in the mean equation.
Statistical significance at the 99% (***), 95% (***), 90% (***) confidence level.
Source: R, Rugarch package.