

Master Thesis

Zurich University of Applied Sciences

School of Management and Law

Natural Language Processing in Finance:
Analysis of Sentiment and Complexity of News and Earnings Reports of Swiss SMEs and their Relevance for
Stock Returns

Student:

Alexander Schlaubitz



Submitted to:

Dr. Bledar Fazlija

ZHAW School of Management and Law

Technoparkstrasse 2, 8400 Winterthur

Winterthur, 30.06.2021

Management Summary

Natural Language Processing (NLP) can be used for the purpose of achieving human-like language interpretation for a range of tasks such as analyzing stock related news and earnings reports. This allows financial professionals to gain a quicker and deeper insight into market information via Machine Learning (ML) algorithms. In literature, significant connections between textual sentiment and stock prices for large cap companies in the English-speaking world are identified. This paper examines if such effects also pertain to Swiss companies that create their reporting in the German language. Accordingly, the interplay of sentiment and complexity of news and earnings reports for publicly listed Swiss companies in the small and mid cap segment are analyzed, together with their potential relevance for stock returns. Additionally, the Transformer algorithm is introduced, and its performance is compared to older ML algorithms.

A literature review showed that the Transformer model was able to set new state-of-the-art performance in language modeling and thus outperform previous language-oriented algorithms. Accordingly, for the sentiment analysis of Swiss earnings report and news articles, a DistilBERT model was trained and fine-tuned using a financial phrase databank. The model, which made use of Transfer Learning, showed sentiment prediction accuracies of 90 percent. Text complexity was analyzed using the well-established Flesch Score and Wiener Sachtextformel. The results for complexity show that both, Swiss earnings reports and financial news articles, are generally difficult to understand and require the reading level of university students. Sentences with either negative or positive sentiment are both equally complex and there is not any noticeable difference in complexity between news and earnings reports, at least for human readers. For sentiment it was found that only a small number of earnings reports contains negative sentiment on an aggregated level, even when firms reported lower earnings. Meanwhile, news articles provide a more balanced data source for sentiment. Significant linear connections in terms of regression analysis of the predictive ability of text sentiment on future stock returns were found for only 4 of total 15 analyzed companies, indicating that sentiment alone is a bad linear predictor for future performance. However, a nonlinear classification model found an increase of 8 percent in accuracy when including sentiment data together with historical stock data, compared to only using historical stock data for the forecast of stock return development.

Table of Contents

1	Introduction.....	1
1.1	Thesis Objective.....	2
1.2	Demarcation	2
1.3	Relevance	3
1.4	Structure	3
2	Natural Language Processing	5
2.1	Sentiment Analysis.....	5
2.1.1	Data collection	6
2.1.2	Text pre-processing.....	7
2.1.3	Text classification	11
2.1.4	Analysis of Results	18
2.2	The Transformer.....	21
2.2.1	The Original Transformer Network	21
2.2.2	Bidirectional Encoder Representations from Transformers (BERT).....	28
2.3	Complexity.....	32
2.3.1	Flesch reading-ease score	33
2.3.2	Wiener Sachtextformel	35
3	Results in Literature.....	36
4	Model Methodology.....	40
4.1	Complexity Model.....	40
4.1.1	Implementation of the Flesch Score	40
4.1.2	Implementation of the Wiener Sachtextformel.....	40
4.2	Sentiment Model	41
4.2.1	Data Gathering.....	42

Analysis of Sentiment and Complexity of News and Earnings Reports of Swiss SMEs and their Relevance for Stock Returns

4.2.2	Data pre-processing	43
4.2.3	Model architecture	45
4.2.4	Model evaluation	49
5	Analysis.....	53
5.1	Data gathering	53
5.2	Data pre-processing.....	56
5.3	Evaluation.....	57
5.3.1	Complexity.....	57
5.3.2	Sentiment	63
5.4	Discussion	78
6	Conclusion	82
7	References.....	85
8	Appendix.....	82
8.1	Complexity Tables	82
8.2	Regression Results	85

List of Tables

Table 1	Overview of pre-processing studies (own creation)	10
Table 2	A 2x2 confusion matrix (own creation).....	18
Table 3	Flesch reading-ease score assessment as per (Flesch, 2016).....	34
Table 4	Available datasets from the Financial phrase databank as per (Malo, Sinha, Korhonen, J., & Takala, 2014)	42
Table 5	Pre-processing performance comparison (own creation)	44
Table 6	Selected companies.....	54
Table 7	Overview of gathered news and earnings reports	55
Table 8	Differences in complexity between small and mid caps (own creation)	59

