ZHAW Zurich University of Applied Sciences

School of Management and Law

Master of Science in Banking and Finance

Master Thesis

Using Financial News for Stock Price Direction Prediction: An Empirical Investigation

---

Submitted by:

Pedro Harder

[redacted]

Supervisor:

Dr. Bledar Fazlija

Filed on:

Winterthur, 1 July 2021

I. Management Summary

The semi-strong form of financial market efficiency states that asset prices reflect all publicly available information. Consequently, natural language processing methods can be used to extract the market sentiment from the information such as the news. However, traditional natural language processing methods have the disadvantage that some information such as the context of words or the structure of sentences get lost.

The purpose of this master thesis is to extract the sentiment of the financial markets from news articles and to use the estimated sentiment scores to predict the price direction of the stock market index Standard & Poor's 500. To overcome the drawbacks of traditional natural language methods, state-of-the-art natural language processing models based on the Transformer architecture are used in this master thesis.

To enable the best possible classification performance of sentiment scores, state-of-the-art bidirectional encoder representations from transformers (BERT) models are used. The pretrained transformer networks are fine-tuned on a labeled financial dataset to be able to estimate the sentiment of the financial markets. After fine-tuning the models, they are applied to news articles from Bloomberg and Reuters to predict the sentiment score of the news. To forecast the price direction of the stock market index, the predicted sentiment scores are fed into a machine learning model. Thereby, the sentiment scores of the titles, the content, and their sentiment scores combined with past time series information of the stock market index are used as input.

The results indicate that the use of sentiment scores generated from news content can be used for stock price direction prediction. The use of sentiment scores extracted from the titles or the combination of sentiment scores from the titles and the content does not improve the quality of the prediction.

Based on the findings of this master thesis, it can be concluded that the sentiment scores can be used for the prediction of the stock price direction. For further research in this area, the author of this master thesis recommends using recurrent deep learning models. Due to their internal state, these deep learning models have a memory that can be useful for predicting stock price directions. Practical recommendations are that the sentiment scores

can be used in a risk-based approach as a complement to the calculation of the value at risk or the expected shortfall.

II. Table of Contents

III. List of Figures

IV. List of Tables

1 Introduction

The assumption that financial markets are random and cannot be predicted has been investigated in numerous research studies, whereby different results have been obtained (Nassirtoussi, Aghabozorgi, Wah & Ngo, 2014, pp. 7,653–7,670). According to Fama (1970, p. 414), the efficiency of a financial market can be divided into three categories: strong, semi-strong, and weak. In the weak form of efficiency, asset prices reflect past market prices. In the semi-strong form of efficiency, asset prices reflect all publicly available information; and in the strong form of efficiency, asset prices reflect all information, including non-public information (Posth, 2019). Therefore, it can be deduced that the strength of market efficiency correlates with the available information. Depending on the market efficiency, certain information about asset prices should be reflected in the daily news about the financial markets.

This master thesis conducts an empirical investigation using financial textual data in an attempt to predict the stock market index price direction of the Standard & Poor's 500 Index (S&P 500®). To the best knowledge of the author of this master thesis, some previous research has been undertaken in this area, with most of it using common natural language processing models such as bag-of-words or other models augmented with a financial lexicon. Accordingly, in the natural language processing part of this master thesis, state-of-the-art transformer models are used, whose theory (Vaswani et al., 2017) was first published in a 2017 research study by Google engineers and made available as open-source software in 2019 (Devin, Chang, Lee, & Toutanova, 2019, pp. 4,171–4,186). The pre-trained, state-of-the-art transformer models should be able to detect the interdisciplinary between behavioral economics, machine learning, and linguistics by fine-tuning them on financial textual data.

After applying the state-of-the-art natural language processing models to the financial data and thus predicting the sentiment score of the news, the output is fed into a machine learning model. Thereby, the sentiment scores of the titles, the content, and their sentiment scores combined with past time series information of the S&P 500 stock market index are used to predict the price direction of the stock market index.

1.1 Motivation

The aim of this master thesis is to apply state-of-the-art natural language processing methods for predicting the stock price direction. This paper includes a theoretical introduction to the topics of machine learning, natural language processing, the state-of-the-art transformer networks as well as their practical implementation through the programming language Python.

1.2 Problem Statement and Research Question

The Internet has significantly increased the amount of available data in recent years. The financial sector is also affected by this technological transformation. With the advancement of research in the areas of machine learning, new opportunities have opened up on generating business-driving information from the available data. Based on this, the following research question is answered in this master thesis:

"Is it possible to predict stock price direction movements using state-of-the-art natural language processing techniques in combination with machine learning methods?"

1.3 Scope

Since the topic of natural language processing is extensive, it is not possible to include all aspects of it in this thesis. Therefore, the following points define the scope of this work:

- The focus of this paper is on the practical implementation of natural language processing applied to financial news datasets to predict the stock price direction of the stock market index S&P 500.
- The algorithms used in the computer implementation are part of open-source Python libraries. This implies that the used algorithms are not developed by the researcher.
- The theoretical background of this master thesis is exposed in a way that it is understandable by students of the master's degree in banking and finance at the Zurich University of Applied Sciences, with specialization in capital markets and data science. For more in-depth explanations, the referenced literature should be consulted.
- The datasets used, as well the implementation of the machine learning models for the empirical part of this master thesis, are limited by the resources available to the researcher.

1.4 Structure of the Research Project

This master thesis is divided into six chapters. After the introductory chapter, the second presents an overview of the theorical background, which is necessary for understanding the rest of the paper. The datasets are presented and described in the third chapter. The fourth chapter presents the empirical models and explains the applied methodology. The fifth chapter discusses the results and states the answer to the research question. This chapter also presents the limitations of this work, formulates recommendations for similar research, and gives an outlook on possible developments. The sixth chapter contains a list of references.

2 Theoretical Fundamentals

After a brief introduction to the master thesis, this chapter presents the notations and terminology used. These are necessary to understand the concepts implemented in the empirical part of this thesis. However, it is assumed that the reader of this paper already has a more in-depth knowledge in the concepts of machine learning, deep learning, and the financial markets. The reader requiring more detailed information is recommended to consult the referenced literature. Each of the subchapters presents definitions, explanations, and an overview of the practical implications of a topic or multiple subtopics of relevance for this study. The first subchapter is a brief introduction to machine learning and concepts closely related to it such as random forest classification. The second subchapter explains the main ideas of natural language processing. The third subchapter deals with the concepts of deep learning and highlights the advantages of transformers compared to traditional models. The fourth subchapter discusses the software used by the author for the empirical part of this master thesis.

2.1 Machine Learning

"Machine learning is the science or art of programming computers to learn from data" (Géron, 2017, p. 4). This topic can be divided into three subtopics: supervised learning, unsupervised learning, and reinforcement learning.

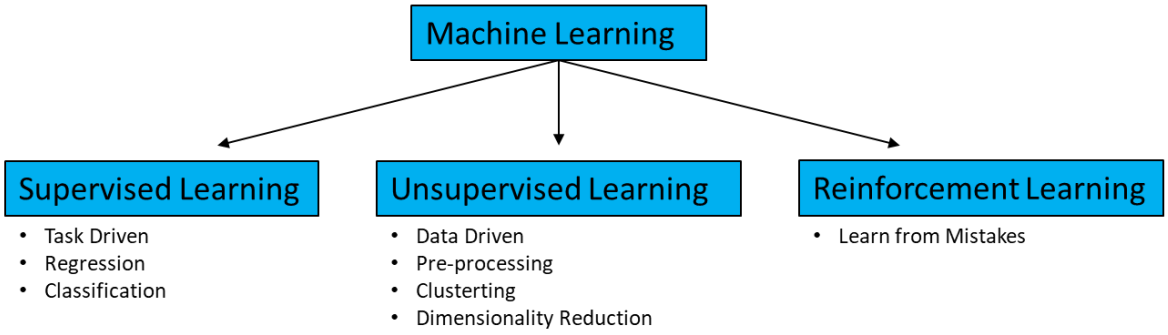

Figure 1: Types of machine learning – own illustration based on Bhatt (2018)

The figure above shows that supervised learning follows a task driven approach and can be further divided into regression or classification problems. In solving such problems, the developer has countless methods at his disposal. In this work, the stock price direction is predicted, which categorizes the problem to be solved into a classification problem. One of the methods used in this master thesis is the random forest algorithm, which is one

of the most widely used models of machine learning and is also one of the workhorses in research as well as in industry (Fazlija, 2020). In contrast, unsupervised learning follows the data driven approach. An essential difference to supervised learning is that no target variable is used during model fitting. A distinction is made between methods such as preprocessing, clustering, or dimensionality reduction. The third subcategory of machine learning is reinforcement learning, where the applied model attempts to learn from the failures and thus improves itself continuously.

2.1.1 Random Forest

The random forest algorithm can be applied to classification and regression problems and is a component of ensemble learning. The idea behind ensemble learning is that there is no single model that works best for all machine learning problems. Depending on the nature of the problem, different models show different strengths and weaknesses (Fazlija, 2020). The main idea of ensemble learning is to create a strong predictive model based on a collection of several simpler models. There are different methods such as bagging, boosting, stacking, or random forest. With ensemble learning, it is possible to counteract the classical machine learning problems such as poor prediction accuracy, high bias, overfitting, and underfitting (Hastie, Tibshirani & Friedman, 2009, pp. 605–507). Figure 2 below explains the structure of a random forest model.

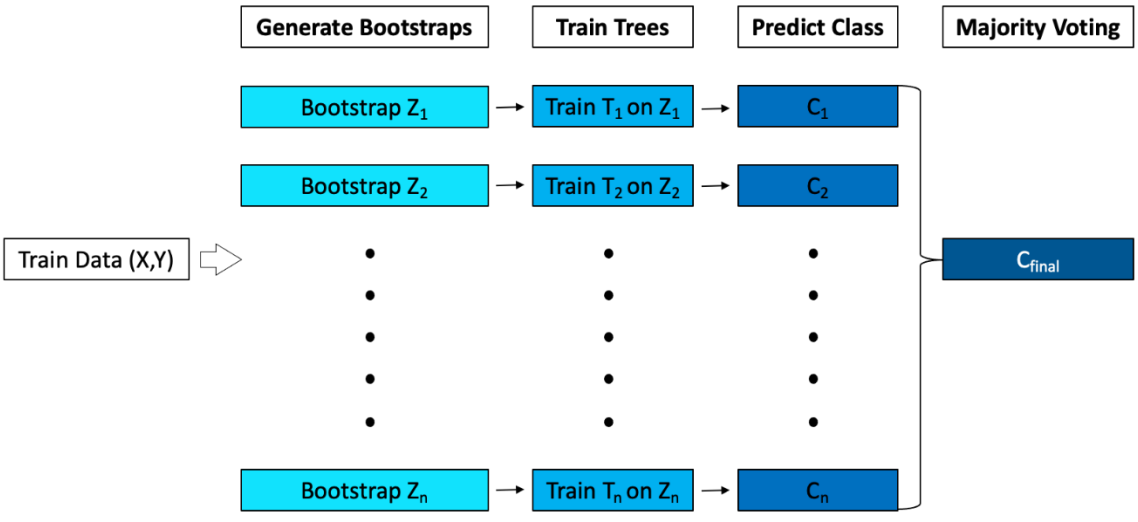

Figure 2: Random forest procedure – own illustration based on Fazlija (2020)

Before explaining the above figure, the variables shown in the illustration are explained below:

- (X,Y) stands for the explanatory respectively target variable in the train dataset

- Z_n stands for a bootstrap sample produced from the training data
- T_n stands for the machine learning model decision tree
- C_n stands for the predicted class
- C_{final} stands for the resulting predicted class of the random forest model

The illustration shows that, in a first step, different bootstrap samples are generated from the training data. In a second step, decision trees are trained on each of these bootstrap samples, which in a third step predict a class for each observation. Since the model now has an arbitrary number of trees, some of them making different predictions, they are aggregated to one prediction in the last step. As mentioned in the introduction of this subsection, the idea of ensemble learning is to combine a collection of models to get a stronger predictor. However, until now the described method is bagging, which is also categorized to ensemble learning, with decision trees.

The random forest algorithm, in contrast, has the same structure with only one significant difference: For each split in the learning process, only a few features are selected randomly. Consequently, the random forest model builds a collection of decorrelated decision trees and then averages them in the last step (majority voting). This is also the reason why the algorithm was named this way. To provide a better understanding of the model, the methodology is shown below (Hastie, Tibshirani & Friedman, 2009, pp. 587–588):

1. For b = 1 to B:
    a. Draw a bootstrap Sample Z of size N from the training data.
    b. Grow a random forest tree T_b to the bootstrapped data by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached:
        i. Select m variables (features) at random from the p variables (features).
        ii. Pick the best variable/split-point among the m.
        iii. Split the node into two daughter nodes.
2. Output the ensemble trees $\{T_b\}_{b=1}^{B}$.