Table 9 Differences in complexity between earnings reports and news articles (own creation)..	60
Table 10 Development of complexity over the past 5 years (own creation)	60
Table 11 Comparison of complexity per sentiment class	61
Table 12 Aggregated sentiment per company over 5 years	64
Table 13 Sentiment over time (own creation).....	65
Table 14 Overview of sentiment distribution	66
Table 15 Dummy notation (own creation).....	67
Table 16 Correlation of sentiment and stock performance for mid caps (own creation)	68
Table 17 Correlation of sentiment and stock performance for small caps (own creation).....	69
Table 18 Regression results for BELL at T+2	70
Table 19 Mid cap complexity overview	82
Table 20 Small cap complexity overview 1 out of 2	83
Table 21 Small cap complexity overview 2 out of 2	84
Table 22 BUCHN regression T+1	85
Table 23 BELL regression T+0	86
Table 24 BELL regression T-1	87
Table 25 BELL regression T+2	88
Table 26 BELL regression T+3	89
Table 27 BUCN regression T-1	90
Table 28 STMN regression T+0	91
Table 29 STMN regression T+3	92
Table 30 ZEHN regression T+0.....	93
Table 31 ZEHN regression T+2.....	94

List of Figures

Figure 1 An overview of sentiment classification methods as per (Abdullah, Manjula, & Lakshman Naik, 2019, p. 150).....	11
Figure 2 A LSTM model as per (Gao, Chai, & Liu, 2017).....	17
Figure 3 The RNN Encoder-Decoder (Cho, et al., 2014, p. 1725)	22
Figure 4 Softmax function (own creation).....	24

Figure 5 The Scaled Dot-Product Attention (left) and the Multi-Head Attention consisting of several attention layers running in parallel (right) (Vaswani, et al., 2017, p. 4) 24

Figure 6 The Transformer architecture (Vaswani, et al., 2017, p. 3)..... 25

Figure 7 ReLU function (own creation)..... 26

Figure 8 Predicting if the second sentence is in context to the first one (Devlin & Chang, 2018)29

Figure 9 BERT input representation shown as a sum of token, segment and position embeddings (Devlin, Chang, Lee, & Toutanova, 2019, p. 4174) 30

Figure 10 The pre-training and fine-tuning procedures for BERT (Devlin, Chang, Lee, & Toutanova, 2019, p. 4173) 31

Figure 11 Code required for the textstat implementation of the Flesch score (own creation)..... 40

Figure 12 A code snippet of the implementation for the Wiener Sachtextformel (own creation) 41

Figure 13 German BERT performance (deepset.ai, 2019) 45

Figure 14 The DistilBERT model (own creation) 46

Figure 15 Most important parameters for a high MCC (own creation) 47

Figure 16 Connections between learning rate, number of epochs and MCC (own creation)..... 48

Figure 17 A constant decrease of learning rates for the first 23 models (own creation) 48

Figure 18 Cross validation results (own creation) 49

Figure 19 Results on the testing dataset (own creation) 49

Figure 20 Classification report for precision, recall and F1-Score (own creation) 50

Figure 21 The confusion matrix for the model with labels 0 : negative, 1 : neutral, 2 : positive (own creation) 50

Figure 22 Using the trained model to classify four sentences (own creation)..... 52

Figure 23 The difference made in prediction in case of capitalization (own creation) 57

Figure 24 Distribution of Flesch scores going from hard to easy (left to right) in terms of understanding (own creation) 58

Figure 25 Distribution of Wiener scores going from easy to hard (left to right) in terms of understanding (own creation) 59

Figure 26 LSTM configuration (own creation) 73

Figure 27 Neural Network architecture (own creation)..... 75

Figure 28 Prediction range of the model with sentiment (left) and without (right) (own creation) 76

Figure 29 Confusion matrix for the model with sentiment (left) and without (right) (own creation) 77

Figure 30 Precision, Recall and F1-score for the model with sentiment (left) and without (right) (own creation) 77

List of Equations

(1) Precision 19

(2) Recall..... 19

(3) F1 Score..... 19

(4) MCC 20

(5) Update step of the RNN 21

(6) Update step of the hidden state for the encoder 21

(7) Calculation of the context summary..... 23

(8) Attention..... 23

(9) Softmax 24

(10) Softmax for probability of label c 31

(11) Flesch Score 34

(12) Flesch Score German 34

(13) Wiener Sachtextformel..... 35

(14) Regressions for sentiment 63

(15) Sample Size..... 67

(16) Standardization..... 73

List of Abbreviations

ANN	Artificial Neural network
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolutional Neural Network
EBIT	Earnings Before Interest and Taxes
EDGAR	Electronic Data Gathering, Analysis, and Retrieval System
FN	False Negative
FP	False Positive
GRU	Gated Recurrent Unit
HTML	Hypertext Markup Language
LSTM	Long short-term memory Network
MCC	Matthews correlation coefficient
MNB	Multinomial Naive Bayes
ML	Machine Learning
NLP	Natural Language Processing
POS	Part-of-speech tagging
RNN	Recurrent Neural Network
SEC	Securities and Exchange Commission of the United States of America
SME	Small and Mid sized companies, in this paper also referring to small and mid caps
SVM	Support Vector Machine
TN	True Negative
TP	True Positive

1 Introduction

Machine Learning (ML) is the most cited trend in terms of driving change in the roles of investment professionals (CFA Institute, 2019). ML, as one of the branches of Artificial Intelligence, uses a range of statistical models based, which use sample data, also known as training data, in order to make predictions or decisions (Koza, Bennett III, Andre, & Keane, 1996, p. 151). Finance professionals have taken to ML techniques as early as 1994, when they first conducted research on forecasting stock prices (Refenes, Zapranis, & Franics, 1994, pp. 357 - 388). Since then many more applications for ML algorithms in finance have been found. Examples include fraud detection, automation of trading activities, robot advisors or ML as a method to gain insights from large amounts of unstructured data, such as news. As ML tends to be more accurate in drawing insights and making predictions when large volumes of data are fed into its system, the professionals of the financial industry can benefit from this, given by the fact that they encounter enormous volumes of data relating to daily transactions, payments, or customers. Some of the above-mentioned examples rely on Natural Language Processing (NLP) to understand human inputs, either via text or speech.

NLP is defined as an area of research that explores how computers can understand and manipulate natural language text or speech to do useful things (Chowdhury, 2005, p. 51). One of the larger benefits of this technology is that it can be applied to analyze and screen new information, for example from financial news, in a fast manner. Market participants are continuously monitoring these financial and economic news to adapt their opinion on the market and build their financial positions accordingly. Based on the efficient market hypothesis, all past information is reflected in stock prices and new information is instantaneously absorbed in determining future stock prices (Fama, 1970). Hence, prompt extraction of trading signals from news is very important for investment decision-making by traders, portfolio managers and investors. Sentiment analysis models can provide an efficient method for extracting potentially actionable signals from news (Mishev K. , Gjorgjevikj, Vodenska, Chitkushev, & Trajanov, 2020, pp. 1 - 2). Sentiment analysis is defined as the process of computationally categorizing text-based opinions, mainly to determine whether the author's attitude towards a particular topic is positive, neutral or negative (Oxford Lexico, 2021). Recent developments in NLP introduced increasingly complex algorithms that apply deep learning for seemingly better results (Yang, Yang, Dyer, He, &

Smola, 2016, pp. 1480 - 1489). Nonetheless, financial sentiment analysis remains challenging due to domain-specific language and unavailability of large labeled datasets (Mishev K. , Gjorgjevikj, Vodenska, Chitkushev, & Trajanov, 2020, pp. 1 - 2). These problems are also encountered in the Swiss market, especially for companies with small or mid-sized market capitalization (in short small or mid cap), which are denoted as SMEs for this paper.

1.1 Thesis Objective

The aim of this master thesis is to introduce modern NLP methods and how they can be applied to financial text-based data to evaluate their sentiment and complexity. In this process the expected benefits of using a more modern technique, such as Bidirectional Encoder Representations from Transformers (BERT), over older techniques, such as dictionary-based approaches, shall be highlighted. Accordingly, as a first step, the Transformer deep learning network is introduced as a state-of-the-art algorithm, and its performance is compared to older NLP methods. In a second step, a sentiment analysis model is created using the Python programming language, that aims to categorize German sentences from Swiss news and company earnings reports into an either positive, neutral or negative category. Additionally, these texts shall also be analyzed on their complexity. In the last step, the results are analyzed in the context of stock prices, to find potential connections between these variables and stock price movements. The thesis therefore aims to answer the following points:

- 1) What are Transformer models, and do they outperform other NLP techniques?
- 2) Does text complexity have any effect on text sentiment?
- 3) Are there any connections between sentiment, complexity, and news and earnings reports for Swiss small- and mid caps and their corresponding stock price movements?

1.2 Demarcation

This paper exclusively focuses on selected publicly listed Swiss companies that create their financial reporting and media announcements in the German language and fall either into the small cap or mid cap segment. Companies that create their financial reporting in multiple languages are not excluded, as long as one of the reporting languages is German and their data is fully accessible to the public. The review period for this paper is 5 years, meaning that news and earnings reports before the 1st of January 2016 will not be considered. Basic knowledge about ML is as-

sumed, and therefore concepts such as Supervised Learning or Neural Networks are not described in explicit detail.

1.3 Relevance

Natural language processing helps computers communicate with humans in their own language and scales other language-related tasks. For example, NLP makes it possible for computers to read text, hear speech, interpret it, measure sentiment and determine which parts are important. The main focus of research within NLP lies on the English language. Hence, an implementation and performance analysis on German text adds to the current state of knowledge about NLPs efficiency in sentiment classification for other languages. Potential connections between textual sentiment or complexity with a company's stock price or other financial performance indicators, such as earnings or analyst ratings, could lead to forecasts about future stock performance. The most basic motivation behind forecasting future stock prices is monetary gain. Technologies, such as NLP, may help to further broaden the understanding of textual sources and their effect on the financial market. Any accurate and reliable forecasting method for financial data can have immense impacts on trading strategies and should therefore be studied in detail. While research for the effects of text sentiment on American and English stocks exists, no such research has been conducted for Swiss stocks, at least not to the knowledge of the author. Furthermore, most research in this field is conducted on large cap companies, while small and mid cap companies are rarely analyzed.