As mentioned previously, a new prediction at point x would be the majority vote of all B votes of each tree. Therefore, let $C_b(x)$ be the class prediction of the bth random forest tree (Hastie, Tibshirani & Friedman, 2009, p. 588):

$$C_{randomforest}(x) = majority\ vote\ \{C_b(x)\}_{b=1}^{B}$$

As the random forest algorithm can also be used for regressions, its formula for new predictions is listed for the sake of completeness:

$$F_{randomforest}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$$

Since decision trees are quite noisy, they benefit from the averaging. Furthermore, the trees are identically distributed, which means that the expectation of an average of B such trees is the same as the expectation of any of them. This means that the bias of the bagged trees is the same as the bias of any single tree. An improvement due to the large number of different trees is that the variance of the entire model should be reduced. The formula for the variance of the average value of different B trees is shown below:

$$\sigma^2{}_{Randomforest} = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

where:

- ρ is the positive pairwise correlation between the trees;
- σ^2 is the variance of the trees; and
- B is the number of trees.

Considering the formula above, it becomes clear that the variance $\sigma^2{}_{Randomforest}$ of model becomes smaller when the number of bootstrap samples B increases. Therefore, the right term in the equation disappears when B is increased. Since the random forest model selects the features randomly (see methodology above), it creates decorrelated trees with a smaller correlation ρ. Therefore, the main idea of random forest is to further optimize or reduce the variance reduction achieved by bagging with decision trees without increasing the variance between the trees too much (Hastie, Tibshirani & Friedman, 2009, pp. 587–588).

For classification problems, the default parameter for m (number of randomly selected features) is often $\sqrt{p}$ (with p for the number of total available features). For regression tasks, it is often $\sqrt{p/3}$. According to Hastie, Tibshirani & Friedman (2009, p. 592), this parameter should be set as a tuning parameter with a minimum value of 1. They also

indicate that a high number bootstrap samples B do not overfit the random forest algorithm.

2.1.2 The Bias Variance Trade-Off

In order to check the quality of a machine learning model, the dataset is split into a training and a test sample. Subsequently, the training data is given as input to the algorithm so that it can learn from the data. In the learning process of the model, the parameters are modified and adjusted with the goal of optimizing a score such as the accuracy, precision, or recall. After the parameters have been optimally determined, the test data is given to the model as input. To evaluate the out-of-sample performance of the algorithm, the resulting output is compared to the correct output. After the evaluation, the behavior of the model can be divided into three categories (Nguyen & Zeigermann, 2018, pp. 14–20):

1. **Underfitting (high bias, low variance)**
   The model is not able to learn the patterns in the training data well enough. Therefore, regardless of whether the model performs in-sample or out-of-sample, its outputs are rather static. Such a model is characterized by a high bias and a low variance (Nguyen & Zeigermann, 2018, pp. 14–20).

2. **Overfitting (low bias, high variance)**
   The results of the training runs show a low error rate, but this is only true for the in-sample data. The error rate for the out-of-sample data is correspondingly higher. The model can be described as over-adapted since it was fitted excessively well on the training data and performs poorly on unseen data. Small changes in the inputs of an overfitted model are sufficient to generate large changes in the outputs. Such a model has a low bias and a high variance (Nguyen & Zeigermann, 2018, pp. 14–20).

3. **Sweet spot (low bias, low variance)**
   Models of this category show a low error rate in sample as well as out-of-sample data. The performance of such models is similar in both cases. Consequently, they have a low bias and a low variance and represent models which generate an ideal output (Nguyen & Zeigermann, 2018, pp. 14–20).

Figure 3 below provides an overview of the bias variance trade-off. The black dots correspond to the outputs of the training phase and the orange dots to the outputs of the test phase.

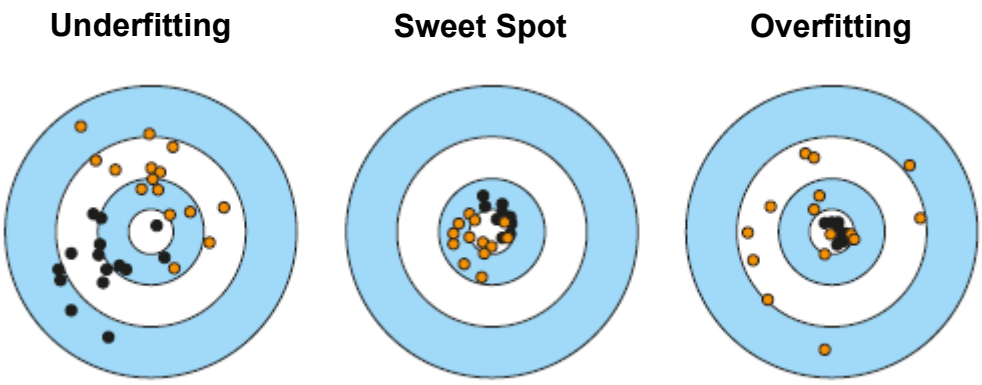

Figure 3: Illustration of the bias-variance trade-off (Nguyen & Zeigermann, 2018, p. 19)

2.1.3 Cross-Validation

To incorporate the bias variance trade-off described in the previous chapter into the training process of the models, the methodology of cross-validation can be used. This means that, in the learning process, the entire training dataset is split into numerous training as well as validation samples. The model is then fitted on the training data and tested on the validation data, with the resulting cross-validated score being averaged from all validation runs (Frey, Ruckstuhl & Sick, 2018, pp. 66–67). In the case of the k-fold cross validation, the original training dataset is split into k subsets with the same size for all subsets without replacement. The model is trained on the $k - 1$ subsets and evaluated on the k subset being the validation subset. The evaluation allows the user to assess the performance of the model out-of-sample. This process is repeated until all k subsets are used once as validation set. The resulting cross validated performance is the average of the k performance measurements on the k validation subsets (Berrar, 2018, p. 3). Figure 4 illustrates the methodology of cross-validation with an example of a five-fold cross-validation.

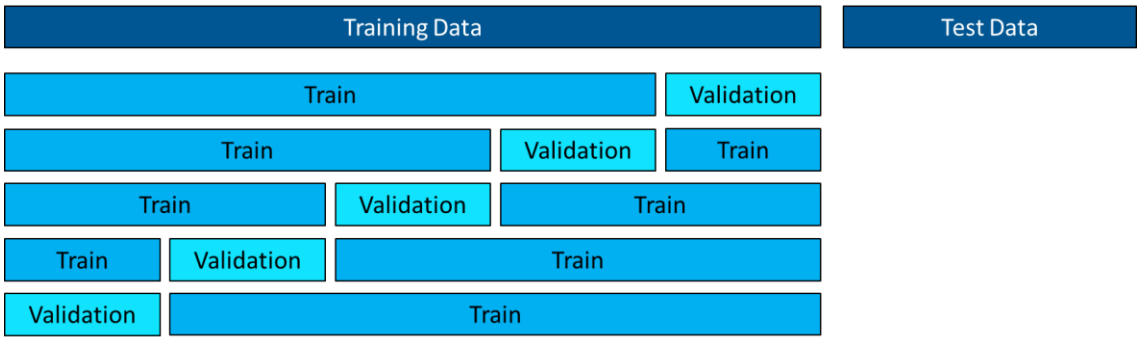

Figure 4: Illustration of the five-fold cross-validation – own illustration

Figure 4 illustrates the process of a five-fold cross-validation. The training data is split without replacement into five subsets of equal size. The figure also shows that each observation of the training data occurs once in the validation set. The advantage of this procedure is that the validation set is not seen by the algorithm during the learning process. Consequently, the resulting cross-validated score replicates the out-of-sample score that would be obtained and is, therefore, a good estimator of the true out-of-sample score which would be obtained from the test data.

2.1.4 Performance Measurement

This subchapter explains the different metrics used to evaluate a model. Since the statement of the problem in this master thesis is a binary classification, the process used in a regression or in a multiclass classification is not explicitly explained. In the case of a classification problem, there are several scores that can be used as performance measures for the evaluation of a model. Accuracy, which is the relative frequency of correct predictions, is the simplest measure. However, depending on the problem, other scores are more suitable. For example, a different problem arises when the classes in the dataset are unbalanced, that is, when class 1 (positive) occurs much more often than class 0 (negative) or vice versa. A different problem can arise if it is more important to predict only one class correctly (Frey, Ruckstuhl & Sick, 2018, pp. 64–67). Once the algorithm has predicted the data, a confusion matrix can be created.

Table 1: Explanation of the confusion matrix (Frey, Ruckstuhl & Sick, 2018, pp. 64–67)

		Predicted Class	
		Positive	**Negative**
True Class	**Positive**	True positives (TP)	False negatives (FN)
	Negative	False positives (FP)	True negatives (TN)

From the confusion matrix, several metrics can be derived, each of them having a greater importance according to a particular problem. The metrics used in this paper are listed below:

Accuracy

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Accuracy quantifies the number of predictions in which the correct class could be predicted correctly (Frey, Ruckstuhl & Sick, 2018, pp. 64–67). The metric is suitable if the importance of the correct predictions of the classes is equally important and if the classes occur homogeneously. Another advantage of accuracy is that it gives the exact proportion of the correct classified values.

Precision

$$precision = \frac{TP}{TP + FP}$$

Precision quantifies the proportion of predictions of the positive class where the classification is correct (Nguyen & Zeigermann, 2018, p. 132). Precision is suitable for causes where it is important that observations classified as positive are indeed positive, while the correct classification of the class negative is secondary.

Recall

$$recall = \frac{TP}{TP + FN}$$

Recall measures the proportion of correctly predicted positive values out of all true positive values (Nguyen & Zeigermann, 2018, p. 132). Recall is suitable for cases where

it is important to maximize the identification of positive observations, while the classification of the negative class is secondary.

F_1-Score

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

The F_1-score is the harmonic mean of recall and precision and weights them equally. The F_1-score is for cases where it is important that both recall and precision be simultaneously high (Nguyen & Zeigermann, 2018, p. 132).

Weighted Scores

The performance metrics listed above have the disadvantage that they deliver distorted results with unbalanced datasets. To avoid this problem, the scores can be calculated per class, and then the weighted score (weighted precision, weighted recall, or weighted F_1 score) can be calculated. The advantage is that this score is weighted by the number of observations in the classes and thus does not produce a biased calculation (Shmueli, 2019).

$$weighted\ score = \frac{\sum_{m=1}^{M} n_m \cdot score_m}{N}$$

where:

- M is the number of classes;
- $n_m$belongs to the data size to class m;
- $score_m$ belongs to the score (precision, recall, or F_1score) to class m; and
- N belongs to the data size of the whole dataset.

2.2 Natural Language Processing

This subchapter provides an overview of traditional natural language processing (NLP) methods. However, since these models are not used in this master thesis, their functionality will not be discussed in-depth. Nevertheless, it is important to the author of this work that these are pointed out in a certain framework, so that it can be highlighted

where their disadvantages lie. The theory of the used models is explained in more detail in the chapter 2.3.1.

NLP attempts to process natural language with the application of algorithms. Different computational techniques are used to analyze the structures in textual data, with the purpose to get a human-like language processing. To provide an overview of possible areas of application of NLP, some examples are listed below (Liddy, 2001, p. 2):

- Text Classification
- Information Retrieval
- Information Extraction
- Question-Answering
- Summarization
- Machine Translation

Since sentiment classification falls under the rubric of text classification, the same type of preliminary processing can be performed. To use the textual data as input to an algorithm, some preprocessing steps, as well as a text representation as a numerical vector, must be conducted. The general processing steps are listed below (Sarkar, Bali & Ghosh, 2018 p. 385):

1. **Sentence Splitting**
   Split a text into sentences.
2. **Tokenization**
   Split sentences into words.
3. **Stemming or Lemmatization**
   Reduce words to their base form. Where the base form of the stemming output is the dictionary form and the output of lemmatization is the root form of the word, also known as lemma.
4. **Text Cleaning**
   Remove stopwords and convert to lower case.

After the above steps have been performed, the individual words are merged back together as text. The next step is the text representation. Here, the document is represented as a vector, where the numbers in the vector represent the weights of the words it contains. To represent the words as a vector, the term “frequency-inverse document frequency” (FT-IDF) representation is used, thus allowing the texts to be entered into a model as numerical values. This model is called bag-of-words (BoW) and is the simplest, most used model. The disadvantage of this model is that some information gets lost, such as the semantics, structure, sentences, and contexts of the words. Because synonyms are different words but have similar meanings, they are handled differently in the BoW model. After the numerical text representation is created, it can be given into an algorithm where multinomial naive bayes, support vector machine, or k-nearest neighbor models are best suited for this purpose (Sarkar et al., 2018 p. 386). As mentioned above, this methodology has the disadvantage that some information, such as semantics, is lost in the BoW model. The next subchapter, Deep Learning, illustrates how this problem can be counteracted.

2.3 Deep Learning

This subchapter introduces some deep learning models and illustrates how they can be used to counteract the classical problems associated with NLP. It also shows where their advantages lie, and which problems cannot be solved through their application. The latter to motivate the application of state-of-the-art models in the field of NLP, the transformers. Furthermore, the next subchapter discusses the state-of-the art models and their functionality. However, to get a deeper insight into the mathematics of the models, the reader is advised to consult the referenced literature.

To counteract the problem of sequence transduction and thus improve the performance of a model in the area of NLP, an algorithm must be used which has a kind of memory. The sentence below better illustrates the problem:

This summer I'm going on vacation to Spain. I hope the weather is nice there.