1.4 Structure

This master thesis is structured into five parts as follows:

Chapter two provides an introduction to NLP and compares newer, transformer-based models with older, dictionary-based models. Additionally, their performance for NLP tasks, especially for sentiment analysis is evaluated. The valuation of text complexity is explained using two established formulas that focus on word and sentence structures.

Chapter three analyzes and compares current results from literature regarding the relationship between sentiment and complexity from news articles or earnings reports and stock price movements.

Chapter four applies the theory from chapter two to describe the creation of a sentiment analysis model using an adapted Transformer model. Additionally, the current methods to analyze text complexity are implemented via the Python programming language.

Chapter five closely analyzes the potential interactions of sentiment, complexity and stock price movements and tries to draw conclusions in terms of importance of these variables for the Swiss stock market. The results are analyzed, and the findings are summarized in **chapter six**.

2 Natural Language Processing

As defined in the introduction to this paper, NLP consists of a broad range of techniques for analyzing and representing texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications (Liddy, 2001, p. 2). While the financial world can benefit from many aspects of NLP, this paper solely focusses on analyzing the sentiment and the complexity of finance specific texts. Accordingly, the different methods for sentiment analysis and determination of text complexity are introduced below. The Transformer model, as one of the most promising technologies in the field of sentiment analysis and NLP in general, is highlighted and explained in detail, which is why it is presented in its own sub-chapter.

2.1 Sentiment Analysis

The core idea behind any form of sentiment analysis lies in the categorization of text into a certain emotional category, such as positive, neutral or negative (Agarwal, Xie, Vosha, Rambow, & Passonneau, 2011, p. 30). As an extension of this principle, text can also be analyzed on its tone. Simply put, this means that the words or sentences of a text are categorized into more specialized categories, such as Negative, Positive, Uncertainty, Litigious, Strong Modal, Weak Modal or Constraining as suggested by Loughran and McDonald (2011, p. 37). Following the explanations of D'Andrea, Ferri, Grifoni & Guzzo (2015, pp. 26 - 33) the complex process of sentiment analysis can be broken down into five different steps. These steps are:

- **Data collection:** The first step of sentiment analysis consists of collecting data from user generated content, which can be contained in blogs, news articles, social networks or other similar sources. Such textual data is disorganized and expressed in different ways by using different languages, vocabularies, slangs and context. Hence, manual analysis is almost impossible. Therefore, text analytics and natural language processing are used to extract and classify the text's sentiment.
- **Text preparation:** Pre-processing the data is the process of cleaning and preparing the text for classification. Online texts usually contain lots of noise and uninformative parts such as HTML tags, scripts and advertisements (Haddi, Liu, & Shi, 2013, p. 27). In addition, on a word level, many words in the text do not have an impact on the general senti-

mental orientation of it. Accordingly, non-textual content and content that is irrelevant for the analysis are identified and eliminated (D'Andrea, Ferri, Grifoni, & Guzzo, 2015, pp. 26 - 33). Depending on the type of text and model, this could include numbers or dates in any form or the removal of punctuation, such as exclamation marks.

- **Sentiment detection:** The extracted and prepared sentences of the reviews and opinions are examined. Sentences with subjective expressions (opinions, beliefs and views) are retained and sentences with objective communication (facts, factual information) are discarded or classified as such, depending on if the inclusion of neutral sentiment is expedient.
- **Sentiment classification:** In this step, the subjective sentences or words are classified into the user-defined categories, such as positive, negative, good, bad, like, dislike or similar. The steps of sentiment detection and sentiment classification usually go hand in hand, meaning that they are often performed in one single step.
- **Presentation of output:** The main objective of sentiment analysis is to convert unstructured text into meaningful information. Therefore, when the analysis is finished, the text results are displayed on graphs like pie charts, bar charts and line graphs.

The above mentioned five steps, as suggested by D'Andrea, Ferri, Grifoni & Guzzo (2015, pp. 26 - 33), provide a rough overview of a general sentiment analysis process. However, especially the steps of text preparation (pre-processing) and sentiment classification usually apply a plethora of different methods and algorithms. Accordingly, they can be explained on in more detail.