When reading "there" in the second sentence, a human understands that "there" refers to "Spain." To recognize this dependency while modeling, a model must be able to recognize

these dependencies and connections. Recurrent neural networks (RNNs) have been used to deal with problems of this nature (Ciacaglia, 2019). To understand how such dependencies are included with the application of RNNs, Figure 5 below illustrates the properties of an RNN:

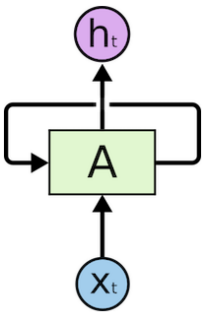

Figure 5: Structure of an RNN (Ciacaglia, 2019)

Where:

- x_t is the input;
- A is the recurrent neuron; and
- h_t is the output of the recurrent neuron.

According to the figure above, it can be seen that the recurrent neuron not only has the output h_t but also has an output that is passed to itself. To understand how this process looks with several inputs, a figure is listed again, in which the RNN is shown rolled up.

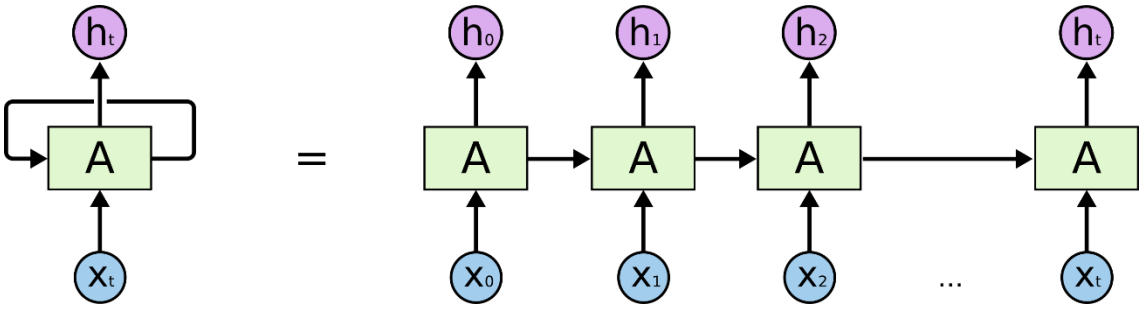

Figure 6: Structure of an unrolled RNN (Ciacaglia, 2019)

It can be seen that the output of the recurrent neuron is transmitted and is thus considered again when the next inputs are processed. This chain-like nature makes it possible to incorporate the connections and dependencies mentioned earlier by combining the information from the previous word with the information from the next words. In practice,

RNNs have proved to be good at recognizing and modeling such dependencies. Nevertheless, they have the problem that if the sentences are longer and the dependencies thus have a large distance, they do not recognize the dependencies sufficiently well. This is due to the fact that the information of the words is always transferred by their chain-like nature. The longer this chain becomes, the more difficult it is for the network to process the information, which already lies far back. The information is therefore lost. Long short-term memory neural networks (LSTMs) are special types of RNNs that can counteract the problem of long-term dependencies. The neurons of LSTMs have a more complex mechanism in which the information is processed differently. The goal of this mechanism is to remember important information and to forget unimportant information in order to recognize long-term dependencies. The graphic below illustrates the process within a neuron of an LSTM (Ciacaglia, 2019):

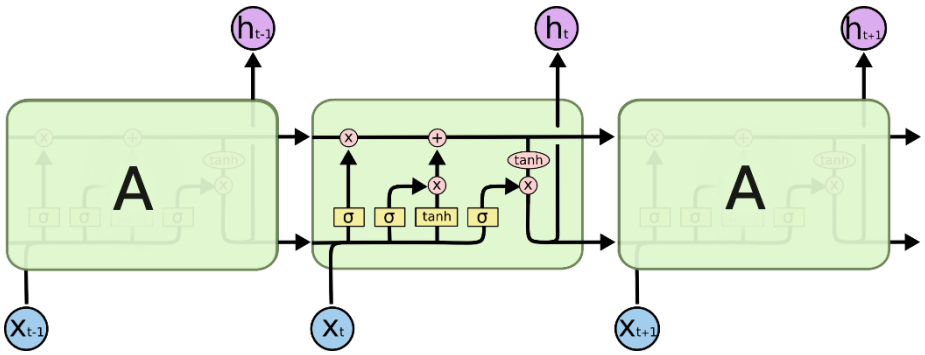

Figure 7: Structure of an LSTM neuron (Ciacaglia, 2019)

In practice, LSTMs perform better than RNNs in the area of NLP, but still exhibit the same problems. If the relevant information in the texts is separated from each other too far, then the performance of LSTMs is still unsatisfactory. Likewise, neural networks cannot be parallelized due to their architectural properties, and word-by-word respectively their embeddings have to be added as input.

2.3.1 Transformers

Vaswani et al. (2017, pp. 6,000–6,010) present in their published research paper a new state-of-the-art network architecture: the transformer. According to their research, the transformers, which are not based on a recurrent network architecture and only on an attention mechanism, achieve a better performance in several NLP fields than models

based on a recurrent or convolutional architecture. To better understand how the transformers interact, their model architecture is illustrated below:

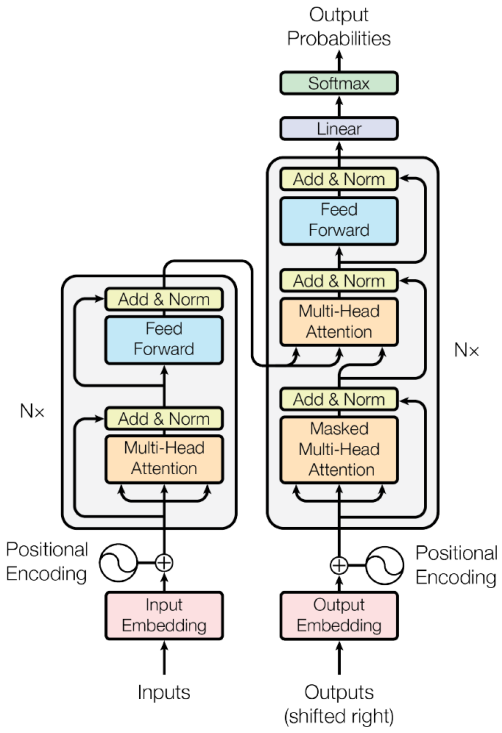

Figure 8: The transformer model architecture (Vaswani et al., 2017, pp. 6,000–6,010)

The transformer has an encoder-decoder structure, where the left part of the figure illustrates an encoder and the right part a decoder. The encoder is composed of $N = 6$ identical layers, each layer containing two sublayers. The first is a multi-head self-attention mechanism and the second is a simple, position-wise fully connected feed-forward network. Both sublayers are followed by a normalization. The decoder on the right part of the figure also contains $N = 6$ identical layers, which additionally include a third sub-layer performing multi-head attention over the output of the encoder. Each of the three sublayers is followed by a normalization as in the case of the encoder. As inputs, the input embeddings are fed to the encoder and the output embeddings to the decoder, whereby pre-learned embeddings are used to convert the input tokens and output tokens to vectors.

After the inputs are fed into the transformer, the positional encoding is performed first. Since the transformers are not recurrent networks and they process the inputs of the

encoder and the decoder in parallel – that is, at the same time – position must be assigned to the inputs. To prevent forward-looking, the position of the output embeddings is shifted by one position (Vaswani et al., 2017, pp. 6,000–6,010). To eliminate the problems (already pointed out several times) which arise from the application of common NLP methods, the attention mechanism is used. The attention function can be described as mapping a query and a set of key-value pairs, producing an output. Whereby the query, keys, values, and the output are all vectors. This is done by applying dot-products, which in practice is much faster and more efficient than, for example, using an additive attention function. According to Vaswani et al. (2017, pp. 6,000–6,010), this process is called scaled dot-product attention. For all inputs to be processed in parallel, this process is performed simultaneously h-times for all word embeddings in the inputs. Whereby this process is called multi-head attention.

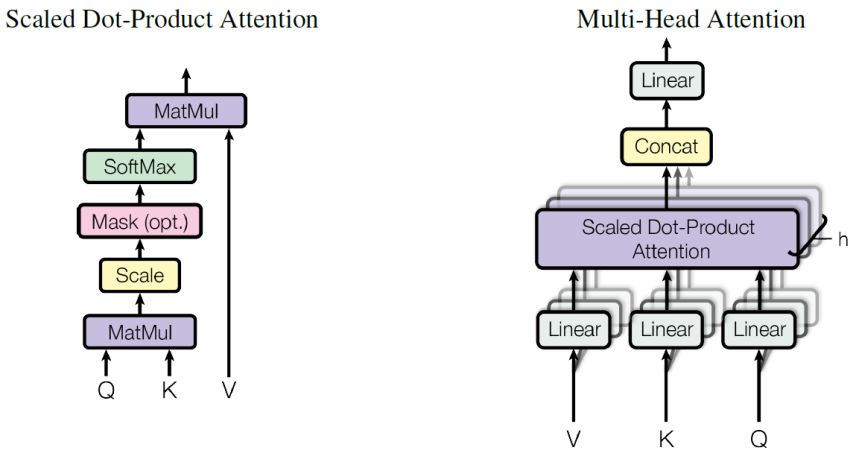

Figure 9:(left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel (Vaswani et al., 2017, pp.6,000–6,010).

Where:

- Q are the query vectors;
- K are the key vectors; and
- V are the value vectors.

Multi-head attention allows the model to get information from different representations at different positions. In other words, during encoding, each word in the sentence (that is, each position in the input sentence) is compared with all other words that occur in the preceding or respectively left side of the sentence. This allows the obtaining of useful information to improve the encoding of the whole sentence. Due to the architecture of the multi-head attention mechanism, this process is performed simultaneously for all words

in the sentence. This process can be viewed in the same way in the decoder, with the difference that the sentences are decoded there (Alammar, 2018).

2.4 Software

The empirical research of this master thesis was conducted in the Python programming language. The programming language as well as the most important packages used are briefly explained below.

2.4.1 Python

Python is an open-source programming language developed by Guido van Rossum. Python supports several programming paradigms such as object-oriented, aspect-oriented, and functional programming. The first full version was released in January 1994. Since then, Python has been continuously developed and newer versions have been released. The current version is 3.9, released in May 2021 (Python Software Foundation, 2021).

2.4.2 Pandas

Pandas is an open-source package that provides an interface to process and manipulate datasets. Pandas was developed by AQR Capital Management in 2008 and made available as an open-source package in late 2009. The goal of the developers is to create a fundamental high-level package that allows data analysis in Python. Furthermore, it is another goal that Pandas becomes the most powerful and flexible open-source data manipulation tool available in any programming language (Pandas Developers, 2021).

2.4.3 Scikit-learn

Scikit-learn is an open-source package that provides state-of-the-art implementations of many well-known machine learning algorithms (Predregosa et al., 2011). The development of the package started in 2007 as a Google Summer Code project and was continued by Matthieu Brucher as part of his thesis. In 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, and Vincet Michel took over the project and made the first public release in 2010 (Scikit-learn Developers, 2021).

2.4.4 Transformers

Transformers is a state-of-the-art open-source package for NLP tasks. The package contains 32+ pre-trained models in more than 100 languages and reaches excellent results on various NLP tasks (The Hugging Face Team, 2021).

2.4.5 Beautiful Soup

Beautiful Soup is an open-source Python package that allows the user to scrape information from XML and HTML documents and thus information from web pages. Likewise, the parse trees can be iterated, searched, or even modified (Richardson, 2021). This Python library was used in this master thesis to scrape additional textual data.

3 Data

This chapter presents the different datasets used in this master thesis as well as their descriptive statistics. For the datasets where it is appropriate, the preprocessing process is also shown, which is required before the data can be given as input to a machine learning model.

3.1 Financial Phrase Bank

The Financial Phrase Bank dataset contains 5,000 phrases in the field of finance and economics and was first used and published by Malo, Sinha, Takala, Korhonen & Wallenius (2013). Because there are few satisfactory datasets of financial text data publicly available in the field of finance and economics, the Financial Phrase Bank dataset was created. It is intended to establish new standards for modeling techniques in a financial context.

During the creation of the dataset, the goal was to classify each sentence into a positive, negative, or neutral category by considering only the information explicitly available in that sentence. For this purpose, each sentence was annotated by 16 annotators with sufficient background expertise in financial markets. Three of the annotators were researchers, and the remaining 13 were master's students at the Aalto University School of Business with majors in finance, accounting, and economics. The annotators were asked to consider the sentences from only an investor's point of view, that is, whether the news could have a positive, negative, or neutral impact on the stock price. Sentences that do not appear to be relevant are considered neutral (Malo et al., 2013).