2.1.1 Data collection

Adding onto the initial explanations of D'Andrea, Ferri, Grifoni & Guzzo (2015, pp. 26 - 33), data collection as the first step in sentiment analysis, or any NLP process in general, is usually also the most labor intensive. Any good model requires an adequate amount of base data to either train or evaluate on. To simplify this process, Application Programming Interfaces (APIs) can be used which allow users to define what they would like to download and then conducts that process for them (Abdullah, Manjula, & Lakshman Naik, 2019, p. 155). A useful example of a valuable API is the SEC EDGAR filings API (U.S. Securities and Exchange Commission (SEC), 2021). The SEC is the Securities and Exchange Commission of the United States of America and EDGAR is its Electronic Data Gathering, Analysis, and Retrieval System, where American com-

panies publish their financial statements. This API allows the user to access, download and manipulate all filed financial statements of publicly listed American companies between 1993 and now. Switzerland does not have any centralized data storage for financial statements, at least not one that is accessible by the public and therefore also no helpful APIs exist in that regard. When no APIs exists for the data of interest, one can consider programming a web scraping tool. Web scraping is the usage of technological tools to automatically extract, organize and analyze data from the web (Krotov & Tennyson, 2018, p. 61). According to Mitchell (2018, p. 1) this process never involves any API and thus is most commonly accomplished by writing an automated program that queries a web server, requests data, usually in the form of HTML, and then parses that data to extract the required information. However, for web scraping to be efficient it requires a fixed and stable web site as a target from where it can download data. If that is not the case or the data is distributed across multiple differently structured websites, web scraping provides little use, or rather the time cost of programming such a complex web scraper would be unproportionally high. Hence, as a last stand, some data must simply be gathered by hand.

2.1.2 Text pre-processing

The core idea behind text pre-processing is that if the data is properly cleaned, the resulting reduction in noise, represented by unimportant text pieces or words, should help improve the performance of the classifier and also increase it's working speed (Haddi, Liu, & Shi, 2013, p. 27). A fast classification process is a must for real-time sentiment analysis, as it would be used for directly classifying financial news at the very moment they are published. Therefore, pre-processing is deemed a vital process for most sentiment analysis models (Angiani, et al., 2016, p. 8). For this purpose, noise can be more specifically defined as every bit of textual information that either obscures the core sentiment of a sentence, or simply introduces non-informational details, such as punctuations. However, the exact benefit of every pre-processing step cannot be generalized as this largely depends on the dataset and application of each unique sentiment model. Therefore, the most common pre-processing steps and techniques are introduced below, to give a basic overview of the most relevant methods:

- **White space removal:** Removing white space is the most basic step and means that any abundant separation between words, sentences or paragraphs is removed.

- **Lower casing:** This eliminates capitalization from the text, meaning that all letters are turned into lower cases. Accordingly, instead of identifying “Finance” and “finance” as two different words, the resulting lower-case “finance” will represent both, thus reducing text variability.
- **Punctuation filtering:** Filtering out punctuation such as question- or exclamations marks again reduces text complexity, especially since certain symbols have no use in normal language, for example “/” or “*”. However, an argument can be made to keep question- and exclamations marks in the text dataset, as they might provide additional information (i.e. an exclamation mark could provide a stronger positive sentiment when comparing the sentences “I am happy.” with “I am happy!”). The exact punctuation filtering therefore depends on the algorithm used and whether it can pick up such details. Note that punctuation at the end of a sentence is usually kept, as this is a clear indication for the end of a sentence.
- **Stop word removal:** A stop word is defined as a frequently appearing word that provides no useful information (Krouska, Troussas, & Virvou, 2016, p. 4). Eliminating such words therefore again reduces the complexity with no loss of information content. Examples for stopwords in the English language are “the, a, it, its, as, at, to, for” etc. Note that a different language than English will therefore also have its own unique stopwords. Luckily, many researchers have already gathered thousands of such words in multiple languages to facilitate this process (Oppenlaender, 2021).
- **Normalization via Replacement, Stemming or Lemmatization:** Normalization is used to make text more readable, not for humans but for the algorithms that try to classify the text. While readability per-se is not that important for an algorithm, normalization also introduces better processability. Normalization via replacement means that outliers (words that are written incorrectly, in all capital letters or similar) are completely replaced with an alternative version. An example can be made with tweets. Twitter users often use abbreviations, emoticons, hashtags and URLs. These can be either completely removed or in the place of emoticons, they can be replaced with tags that express their sentiment (i.e. :) → smile-happy) (Angiani, et al., 2016, p. 5). Normalization via Stemming is the process of reducing words to their word stem, base or root form. This reduces text entropy as

the amount of words in the dataset such as “greatly”, “greatest”, and “greater” are all transformed into the stem of the word, which is “great” (Angiani, et al., 2016, p. 8). Normalization via Lemmatization is very similar to stemming in that it transforms words to their lemma. The lemma is the lexical “headword” of a given word form (Kanis & Skorkovska, 2010, p. 94). Lemmatization therefore tries to identify the meaning of a word in a sentence. As example, the word "better" has "good" as its lemma. Stemming would not change this word, while Lemmatization would. However, for the word "fisher" both, Stemming and Lemmatization, would transform this into its stem "fish".