Once the 5,000 sentences have been classified by the annotators, it is necessary to aggregate these classifications. Malo et al. (2013) provided four datasets that differ in the agreement rate of the classifications. The agreement rates of the classifications in the datasets are 100%, more than 75%, more than 66%, and more than 50%. In their published paper, it turned out that the performance of a classification model did not vary much, depending on which of these datasets it was trained on. Accordingly, the author of this master thesis decided to use the dataset with more than 50% agreement. The advantage is that this dataset contains 4,846 observations, respectively classified sets, whereas the

number of observations decreases for the more precisely classified datasets. In Table 2, three positive, three neutral, and three negative sentences are shown as examples:

Table 2:Sentences of the Financial Phrase Bank Dataset (Malo et al., 2013)

Positive Sentiment
Sales have risen in other export markets.
The fair value of the property portfolio doubled as a result of the Kapiteeli acquisition and totalled EUR 2,686.2 1,259.7 million.
Componenta increased its stake in Turkish steel company Doktas Dokumculuk Ticaret ve Sanayi A.S. to 92.6 pct stake in March 2007.
Neutral Sentiment
When this investment is in place, Atria plans to expand into the Moscow market.
The fuel purchase contracts have been signed with three months' delivery from this September to November.
However, the brokers' ratings on the stock differ.
Negative Sentiment
However, the growth margin slowed down due to the financial crisis.
The company slipped to an operating loss of EUR 2.6 million from a profit of EUR 1.3 million.
The low capacity utilization rate in steel production considerably increases the fixed costs per unit of steel produced.

The dataset contains 2,879 neutral, 604 negative, and 1,363 positive observations. It is used further in the next chapter of the empirical research to fine-tune a sentiment classification model.

3.2 Financial News Dataset from Bloomberg and Reuters

The financial news dataset collected by Ding, Zhang, Liu & Duan (2014, p. 2) was used to predict the sentiment, and thus to predict further stock price movements. This dataset seems to be one of the largest public available datasets which can be downloaded from Remy & Ding (2015). The Bloomberg dataset contains 447,279 observations, which include the publication date, the title, and the content of the news.

Figure 10 provides an overview of the amount of news by publication date:

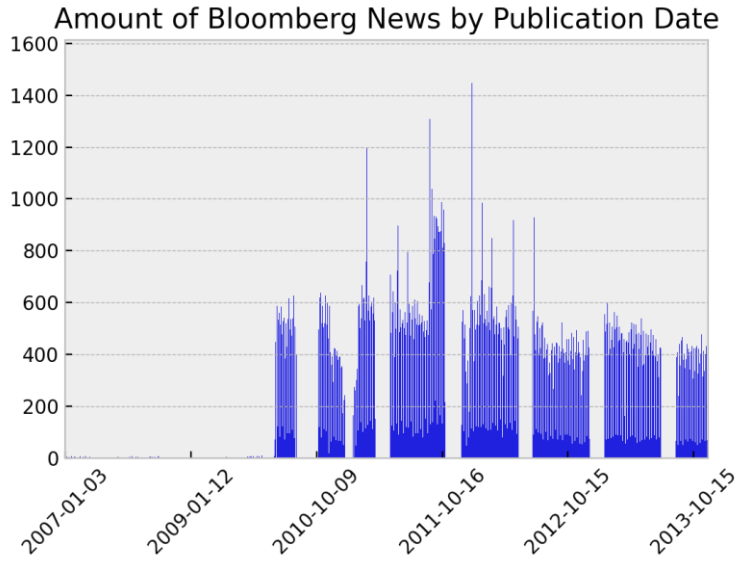

Figure 10: Amount of Bloomberg News (Remy & Ding, 2015) by publication date – own illustration

It can be seen in Figure 10 that the news was published between 01/03/2007 and 11/26/2013. It can also be seen that the number of messages per day varies significantly, and that at the beginning, there are only a few news items available for each day. Furthermore, the dataset shows some timestamps which do not contain any news. In order to further analyze the dataset, the number of messages per weekday was plotted in a bar chart:

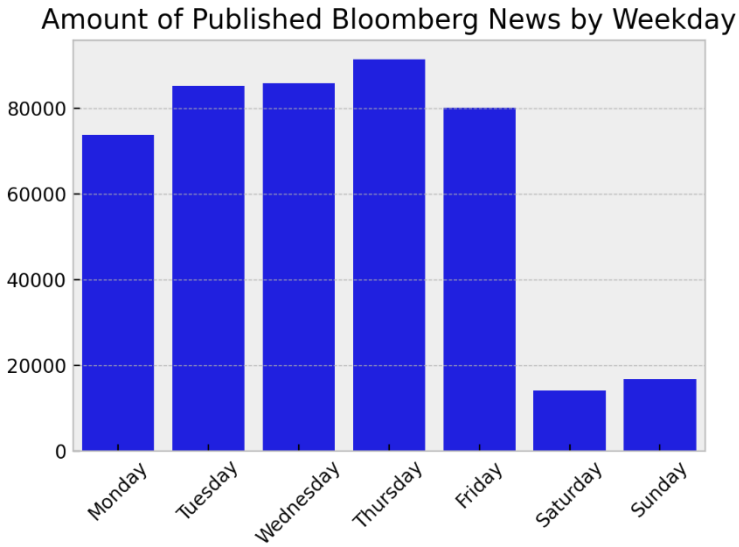

Figure 11: Amount of Bloomberg News (Remy & Ding, 2015) by weekday – own illustration

Figure 11 illustrates that most of the news items were published between the midweek days. The amount of news published on Saturday and Sunday is almost a quarter compared to the other days of the week. After the Bloomberg dataset has been considered, the same procedure is performed for the Reuters news dataset. Unlike the Bloomberg dataset, the Reuters news dataset contains 8,556,310 news items published between 01/01/2007 and 08/16/2016. The dataset includes the publication date, title, and URL link for each observation.

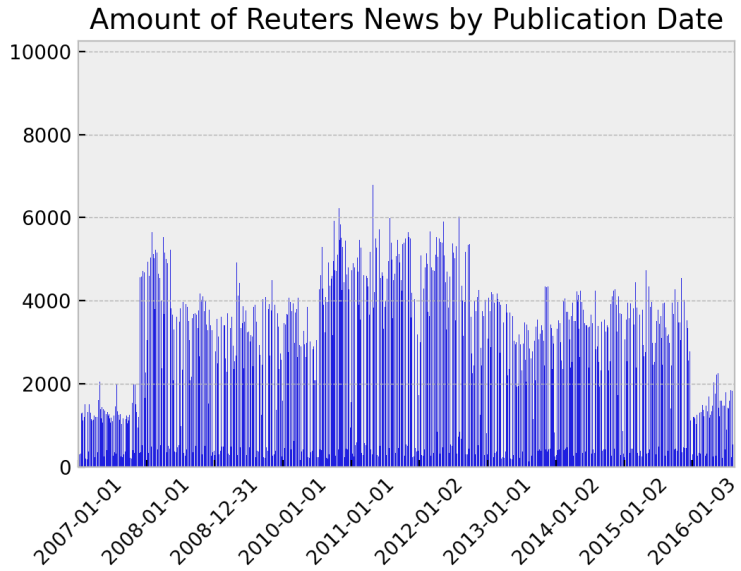

Figure 12: Amount of Reuters News (Remy & Ding, 2015) by publication date – own illustration

Figure 12 illustrates that the number of news items per publication date does not fluctuate as much as in the Bloomberg dataset. Furthermore, the dataset does not show any time periods when no news was published.

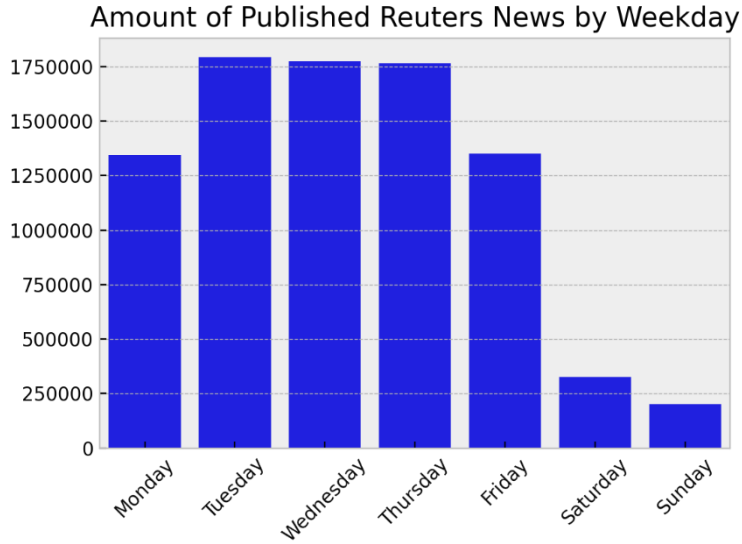

Figure 13: Amount of Reuters News (Remy & Ding, 2015) by weekday – own illustration

From Figure 13 above, it can be seen that the number of news titles published per weekday behaves similarly to the Bloomberg dataset. After both datasets have been shown and explained, the next subchapter describes the web scraping as well the preprocessing of both datasets.

3.2.1 Web Scraping and Preprocessing of the Financial News Datasets

The first step is to delete all observations with an empty content in both datasets. Since the Bloomberg dataset has the title and associated content, this dataset is used as the reference dataset. This means that on the days when there are not any or too few observations in the Bloomberg dataset, additional news is downloaded via web scraping. As mentioned previously, the Reuters dataset shows per observation the publication date, the title, and the URL link, which is used for web scraping. However, to determine which news items are additionally downloaded, the number of published news items in the Bloomberg dataset was analyzed. Thereby, the date when fewer than 50 news items are available is written out and stored in a date vector. This date vector is then used to filter out the URL links from the Reuters dataset. For a better overview of the process, see the list below:

1. Filter out the data where the Bloomberg dataset contains 50 or fewer observations, with the output of this step being a date vector.
2. Extract the observations of the Reuters dataset according to the date vector of step (1).
3. Apply the URL link of the filtered observations from step (2) to download additional articles via web scraping. If there are more than 200 articles, 200 are randomly selected.
4. Since the Reuters dataset has a longer date vector than the Bloomberg dataset, additional 200 articles per day are downloaded via web scraping from 11/26/ 2013 (date of the last observation of the Bloomberg dataset) to 08/16/2016 (date of the last observation of the Reuters dataset).

In steps (3) and (4), some URL links, which do not work, were adjusted by trial and error method to make them valid again. The problem with these links was that the year in which the news was published was also included in the link and had to be removed. To check if these really are the correct new items, the author of this master thesis randomly read through some of the contents and checked their publication date. Since via web scraping

all contents of the homepage are retrieved, the title was also downloaded. In a second control, the downloaded title was compared to the titles from the Reuters dataset. Once the 345,776 additional articles were downloaded, this data was merged with the Bloomberg dataset. However, to use the dataset as input for a machine learning model, a constant number of observations per day is required. Consequently, the dataset is preprocessed even further so that it contains 58 observations per day. This corresponds to the smallest number of available observations per day and, in the process, the 58 observations per day are selected randomly.

3.3 Timeseries

The Standard & Poor's 500 Index (S&P 500®) was used as the time series. The S&P 500 is considered the best single indicator for US large cap equities. The index comprises 500 leading companies from the United States and covers approximately 80% of the available market capitalization. It is rebalanced once a year, whereby new companies can be included in the index or existing companies can be excluded. Rebalancing takes place quarterly, whereby the weights of the individual companies in the index are adjusted (S&P Dow Jones Indices, 2021).

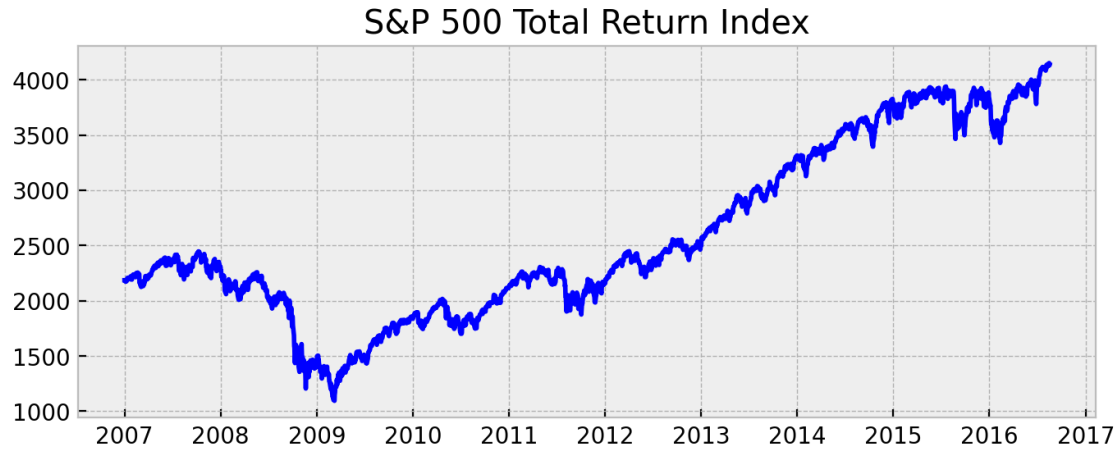

Figure 14: Price chart of the S&P 500 total return index – own illustration

The above chart shows the closing prices of the S&P 500 total return index. Total return means that in the performance measurement not only the price development but also reinvested cash distributions, such as reinvested dividends, are taken into account. The reason for using a total return index and not a price index is that, for example, dividend distributions influence the price of a stock and news reports write about these

distributions. Since this master thesis is about price direction prediction, an attempt is made to predict the direction of the price for the next day. Accordingly, the time series must still be preprocessed such that a binary classification can be performed. First, the daily returns are calculated using the formula below.

$$r_t = \frac{P_t - P_{t-1}}{P_{t-1}}$$

Where:

- r_t is the relative return at time t
- P_t is the price at time t
- P_{t-1} is the price at time $t-1$

Once the daily relative returns have been calculated, they are further preprocessed using the signum function.

$$d_t = sign(r_t) = \begin{cases} 1 \; if \; r_t > 0 \\ 0 \; if \; r_t = 0 \\ -1 \; if \; r_t < 0 \end{cases}$$

Where:

- d_t is the price direction at time t
- r_t is the relative return at time t

According to the above formula, the price direction has a value of zero if the return also has a value of zero. In this case, the problem to be solved would be a multi-classification problem. Since it is more of interest to predict a rising or falling return, the price directions which have a value of zero were over-sampled with the previous day's value. In this case, the problem to be solved turned to a binary classification problem.

4 Empirical Research

In this chapter, the models discussed in the theory are implemented and applied using the Python programming language. Figure 15 below provides an overview:

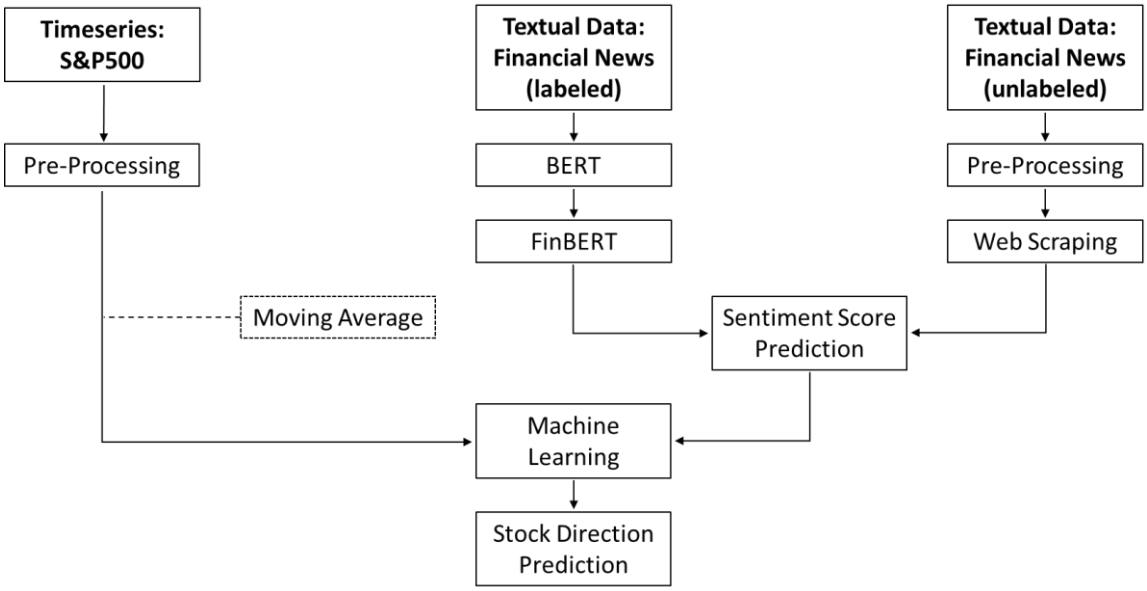

Figure 15: Overview of the empirical research – own illustration

The purpose of the empirical research is to predict the S&P 500 stock market index using the text data explained in the previous chapter. For this purpose, a pre-trained transformer network is fine-tuned on the labeled dataset, the financial phrase bank dataset presented in chapter 3.1. After the model is fine-tuned, the preprocessed textual data from the Bloomberg and Reuters dataset, presented in Section 3.2, is fed as input to the model and sentiment scores are calculated from the 58 daily news stories. A random forest model is used as a machine learning model to predict the stock market index price direction based on the calculated sentiment scores. As pointed out by Liu, Cheng, Su, and Zhu (2018, p. 1), in previous researches where textual data is used to predict stock prices, there are disagreements whether to use the title or the whole content. For this reason, in this master thesis, different models are developed that predict stock price direction using the title, the contents, the titles combined with the contents, and the titles combined with the contents and some moving averages of the S&P 500 stock market index.

Thereby, the implementation of the pre-trained transformer networks is explained in chapter 4.1. In chapter 4.2, a simple stock price direction prediction based on the sentiment scores without the use of machine learning is presented. In chapter 4.3 the

implemented random forest models are discussed. A comparison of the results of the machine learning models is provided in chapter 5.1.

4.1 Sentiment Score Calculation

Bidirectional encoder representations from transformers (BERT), the used pre-trained transformer networks, have been published as an open-source package by Devlin, Chang, Lee & Toutanova (2019, pp. 4,171–4,186). The pre-trained BERT models can be loaded from the Python package transformers, as explained in Section 2.4.4, and can thus be applied for specific natural language tasks. BERT achieves state-of-the-art results in eleven NLP tasks and can be fine-tuned by adding an additional layer for specific NLP tasks, which achieves excellent results compared to traditional methods (Devin, Chang, Lee, & Toutanova, 2019, pp. 4,171–4,186). The model architecture, as well as the concept of BERT, is based on the transformer networks, which were first published by Vaswani et al. (2017, pp. 6,000–6,010) and explained in chapter 2.3.1. During the pre-training, the model was trained on unlabeled data over different tasks. Therefore, by adding a single output layer and subsequent fine-tuning, the model performs well in countless NLP tasks such as question-answering and language inference, etc.

While doing the pre-training, the WordPiece embeddings (Wu et al., 2016) were used (Devin et al., 2019, pp. 4,171–4,186). In contrast to the transformer architecture published by Vaswani et al. (2017, pp. 6,000–6,010), the architecture of BERT does not include any transformer decoder layer but has transformer encoder layers. The encoder layers serve to decompose and understand the inputs, whereas the additional output layer is task specific and may be regarded as a decoder from the point of view of its function. Another difference is that in the self-attention mechanism not only the words on the left side of the sentence but also the words on the right side of the sentence are considered. This is also the reason why there are no decoders in the BERT architecture. Devlin et al. (2019, pp. 4,171–4,186) published two BERT architectures, which differ based on their size. $BERT_{BASE}$ contains 12 encoder layers, and $BERT_{LARGE}$ contains 24 encoder layers.

The BERT model used in this master thesis is FinBERT (Araci, 2019), which uses the $BERT_{BASE\ uncased}$ model as a pre-trained model. Uncased means that the text is lowercased before training is started. In addition to the $BERT_{BASE\ uncased}$ model, a classification layer

was added as output layer, which predicts the labels positive, neutral, and negative. Some model parameters are listed below:

- train_batch_size: 32
- eval_batch_size: 32
- max_seq_length: 48
- learning_rate: $2 \cdot 10^{-5}$
- epochs: 4
- lower_case: True
- encoder_no: 12

FinBERT was created in a previous work by Araci (2019) with the aim of predicting sentiment analysis based on textual financial data. Accordingly, the "Fin" can also be derived, as it is a BERT model in context of finance. The batch size is 32 in the training as well as in the evaluation. The parameter "max_seq_length" stands for the maximum length of the sentences that are used. This means that for a sentence with a length of 70 words, only the first 48 words are used. The learning rate and the epochs were initiated with values of $2 \cdot 10^{-5}$ and 4, respectively. The model parameter "lower_case" means that the texts are written in lower case. Since the BERT base model is the $\text{BERT}_{\text{BASE uncased}}$ model, it makes no difference if this parameter is set to True since this is already done in the base model, anyway. The parameter "encoder_no" stands for the number of pre-trained encoders that are fine-tuned during the downstream task. In this case, all 12 encoders are adjusted during fine-tuning.

For fine-tuning, the subset of dataset with more than 50% agreement from the Financial Phrase Bank dataset is used, where the dataset is split into a train, validation, and test dataset with the proportions of 72-8-20. As pointed out in Section 3.1, the target variable is the sentiment of the news, with the three labels being positive, neutral, and negative. Accordingly, the model is trained, validated, and then tested on the test dataset. The resulting confusion matrix is shown in Figure 16:

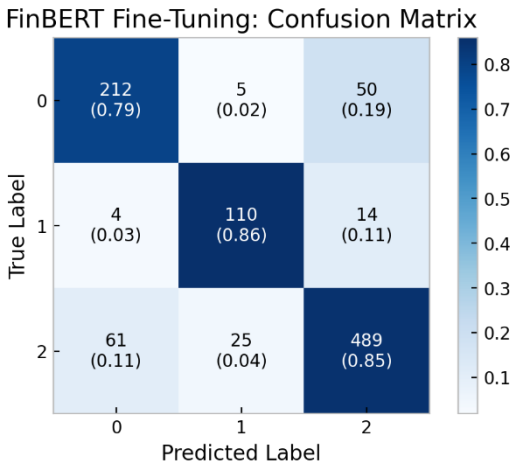

Figure 16: Confusion matrix for FinBERT sentiment classification – own illustration

$$accuracy = 0.836$$
$$weighted\ precision = 0.839$$
$$weighted\ recall = 0.836$$
$$weighted\ f1 - score = 0.837$$

The fact that all performance indicators have a rounded value of 0.84 is a coincidence. The labels 0, 1, and 2 in the confusion matrix represent the labels positive, negative, and neutral. After the FinBERT model has been fine-tuned on the Financial Phrase Bank dataset, the sentiment score prediction of the aggregated Reuters and Bloomberg dataset presented in chapter 3.2.1 is started. For this purpose, the fine-tuned FinBERT model is used, and the sentiment scores of the titles as well as the contents are predicted. In BERT models, individual sentences are given as input instead of the whole texts as observations. The added output layer in FinBERT provides as output the probabilities whether the financial text is classified as positive, neutral, or negative. The highest of the three probabilities is used for the classification, as it is the case in fine-tuning. To calculate the sentiment score, it is defined that a score of 1 is highly positive and a score of -1 is highly negative. A sentiment score of 0 can be considered neutral. However, to better understand how the sentiment score is calculated, the following sentence is used as an example:

“TREASURIES–Bonds creep higher in thin trade.”

This sentence is also a title of an observation of the Reuters dataset and was published on 01/02/2007. After the sentence is given as input to the FinBERT model, the model provides the following probabilities for the sentiment classification:

$$positive = 0.965$$
$$negative = 0.0175$$
$$neutral = 0.0175$$

The calculation of the sentiment score is simple and can be calculated from the difference of the probability that a text is positive and the probability that a text is negative.

$$Sentiment\ Score = 0.965 - 0.0175 = 0.9475$$

It can be seen that the sentiment score is high and also corresponds to a classification of positive. Texts that are classified as neutral usually also contain a low probability of a positive or negative label. Consequently, their sentiment score value is around 0. Since in BERT models sentences and not whole texts are considered as observations and a news story often consists of several sentences, the average value of all sentiment scores from the sentences is used as sentiment score for the entire news content.

Once the sentiment scores of the 58 news stories per day are predicted, two new DataFrames are created, which can be used as input into a machine learning model. The DataFrames have the date vector as index and columns with the 58 sentiment scores, whereby the datasets differ in that one contains the sentiment scores of the titles and the other the sentiment scores of the content.

In the random forest models, the sentiment scores are used as explanatory variables, and the price direction of the S&P 500 stock market index, described in chapter 3.3, is used as the target variable. Since the stock index is only traded on weekdays, and thus contains a different date vector than the DataFrames with the sentiment scores, the DataFrames of the sentiment scores are reindexed to the date vector of the S&P 500 stock index. This means that published messages on weekends are deleted from the DataFrames and that if there are no sentiment scores on a trading day, the values are forward-filled with the previous sentiment scores. This also prevents that no news respectively sentiment scores are used for the prediction of the return direction of a the same day. To continue using the

scores from the weekends, the sentiment scores from Friday, Saturday, and Sunday are averaged so that they can be used to predict the price direction for Mondays.

4.2 Stock Price Direction Prediction Based on Sentiment Scores

This chapter presents a simple stock price direction prediction based on the sentiment scores from the contents and the titles. In a first step, the average of the 58 sentiment scores is calculated per day. In a second step, the signum function is applied to this average. If the average is greater than 0 and thus represents a rather positive sentiment, it is rounded up to 1. Accordingly, a negative average value is rounded down to -1. Subsequently, this vector with the trading signals is shifted by one day and multiplied by the returns of the next trading day. As an example, one can use the news that was published on Monday. First, the sentiment scores are calculated as described in the previous chapter, and then the average value is calculated from them, rounded to 1 or -1. A positive value is rounded up to 1, which stands for a buy signal, and is multiplied by the return on Tuesday. A negative value is rounded to -1 and multiplied by the return on Tuesday, which represents a 100% short position. For this purpose, transaction costs are neglected; however, it is possible to replicate this trading strategy cheaply using S&P 500 futures. In order to compare this simple strategy with the strategies explained in the next chapter, the returns are calculated and indexed from 01/01/2011 to 08/16/2016, where the starting value is 100. This can be considered as being invested in 100 USD at the beginning.