- **Tokenization:** In order to use raw textual data for predictive modeling, the text should be parsed and broken up – this process is called tokenization. The mechanism involves splitting or fragmenting the text into its smallest possible form, called tokens (Rai & Borah, 2021, p. 93). These tokens can either be sentences, words, characters, or sub-words. However, text is most commonly already split into a single sentence format. Hence, tokenization is broadly classified into three types – word, character, and sub-word (n-gram characters) tokenization. In n-grams one can select how many (n) words should represent a token. For example, the German sentence *"Es wurde ein Rekordgewinn erzielt"*, with n=2 n-grams, also called bigrams, would be [*"Es wurde"*, *"wurde ein"*, *"ein Rekordgewinn"*, *"Rekordgewinn erzielt"*]. Based on that logic, tokenization where each word is a token, would be the same as using n=1 n-grams.

Having introduced the most commonly applied forms of text pre-processing it is also of interest how large the effect of such implementations is. As previously mentioned, there is no clear guarantee for specific performance increases for every single method, however analyzing comparisons from literature can help in gauging the usefulness of certain methods.

Multiple researchers have analyzed the effect of pre-processing on accuracy and computation speed of text pre-processing. Angiani et. al. (2016, pp. 8 - 10) have conducted sentiment analysis on twitter data using a Multinomial Naïve Bayes (MNB) algorithm, a probabilistic classifier which works on the basis of the Bayes Theorem. They have introduced multiple steps of pre-processing, ranging from no pre-processing, to basic cleaning (i.e. white space and outlier removal), to stemming and stopword removal. As a result, it shows that pre-processed models have better performances, measured by an average accuracy increase of around two percent, compared

to models working with unprocessed data. The only method that did not introduce any benefits was a dictionary, that automatically identified and replaced misspelled words (Angiani, et al., 2016, p. 10). Another study conducted by Mat Zin, Mustapha, Murad & Sharef (2017, pp. 1 - 8) analyzed a set of movie reviews from IMDB using Support Vector Machine (SVM) algorithms. They have split their pre-processing into three tiers, ranging from removing stop words (tier 1), removing stop words and meaningless characters such as symbols (@ or #) (tier 2) and for tier 3 they removed stop words, meaningless characters, numbers and all words that have less than three characters. On average, the performance, measured by F1-Scores and accuracy, oscillated for both scores between 0.81 and 0.83, where tier 2 and tier 3 have shown better performances than tier 1 pre-processing. A third study, conducted by Alam and Yao (2018, pp. 319 – 335), also analyzed data from twitter. They compared the results of an MNB algorithm, an SVM algorithm and a Maximum Entropy algorithm when applied on an unprocessed and pre-processed dataset. The pre-processed dataset was changed by applying stop word removal, stemming and n-gram tokenization. As a result, the accuracy of the SVM was improved to 81.63% from 81.09%. Similarly, the accuracy of MNB was improved to 91.81% from 83.69%. There was no change in the accuracy of the Maximum Entropy algorithm.

To summarize, the results of all three research studies agree that pre-processing leads to better model performance, although to a varying degree and depending on the algorithm used. All three studies also highlighted the increase in computational speed, given by lower text size, as pre-processing removes words and thus also text complexity.

Table 1 Overview of pre-processing studies (own creation)

Research conducted by	Average performance increase from pre-processing	Algorithm used	Agrees that pre-processing increases model performance?
(Angiani, et al., 2016)	2%	MNB	Yes
(Mat Zin, Mustapha, Murad, & Sharef, 2017)	2%	SVM	Yes
(Alam & Yao, 2018)	Ranging from 0% to 10%	MNB, SVM & Maximum Entropy	Yes

2.1.3 Text classification

Given the popularity of NLP, not only for sentiment analysis, it comes with no surprise that there are many possible classification methods available. Over the past 20 years these algorithms have become more complex due to the introduction of neural networks, but in turn they have also strongly increased in performance. The following provides an overview of the most prevalent classification algorithms used in sentiment analysis, starting with older, lexicon-based approaches and ending with the most modern, state-of-the-art Transformer models. Comparing these models then allows to identify the best suited model for the sentiment analysis task of this paper, as described in chapter one “Thesis Objective”. Understanding the function of different NLP models is also required to properly analyze the current research on the connection between text sentiment and stock performance, as older models and the deductions made from their results might not agree with the results from using more modern and therefore also potentially more precise models.

Sentiment classification can be split-up into three approaches: a Lexicon-based approach, a Machine Learning approach and a mixed form of both, denoted as hybrid techniques (Abdullah, Manjula, & Lakshman Naik, 2019, p. 150). The most established algorithms for Lexicon- and ML based approaches will be analyzed in more detail.

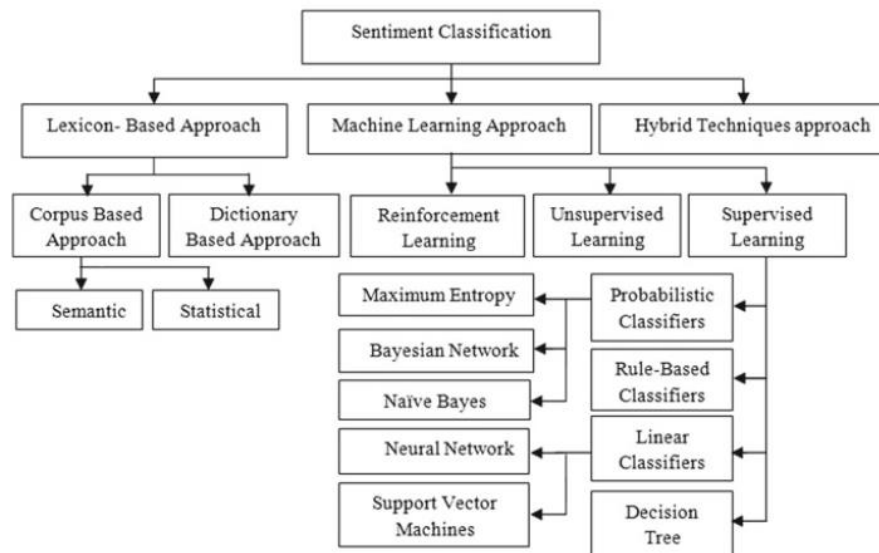


Figure 1 An overview of sentiment classification methods as per (Abdullah, Manjula, & Lakshman Naik, 2019, p. 150)

2.1.3.1 *Lexicon-based algorithms*

Lexicon-based algorithms work on the simple idea that some words strictly convey a positive sentiment, while other words strictly convey a negative sentiment. Hence, if one gathers enough words and assigns them to a positive, neutral or negative sentiment, one could classify a whole sentence or even a complete text. There are two methods that can be summarized under the lexicon-based approach. The first one is the corpus-based approach, and the second one is the dictionary-based approach. For both approaches, the value of pre-processing becomes visible, as a smaller amount of words and text complexity benefits these approaches.

Corpus-based approach: A corpus is defined as a large and structured set of machine-readable texts or words in either one or multiple languages (mono- or multilingual corpora). In order to make use of a corpus, the text within a corpus is often annotated with various tags. An example is Part-of-Speech tagging (POS), in which information about each word's part of speech (verb, noun, adjective, etc.) is added to the corpus in the form of tags (Zeroual & Lakhouaja, 2018, p. 87). Accordingly, one can also include semantic tags (positive or negative sentiment), or add the original word's synonyms and antonyms as tags (Kim & Hovy, 2004, p. 2). Another useful addition is sentiment intensity, so that one can differentiate between “the food here is *exceptional*” and “the food here is *okay*” (Hutto & Gilbert, 2014, p. 2). The sentence with “*exceptional*” would be given a stronger positive score than the sentence containing the word “*okay*”.

Dictionary-based approach: This works by gathering a smaller amount of so-called seed words from the to be analyzed text and assigning them a corresponding sentiment by hand (Kim & Hovy, 2004, p. 2). This data structure is called a dictionary. The dictionary is then expanded by adding synonyms and antonyms of these words, assuming that synonyms automatically also carry the same sentiment as the original seed word, e.g., the positive word “good” has the synonyms “virtuous, honorable, righteous” and the antonyms “evil, disreputable, unrighteous”. Hence synonyms of positive words are added to the positive list, and antonyms to the negative one. Therefore, a dictionary represents a corpus that is more specialized on the text that is to be analyzed.

To analyze the text, the amount of positive, neutral and negative words is identified and counted. The higher the amount of positive words, the higher the likelihood that the text conveys a positive sentiment. While this might sound simple, such methods have been used to identify signs of

depression in individual texts on social media, measure national happiness based on Facebook status updates and to determine if a couple is happy or unhappy according to their private messages (Pennebaker, Chung, Ireland, Gonazles, & Booth, 2007). The benefits of the lexicon-based approaches lie in their wide term coverage, as current corpora, such as the British National Corpus, cover up to 100 million English words (Burnard, 2021). However, they also face strong limitations as they assign a fixed sentiment orientation to a word (D'Andrea, Ferri, Grifoni, & Guzzo, 2015, p. 28). Language is complex and therefore such models face problems with negations in sentences, such as “the market did not outperform today”, as although a very positive word (outperform) is present, the sentence conveys a negative sentiment. This would be harder to notice for a dictionary-based model.

2.1.3.2 *Machine Learning algorithms*

ML approaches make use of either supervised, unsupervised or reinforcement learning to classify words, sentences or whole documents into a certain sentiment category. The goal of supervised learning algorithms is to find a function (h) that maps an input value x to an output value y , based on a provided set of training data (Russell & Norvig, 2003, pp. 650 - 651). The outcome variables y , are available in the training process of the algorithm and guide it in the right direction (Hastie, Tibshirani, & Friedman, 2009, pp. 485 - 490). The aim of unsupervised learning is to learn more about the features themselves, without making any specific predictions as in supervised learning. The algorithms detect what the features have in common or what the abnormalities in the dataset are, or if there are any hidden patterns (Langley, 2011, pp. 275 -279). The algorithms in Reinforcement Learning learn how to react to an environment based on trial-and-error, where actions have positive or negative rewards. The algorithm is however not told which actions yield the highest rewards and hence must try to discover them by itself (Sutton & Barto, 2017, pp. 1 - 5). While reinforcement learning has been successful in NLP domains such as dialogue generation and text-based games, it typically faces the problem of sparse rewards that leads to slow or no convergence (Deshpande & Fleisig, 2020, p. 1). Reinforcement learning is therefore not further analyzed in this paper.