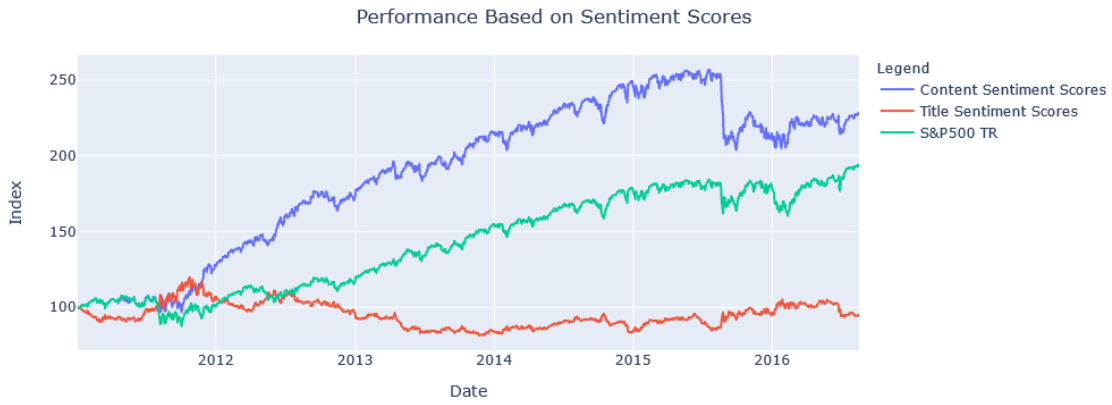

Figure 17: Performance based on sentiment scores – own illustration

In Figure 17 above, it can be seen that the strategy based on the content sentiment scores outperforms the S&P 500 stock market index. A closer look shows that an

outperformance is generated at the end of 2011, which explains the outperformance of this strategy since in the following years, all price downturns are taken along and are not correctly reflected by the content sentiment scores, as it is the case for example in the price decline in 2015. The strategy based on the title sentiment scores is generally poor and even shows a negative performance in this period. This also reflects the findings from previous research (Cheng, Su, & Zhu 2018, p. 1) that the content of the news should also be taken into account and not only the title should be used to calculate the sentiment scores, which should reflect the market movements. Table 3 below provides a better insight into the strategies:

Table 3: Performance measures of strategies based on sentiment scores from 01/01/2011 to 08/16/2016.

	S&P 500	Content SC[1]	Title SC[1]
Return p.a.	12.045%	15.208%	-0.934%
Volatility	15.108%	15.097%	15.128%
Return / Risk	0.797	1.008	-0.062
Max. Drawdown	18.641%	20.598%	32.089%

[1] *The abbreviation SC stands for sentiment scores*

The annualized returns are calculated using the geometric average, and the volatility is calculated using the standard deviation of the returns. To provide a better overview of the calculation, the formulas (Posth, 2019, pp. 27–39) are listed below:

$$return\ p.a. = \left(\prod_{t=1}^{T}(1 + return_t)\right)^{\frac{252}{T}} - 1$$

$$volatility = \sqrt{\frac{1}{T-1}\sum_{t=1}^{T}(return_t - \overline{return})} \cdot \sqrt{252}$$

$$return\ /\ risk = \frac{return\ p.a.}{volatility}$$

$$max.drawdown = \frac{Peak - Lowest\ value}{Peak}$$

Where peak refers to the peak value before largest drop, and the lowest value refers to the lowest value before a new peak is established, considering the whole period.

4.3 Random Forest Classification

This chapter presents the simulation settings for the implementation of the random forest algorithm. The settings for all simulations, which are presented in the next subchapters, are always the same and only differ from the input data.

Since the financial text data is available from 01/01/2007 to 08/16/2016, not just one model is fitted on one period and then tested on another period. The implemented random forest models are first fitted on the first four years and tested on the following year. In a second run, the models are fitted on the first five years and tested on the sixth year, and so on. To provide a better overview, the in-sample and out-of-sample periods are listed in Table 4 below:

Table 4: In-sample and out-of-sample periods

In-sample Period	Out-of-sample Period
01/01/2007 to 12/31/2010	01/01/2011 to 12/31/2011
01/01/2007 to 12/31/2011	01/01/2012 to 12/31/2012
01/01/2007 to 12/31/2012	01/01/2013 to 12/31/2013
01/01/2007 to 12/31/2013	01/01/2014 to 12/31/2014
01/01/2007 to 12/31/2014	01/01/2015 to 12/31/2015
01/01/2007 to 12/31/2015	01/01/2016 to 08/16/2016

During the fitting on the in-sample time period, some hyperparameters are tested at the same time. Therefore, the implemented grid search method of Scikit-learn is used for hyperparameter tuning. Likewise, the train dataset is cross-validated, with a train validation split proportion of 87.5–12.5. The input variables are always the calculated sentiment scores, and the target variable the price direction of the next trading day. Table 5 below lists the fine-tuned hyperparameters:

Table 5: Tested parameter during hyperparameter tuning

Parameter	Value
n_estimators	250, 500, 750, 1000
max_depth	1, 3, 5, 10, 20, 30
max_features	2, 5, 10, 15, 30, 40, 58

criterion	gini, entropy
random_state	333
bootstrap	True

The left side of the table above lists the parameters, which are fine-tuned during the hyperparameter tuning. The values tested are shown on the right side. The method "GridSearchCV" tests all possible parameter combinations and cross-validates them simultaneously. The model is also given a score as a performance measurement. This means that the model tests different random forests with the specified parameters from Table 5, cross-validates them, and returns the random forest model with the highest cross-validated score as output. The model with the highest cross-validated score is then fitted to the whole in-sample dataset and applied during the out-of-sample period. The weighted f1-score and the weighted precision score are used as performance scores. It is important to note that this procedure is applied to all simulations. The results of the simulations where the weighted precision score is used as performance score are shown in the next subsections. Since the procedure remains the same for different performance metrics, the results where the weighted f1-score is used as performance metric are only shown in chapter 5.1 and compared with the other results.

4.3.1 Stock Price Direction Prediction Based on Title Sentiment Scores

In a first run, the 58 daily title sentiment scores are used as input for the random forest models, where first the hyperparemeters are fine-tuned according to the grid search method described in the previous chapter. An overview of the fine-tuned hyperparameters and the cross-validated weighted precision is listed in Table 6 below:

Table 6: Fine-tuned hyperparameter and cross validated weighted precision scores based on title sentiment scores

Fine-tuning until 2010	Fine-tuning until 2013
n_estimators: 250	n_estimators: 1000
max_depth: 5	max_depth: 3
max_features: 5	max_features: 30
criterion: entropy	criterion: entropy
cv weighted precision: 0.566	cv weighted precision: 0.611

Fine-tuning until 2011	Fine-tuning until 2014
n_estimators: 750	n_estimators: 500
max_depth: 20	max_depth: 5
max_features: 40	max_features: 15
criterion: entropy	criterion: gini
cv weighted precision: 0.527	cv weighted precision: 0.533
Fine-tuning until 2012	Fine-tuning until 2015
n_estimators: 500	n_estimators: 250
max_depth: 20	max_depth: 5
max_features: 2	max_features: 30
criterion: entropy	criterion: gini
cv weighted precision: 0.529	cv weighted precision: 0.512

Considering Table 6 above, the parameters fluctuate significantly, suggesting that there is no general parameter constellation that provides a robust model. Likewise, the cross validated weighted precision of the fine-tuned models is listed. It can be seen that the cross validated weighted precision is always above a value of 0.50. After the models have been optimized on the in-sample data, they are again fitted with the parameter values listed in Table 6 on the whole in-sample periods without cross validation and then tested on the out-of-sample data. Subsequently, the out-of-sample confusion matrix is illustrated with the corresponding out-of-sample performance measures in Figure 18 below:

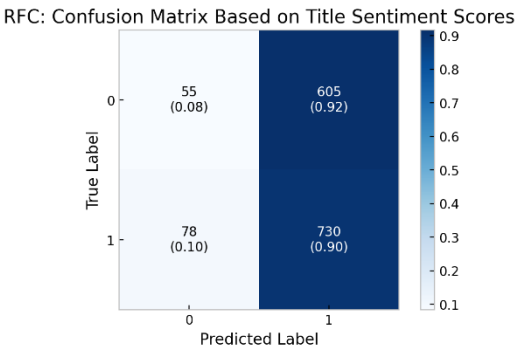

Figure 18: Out-of-sample confusion matrix of random forest classification based on title sentiment scores – own illustration

$$accuracy = 0.535$$
$$weighted\ precision = 0.487$$
$$weighted\ recall = 0.535$$
$$weighted\ f1 - score = 0.437$$

The out-of-sample weighted precision is slightly below the optimized cross-validated weighted precision scores in Table 6, indicating that the models are generally well-fitted and not over- or underfitted. Similarly, from the confusion matrix, it can be seen that most of the predictions predict the label 1, which indicates a rising price. It can be said that this is generally not a bad thing, as the S&P 500 stock market index showed a strong upward trend during this period. Consequently, it is not bad to predict an uptrend over a longer period of time and to take the less present days where the stock market index has a negative return. However, in order to provide a better overview, the indexed out-of-sample performance is shown in Figure 19 below:

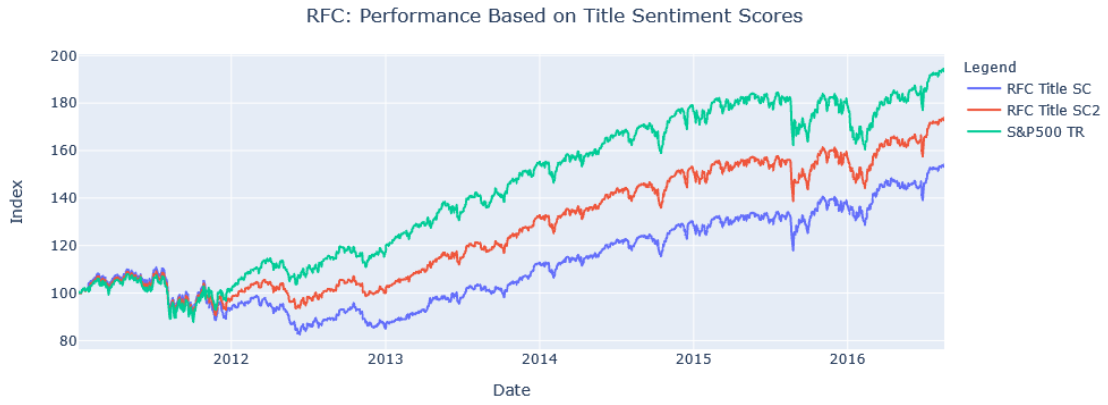

Figure 19: Out-of-sample performance random forest classification based on title sentiment scores – own illustration

In the figure above, it can be seen that no strategy can outperform the S&P 500 stock market index. However, it is worth mentioning that the strategy "RFC Title SC" stands for random forest classification based on title sentiment scores and takes a short position when a negative return is predicted. In the second strategy, if a negative return is predicted, only divestment is done, which means that the simulated return on such days is zero. Between the two strategies, strategy 2 performs better because in the case of a false negative prediction, only divestment is done, and no short position is taken. The financial performance measures are shown in the Table 7 below:

Table 7: Performance measures of strategies based random forest classification on title sentiment scores from 01/01/2011 to 08/16/2016.

	S&P 500	RFC Title SC	RFC Title SC 2
Return p.a.	12.045%	7.673%	9.920%
Volatility	15.108%	15.119%	14.606%
Return / Risk	0.797	0.508	0.679
Max. Drawdown	18.641%	25.518%	18.669%

Compared to the S&P 500 stock market index, both strategies show less attractive financial performance characteristics. The defensive strategy 2 shows a lower volatility, which is due to the divestment in the case of the prediction of a falling index price. Nevertheless, the defensive strategy is not able to reduce the maximum drawdown.

4.3.2 Stock Price Direction Prediction Based on Content Sentiment Scores

This chapter presents the results of the random forest simulations using the 58 content sentiment scores as input. The target variables are the price direction of the next trading day and are, therefore, still the same. First, the fine-tuned hyperparameters and the corresponding cross-validated weighted precision scores are shown in Table 8 below:

Table 8: Fine-tuned hyperparameter and cross validated weighted precision scores based on content sentiment scores

Fine-tuning until 2010	Fine-tuning until 2013
n_estimators: 250	n_estimators: 1000
max_depth: 30	max_depth: 5
max_features: 5	max_features: 5
criterion: gini	criterion: gini
cv weighted precision: 0.526	cv weighted precision: 0.638
Fine-tuning until 2011	Fine-tuning until 2014
n_estimators: 250	n_estimators: 1000
max_depth: 5	max_depth: 3
max_features: 5	max_features: 58
criterion: entropy	criterion: entropy
cv weighted precision: 0.571	cv weighted precision: 0.589

Fine-tuning until 2012	Fine-tuning until 2015
n_estimators: 250	n_estimators: 250
max_depth: 5	max_depth: 3
max_features: 5	max_features: 30
criterion: entropy	criterion: entropy
cv weighted precision: 0.570	cv weighted precision: 0.544

Compared to the cross-validated weighted precision scores of the random forest classifications based on the title sentiment scores, those based on the content sentiment scores have a higher value. This is to be expected, however, as it has already been seen in chapter 4.2 that the content sentiment scores are generally better suited to predict the index stock market price developments. Similarly, the values of the parameters "max_depth" and "max_features" do not vary significantly in the fine-tuned models. This similar parameter selection of the fine-tuned models indicates that the models are generally stable and do not depend that much on the in-sample data. The out-of-sample confusion matrix is shown in Figure 20 below:

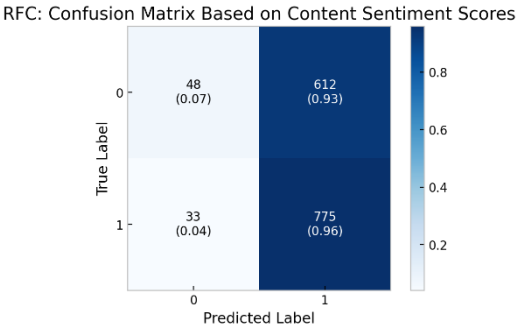

Figure 20: Out-of-sample confusion matrix of random forest classification based on content sentiment scores – own illustration:

$$accuracy = 0.561$$
$$weighted\ precision = 0.574$$
$$weighted\ recall = 0.561$$
$$weighted\ f1 - score = 0.447$$

According to the confusion matrix, it can be seen that still only few falling prices of the S&P 500 stock market index are predicted. Nevertheless, the number of false negative predictions could be reduced compared to the random forest classification based on title

sentiment scores. Consequently, the indexed out-of-sample performance should also be more attractive. The performance with the financial performance measures is illustrated below:

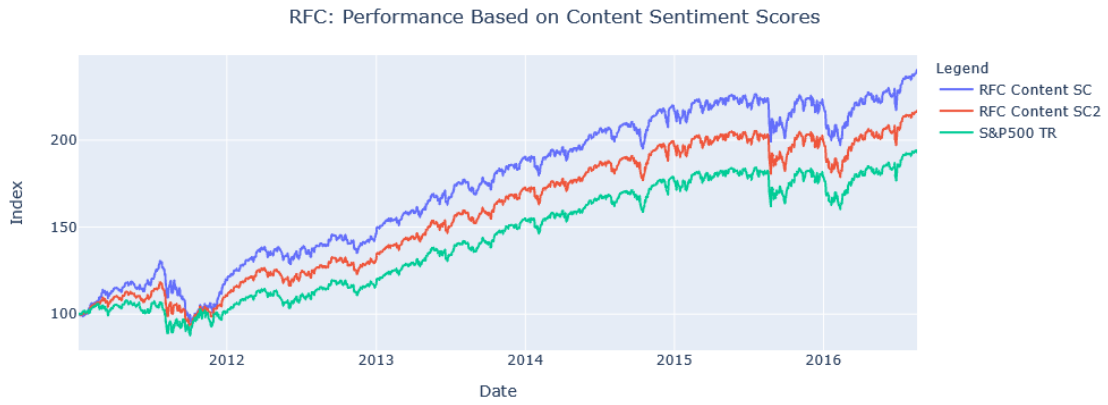

Figure 21: Out-of-sample performance random forest classification based on content sentiment scores – own illustration

Table 9: Performance measures of strategies based random forest classification on content sentiment scores from 01/01/2011 to 08/16/2016

	S&P 500	RFC Content SC	RFC Content SC 2
Return p.a.	12.045%	16.261%	14.226%
Volatility	15.108%	15.089%	14.442%
Return / Risk	0.797	1.078	0.985
Max. Drawdown	18.641%	28.187%	20.755%

According to the chart and table above, it can be seen that both strategies outperform the S&P 500 stock market index. Apart from the maximum drawdowns, both strategies show more attractive return and risk characteristics compared to the stock market index.

4.3.3 Stock Price Direction Prediction Based on Title and Content Sentiment Scores

Liu et al. (2018, p. 1,606) indicate in their previous work that the combination of the title and the content of the news can lead to a more robust performance. For this reason, this approach is tested by using the sentiment scores of the title and the content as input. This means that 116 sentiment scores are fed as input to the random forest models per day. Since the random forest models are now fed with twice the number of inputs, the values

for the parameters “max_depth” and “max_features” are extended for hyperparameter tuning. The values are listed in Table 10 below:

Table 10: Extended parameter grid for hyperparameter tuning

Parameter	Value
n_estimators	250, 500, 750, 1000
max_depth	1, 3, 5, 10, 20, 30, 40
max_features	5, 10, 15, 30, 40, 60, 80, 100, 116
criterion	gini, entropy
random_state	333
bootstrap	True

Since the trees have more features at their disposal, it makes sense to increase the parameter “max_features” since the trees can use a larger number of features during the fitting. However, this also implies that the algorithm should have a more complex structure so that it can process the additional information that can be explained by a higher number of features. For this reason, the parameter “max_depth” is also extended. The added values are marked in blue in the Table 10 above. After the parameter grid is extended, the fine-tuned hyperparameters are shown again:

Table 11: Fine-tuned hyperparameter and cross validated weighted precision scores based on title and content sentiment scores

Fine-tuning until 2010	Fine-tuning until 2013
n_estimators: 1000	n_estimators: 500
max_depth: 10	max_depth: 3
max_features: 2	max_features: 30
criterion: gini	criterion: gini
cv weighted precision: 0.587	cv weighted precision: 0.667
Fine-tuning until 2011	Fine-tuning until 2014
n_estimators: 750	n_estimators: 250
max_depth: 3	max_depth: 5
max_features: 40	max_features: 15
criterion: gini	criterion: entropy

cv weighted precision: 0.614	cv weighted precision: 0.566
Fine-tuning until 2012	Fine-tuning until 2015
n_estimators: 750	n_estimators: 500
max_depth: 3	max_depth: 10
max_features: 60	max_features: 2
criterion: entropy	criterion: entropy
cv weighted precision: 0.594	cv weighted precision: 0.535

According to the selected parameters from Table 11, it can be seen that the random forest models do not always exploit the extended parameters. Comparing the cross-validated weighted precision scores with those from the previous chapter, it also can be seen that they do not show a big difference. However, to provide a better overview of the classification, the out-of-sample confusion matrix is listed below:

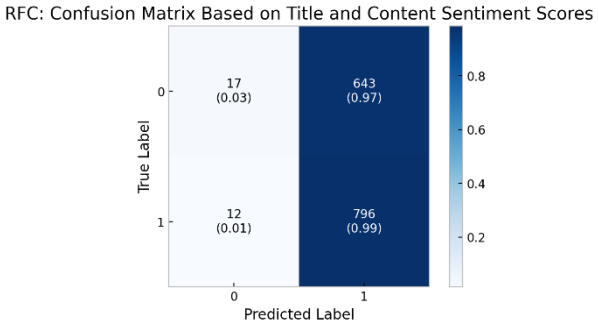

Figure 22: Out-of-sample confusion matrix of random forest classification based on title and content sentiment scores – own illustration

$$accuracy = 0.554$$
$$weighted\ precision = 0.568$$
$$weighted\ recall = 0.554$$
$$weighted\ f1 - score = 0.412$$

From the confusion matrix in this simulation only 29 times a negative return of the S&P 500 stock market index is predicted. Compared to the previous simulations, this is even much less. This leads to the expectation that the overall performance of the strategies does not differ much from the stock market index. However, to give an overview of the financial performance, the indexed performance is shown below:

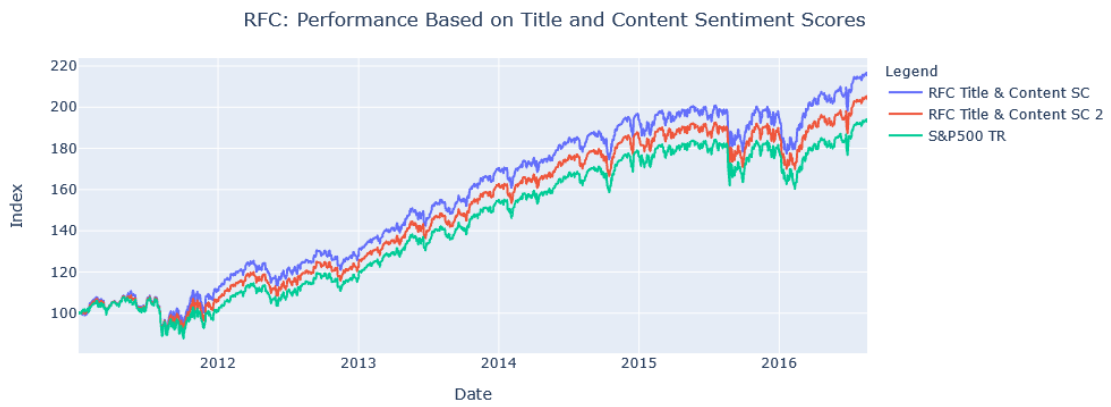

Figure 23: Out-of-sample performance random forest classification based on title and content sentiment scores – own illustration

As already expected, it can be seen from the above Figure 23 that the results of the two strategies do not differ much from the stock market index. Nevertheless, both strategies show a slight outperformance. Therefore, it can be concluded that both strategies have a more attractive return characteristic. To provide a better overview, the financial performance measures are again listed below:

Table 12: Performance measures of strategies based random forest classification on title and content sentiment scores from 01/01/2011 to 08/16.2016

	S&P 500	RFC Title & Content SC	RFC Title & Content SC2
Return p.a.	12.045%	14.143%	13.093%
Volatility	15.108%	15.096%	14.975%
Return / Risk	0.797	0.937	0.874
Max. Drawdown	18.641%	18.810%	17.587%

According to the table above, it can be seen that both strategies provide more attractive returns and risk characteristics compared to the S&P 500 stock market index. To further improve the model, it is combined with some past information of the S&P 500 stock market index. The results are shown in the next chapter.

4.3.4 Stock Price Direction Prediction Based on Title and Content Sentiment Scores and past Timeseries Information

Mohan, Mullapudi, Sammeta, Vijayvergia and Anastasiu (2019, pp. 205–208) show in their previous research that combining sentiment data with past timeseries information can improve the performance of a regression model. For this reason, in this chapter, the title as well as content sentiment scores are again combined with past timeseries information and given as input to the random forest algorithm. The moving average of the S&P 500 stock market index is used as past timeseries information. The author decided to calculate the moving average of the last 120, 60, 30, 15, and 5 days and to create a DataFrame from these five time series and the 116 daily sentiment scores. In order for the random forest algorithm to be able to process the time series better, they are standardized. Again, the extended grid is used for hyperparameter tuning.

Table 13: Fine-tuned hyperparameter and cross validated weighted precision scores based on title and content sentiment scores and moving averages

Fine-tuning until 2010	Fine-tuning until 2013
n_estimators: 1000	n_estimators: 500
max_depth: 10	max_depth: 3
max_features: 5	max_features: 80
criterion: entropy	criterion: gini
cv weighted precision: 0.545	cv weighted precision: 0.555
Fine-tuning until 2011	Fine-tuning until 2014
n_estimators: 750	n_estimators: 1000
max_depth: 10	max_depth: 10
max_features: 5	max_features: 60
criterion: entropy	criterion: gini
cv weighted precision: 0.545	cv weighted precision: 0.531
Fine-tuning until 2012	Fine-tuning until 2015
n_estimators: 750	n_estimators: 750
max_depth: 3	max_depth: 30
max_features: 116	max_features: 100
criterion: entropy	criterion: entropy
cv weighted precision: 0.565	cv weighted precision: 0.553

Observing the cross-validated weighted precision scores from Table 13 above, it can be seen that the values are stable over the entire period and have a value around 0.54. The confusion matrix is shown again:

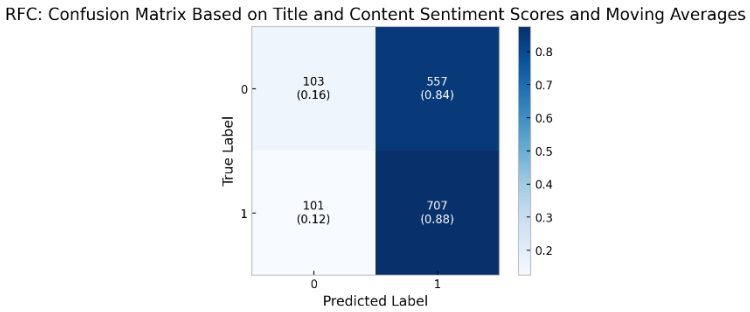

Figure 24: Out-of-sample confusion matrix of random forest classification based on title and content sentiment scores and moving averages – own illustration

$$accuracy = 0.552$$
$$weighted\ precision = 0.535$$
$$weighted\ recall = 0.552$$
$$weighted\ f1 - score = 0.483$$

In contrast to the previous simulations, negative returns are predicted more often in this simulation. Nevertheless, the number of false negative forecasts is about the same as the number of true negative forecasts. In the following, the indexed out-of-sample performance of the strategies and the S&P 500 stock market index as well as their financial performance measures are shown:

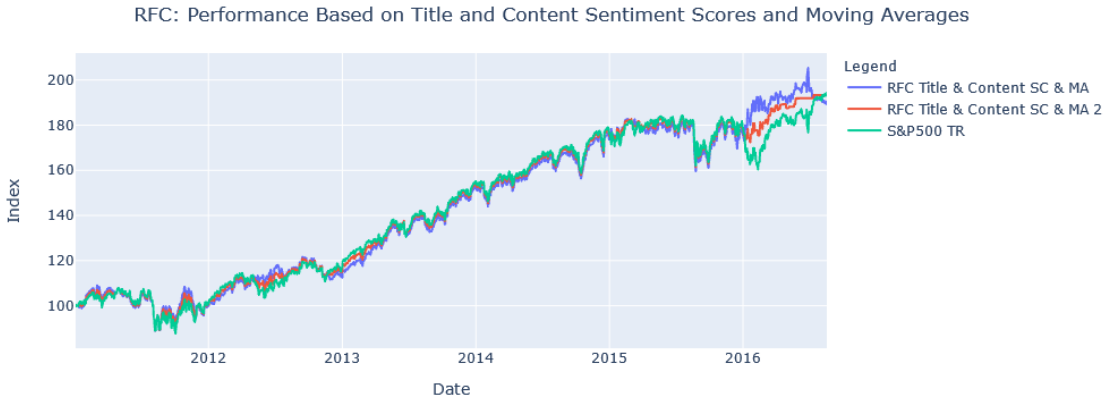

Figure 25: Out-of-sample performance random forest classification based on title and content sentiment scores and moving averages – own illustration

Table 14: Performance measures of strategies based random forest classification on title and content sentiment scores and moving averages from 01/01/2011 to 08/16/2016

	S&P 500	RFC Title & Content SC & MA	RFC Title & Content SC & MA 2
Return p.a.	12.045%	11.658%	11.990%
Volatility	15.108%	15.104%	14.154%
Return / Risk	0.797	0.772	0.847
Max. Drawdown	18.641%	18.406%	17.422%

According to Figure 25, it can be seen that both strategies do not differ much from the S&P 500 stock market index. Nevertheless, the random forest classification is able to predict the stock price downturn at the beginning of 2016 and shows more attractive properties in this period. The financial performance measures from table 14 show that the metrics of the three time series do not differ too much.

5 Conclusion and Outlook

In this chapter, the results of the empirical research are summarized, described and compared with each other. As already mentioned in chapter 4.3, the random forest classification models were optimized according to the performance measures weighted precision and weighted f1-score. Therefore, their results are listed in two tables in the next subsection. Another useful optimization would be to optimize the random forest models according to the performance measure recall or weighted recall, because this performance measure minimizes the number of false negative predictions. Since the random forest models minimized the number of false negative predictions by classifying all or almost all predictions into the label positive, their results are not listed in this chapter. However, the random forest models that were optimized according to the performance measure weighted f1-score are also indirectly optimized according to the recall score, since the recall score is used in the calculation of the weighted f1-score. Furthermore, the limitations of this master thesis as well as recommendations for further research and implications for practice will be pointed out in the next subsections.

5.1 Summary and Discussion of Empirical Results

In this subchapter, the empirical results presented in sections 4.2 and 4.3 are compared with each other and presented in a table below; the best variant for each performance measure is market blue.

Table 15: Summary of empirical results, random forest classification optimized by the weighted precision score

	Chapter / Variant Strategy											
		4.2 / Simple		4.3.1 / RFC Title		4.3.2 / RFC Content		4.3.3 / RFC Title & Content		4.3.4 / RFC Title, Content & MA	
Metric	**S&P 500**	**Content**	**Title**	**1[1]**	**2[2]**	**1[1]**	**2[2]**	**1[1]**	**2[2]**	**1[1]**	**2[2]**
Accuracy	-	-	-	0.535		0.561		0.554		0.552	
Weighted precision	-	-	-	0.487		0.574		0.568		0.535	
Weighted recall	-	-	-	0.535		0.561		0.554		0.552	
Weighted f1-score	-	-	-	0.437		0.447		0.412		0.483	
Return p.a.	12.05%	15.21%	-0.93%	7.67%	9.92%	16.26%	14.23%	14.14%	13.09%	11.66%	11.99%
Volatility	15.11%	15.10%	15.13%	15.12%	14.61%	15.09%	14.44%	15.10%	14.98%	15.10%	14.15%
Return / Risk	0.80	1.01	-0.06	0.51	0.68	1.08	0.99	0.94	0.87	0.77	0.85
Max. Drawdown	18.64%	20.60%	32.09%	25.52%	18.67%	28.19%	20.76%	18.81%	17.59%	18.41%	17.42%

[1] *Strategy 1 refers to the strategy of taking a short position on a prediction of a falling stock market index price of the S&P 500.*
[2] *Strategy 2 refers to the strategy of divesting on a prediction of a falling stock market index price of the S&P 500.*

Enclosed the results of the random forest simulations, optimized according to the weighted f1-score, are listed in Table 16; the best variant for each performance metric is marked in blue.

Table 16: Summary of empirical results, random forest classification optimized by the weighted f1-score

	Chapter / Variant Strategy										
		4.2 / Simple		4.3.1 / RFC Title		4.3.2 / RFC Content		4.3.3 / RFC Title & Content		4.3.4 / RFC Title, Content & MA	
Metric	**S&P 500**	**Content**	**Title**	**1[1]**	**2[2]**	**1[1]**	**2[2]**	**1[1]**	**2[2]**	**1[1]**	**2[2]**
Accuracy	-	-	-	0.534		0.531		0.540		0.524	
Weighted precision	-	-	-	0.509		0.506		0.517		0.513	
Weighted recall	-	-	-	0.534		0.531		0.540		0.524	
Weighted f1-score	-	-	-	0.482		0.486		0.486		0.512	
Return p.a.	12.05%	15.21%	-0.93%	4.20%	8.25%	3.71%	8.06%	7.38%	9.89%	5.45%	9.125%
Volatility	15.11%	15.10%	15.13%	15.13%	13.87%	15.12%	13.29%	15.11%	13.71%	15.12%	12.11%
Return / Risk	0.80	1.01	-0.06	0.28	0.60	0.25	0.61	0.49	0.72	0.361	0.75
Max. Drawdown	18.64%	20.60%	32.09%	-37.92%	-19.62%	23.06%	18.97%	24.79%	20.73%	21.35%	14.44%

[1] *Strategy 1 refers to the strategy of taking a short position on a prediction of a falling stock market index price of the S&P 500.*
[2] *Strategy 2 refers to the strategy of divesting on a prediction of a falling stock market index price of the S&P 500.*

Some insights can be gained by looking at the tables above. In case of an underperformance to the stock market index S&P 500, the defensive strategy 2 generally shows more attractive financial performance measures compared to the aggressive strategy 1. In the case of an outperformance, the opposite can be observed and the more aggressive strategy 1 proves to be more attractive in terms of returns. It can also be seen that the random forest simulations, which are optimized according to the weighted precision score, have more attractive properties than the random forest models, which are optimized according to the weighted f1-score. This can be attributed to the fact that the stock market index showed a strong upward trend in the observed period. Since the random forest models optimized by the weighted f1-score predict more often a falling price movement than the random forest models optimized by the weighted precision score, the chance is higher that this prediction is wrong due to the strong upward trend.

When optimizing random forest models according to the weighted precision score, the random forest classification model based on content sentiment scores exhibits the most attractive classification metrics and return properties. An extension by combining the sentiment scores from the titles and the content with past time series information such as the moving average does not lead to a significant improvement of the predictions.

Among the weighted f1-score optimized models, the random forest classification model based on the title and content sentiment scores has the most attractive classification metrics and, compared to the other random forest models, also the most attractive financial performance measures. Also in this scenario, an extension by adding some moving averages as input does not improve the forecasts.

Another interesting finding is that the simple models using the average of the 58 sentiment scores from the previous day as a prediction for the next trading day's return generally work well when using the content sentiment scores.

In conclusion, based on the above results, the research question cannot be answered positively. However, the fact that some strategies outperform the S&P 500 in the observed period suggests that the use of sentiment scores calculated by state-of-the-art NLP methods can improve the performance of an investment strategy.

5.2 Limitations of this Study

This master thesis sets a good foundation for further research in the field of stock prediction using state-of-the-art NLP methods such as BERT. Although the strategies obtained can partially outperform the S&P 500 stock market index, some approaches can be further investigated. In particular, the use of more specific news can be used targeted to the stock market index or to the company to be analyzed. Similarly, because of the limited resources, only 58 different news items per day were used in this master thesis. The use of more news articles per day could increase the performance of the machine learning models.

5.3 Recommendations for Further Research

Recommendations for further research can be made on various levels. From a data perspective, the author proposes to use news from all companies included in the stock market index. Thus, a data set can be created which has a column for each company. In this way, the company-specific sentiment scores can be considered as features. From the methods standpoint, further machine learning models or recurrent deep learning models, which have a memory due to their internal state, can be used. Additionally, other past time series information such as a moving value at risk or a moving expected shortfall of the last days can be added as input for improving the quality of the predictions.

5.4 Implications for Practice

Based on the findings of this master thesis, it can be concluded that the sentiment scores calculated from state-of-the-art NLP methods can be used for stock price forecasting. In practice, there are different scenarios where the models shown in this master thesis can be used. Since the sentiment scores should also reflect the market sentiment, the scores can be used for a risk-based approach, whereby a combination with other proven indicators is recommended. Another possibility arises from the use of the NLP methods shown in relation to sustainable investing. For example, reports can be used to find out whether the company takes ethical or ecological aspects into account and, accordingly, by automating the reading of the reports, a pre-selection of sustainable and less sustainable companies can be made.

6 List of References

Alammar, J., (2018). The Illustrated Transformer. Retrieved from https://jalammar.github.io/illustrated-transformer/

Araci, D., FinBERT: Financial Sentiment Analysis with Pre-Trained Language Models. University of Amsterdam. Retrieved from https://arxiv.org/pdf/1908.10063.pdf

Berrar, D. (2018). Cross-Validation. Retrieved from https://www.researchgate.net/publication/324701535_Cross-Validation

Bhatt, S. (2018). Reinforcement Learning 101 – Learn the essentials of Reinforcement Learning. Retrieved from https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292

Ciacaglia, C. (2019). How Transformers Work. Retrieved from https://towardsdatascience.com/transformers-141e32e69591

Devin, J., Chang, M., Lee, L., Toutanova, K., (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1. Minnesota, 4171-4186

Ding, X., Zhang, Y., Liu, T., Duan, J., (2014). Using Structured Events to Predict Stock Price Movement: An Empirical Investigation. Retrieved from http://emnlp2014.org/papers/pdf/EMNLP2014148.pdf

Fama, E. (1970). Efficient Capital Markets: A Review of Theory and Empirical Work. The Journal of Finance, 25(2), p. 383-417

Fazlija, B., (2020). Class materials from the Machine Learning course. Fall semester 2020/2021. Winterthur: University of Applied Science Zurich, School of Management and Law

Frey, M., Ruckstuhl, A. & Sick, B. (2018). Vorlesungsskript Statistisches Data Mining. Fall semester 2018/2019. Winterthur: University of Applied Science Zurich, Technikum.

Géron, A., (2017). Hands-On Machine Learning with Scikit-Learn and TensorFlow. First Edition. Sebastopol: O'Reilly Media, Inc.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction. Second Edition. New York: Springer.

Liddy, E.D. (2001). Natural Language Processing. Second Edition. In Encyclopedia of Library and Information Science. NY. Marcel Decker, Inc.

Liu, Q., Cheng, X., Su, S., Zhu, S., (2018). Hierarchical Complementary Attention network for Predicting Stock Price Movements with News. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18). Association for Computing Machinery, New York, NY, USA, 1603–1606.

Malo, P., Sinha, A., Takala, P., Korhonen, P. & Wallenius, J. (2013): Good debt or bad debt: Detecting semantic orientations in economic texts. Journal of the American Society for Information Science and Technology.

Mohan, S., Mullapudi, S., Sammeta, S., Vijayvergia, P., & Anastasiu, C., (2019). Stock Price Prediction Using News Sentiment Analysis. 2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService), p. 205-208

Nassirtoussi, A.K., Aghabozorgi, S., Wah, T.Y., & Ngo, D. (2014). Text mining for market prediction: A systematic review. Expert Syst. Appl., 41, p. 7653-7670.

Nguyen, C. N. & Zeigermann, O. (2018). Machine Learning kurz & gut. First Edition. Heidelberg: dpunkt.verlag GmbH

Pandas Developers. (2021). About pandas. Retrieved from https://pandas.pydata.org/about/index.html

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research. Volume 12

Posth, J., (2019). Class materials from the Investments course. Fall semester 2019/2020. Winterthur: University of Applied Science Zurich, School of Management and Law

Python Software Foundation. (2021). What's new In Python 3.9. Retrieved from https://docs.python.org/3/whatsnew/3.9.html

Remy, P., Ding, X., (2015). Financial News Dataset from Bloomberg and Reuters. Retrieved from https://github.com/philipperemy/financial-news-dataset

Richardson, L., (2021). Beautiful Soup. Retrieved from https://pypi.org/project/beautifulsoup4/

S&P Dow Jones Indices. (2021). S&P U.S. Indices Methodology. Retrieved from https://www.spglobal.com/spdji/en/indices/equity/sp-500/#

Sarkar, D., Bali, R. & Ghosh, T. (2018). Hands-On Transfer Learning with Python: Implement advanced deep learning and neural network models using TensorFlow and Keras. Packt Publishing.

Scikit-learn Developers (2021). About us. Retrieved from https://scikit-learn.org/stable/about.html

Shmueli, B. (2019). Multi-Class Metrics Made Simple, Part II: the F1-Score. Retrieved from https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1

The Hugging Face Team, (2021). Transformers. Retrieved from https://huggingface.co/transformers/

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., & Polosukhin, I., (2017). Attention Is All You Need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010.