The main algorithms in supervised learning for sentiment classification consist of MNB, Maximum Entropy, Decision Trees, SVM and Artificial Neural Networks (ANNs).

MNB is a probabilistic classifier that makes use of Bayes' theorem and therefore assumes conditional independence of all features. The word naïve is used to express the fact, that this assumption of independence is a very strong one and thus it is almost naïve to assume it (Hand & Yu, 2001, p. 385). To run an MNB classifier, all words in a text dataset are identified and a histogram is created. This is also called a bag of words approach. Using the number of observations of a certain word, for a certain sentiment classification (positive, neutral or negative), MNB calculates the likelihood of a text belonging to a certain category (positive, neutral or negative) based on the information it has from training data and then decides into which category it belongs, based on the highest probability. MNB treats all words equally, regardless of their position in the text. MNB works very fast and is useful for low amounts of training data, at least compared to other models (Kibriya, Frank, Pfahringer, & Holmes, 2004, p. 488). However, as mentioned, one assumes feature independence, which is hard to find in real life. An example would be that the word "ship" is probably more often found in a text in combination with the word "water", than with the word "grass" and hence the features (words) are not independent of each other.

The Maximum Entropy (log-linear) classifier is also a probabilistic classifier, but it is based on the principle of maximum entropy and therefore does not make an independence assumption as MNB does (Osborne, 2002, p. 1). The principle of maximum entropy states that the probability distribution which best represents the current state of knowledge about a system is the one with largest entropy. Entropy quantifies how much information there is in a random variable, or more specifically its probability distribution (Mitchell T. , 1997, p. 58). An event that is unlikely to happen has a larger entropy than an event with high probability of happening. The algorithm then tries to classify words or sentences into a category based on the highest likelihood of them belonging to a certain category. However, unlike MNB, it is not simply counting the co-occurrences of features and classes, but learns the optimal weights, which are maximized using a maximum a posteriori (MAP) estimation, using an iterative procedure. The benefit of this classifier is that it does not make an independence assumption for the individual datapoints, which is far more realistic. However, it is not as fast as MNB and therefore not appropriate given a very large number of classes to learn.

Decision Trees are non-parametric models that consist of nodes (decision points) and branches (connectors of nodes) (Mitchell T. , 1997, p. 53). At every node, each feature within the dataset is

evaluated, so that the observations in the dataset can be split into groups. The training dataset is used to build a decision tree model and a validation dataset can be used to decide on the appropriate tree size needed to achieve the optimal final model. Tree size is important, as a large tree, which has high number of nodes (decision points), would be very specifically created for a training set. Accordingly, it would likely overfit on the training data and not perform as well on new, unseen data. Therefore, the decision tree can be pruned, meaning that nodes that provide less information are removed (Hastie, Tibshirani, & Friedman, 2001, pp. 269 - 272). The main advantage of decision trees lies in their simplicity and hence their calculations and decisions are easy to understand. Therefore, their strength is that they are somewhat of the opposite of a black box algorithm. They also perform well with lower amounts of training data. However, as mentioned, they tend to overfit if not properly handled and their computation time for complex problems is also high.

SVM classifiers create a maximum-margin hyperplane that lies in a transformed input space and splits the example classes, while maximizing the distance to the nearest cleanly split examples (Shmilovici, 2005, p. 257). The parameters of the solution hyperplane are derived from a quadratic programming optimization problem. Concretely put, this means that each data point in the training data set is plotted as a point in n-dimensional space (where n is the number of features). Classification is then performed by trying to separate the datapoints via finding a hyperplane that separates the datapoints correctly into their classes. As there are multiple solutions to this, the hyperplane that has the largest distance between the nearest datapoints of each class is chosen as the best separator. This algorithm works very well for clearly separated classes but is constrained in performance if the classes are overlapping (noisy). SVMs are also computationally expensive for large datasets (Auria & Moro, 2008, p. 8).

Lastly, **artificial neural networks (ANNs)** can also be used for classification tasks. Due to their versatility, but also complexity, they can be used in supervised learning, but also for other types of learning. ANNs are capable of learning any nonlinear function. Hence, these networks are popularly known as Universal Function Approximators. They are designed to simulate the way the human brain analyzes and processes information. The main component of any feed-forward ANN is the perceptron. Each data point (feature x_i) is assigned an initial weight. Together with a bias, an augmented weighted sum between data points and weights is created. The perceptron

takes this as an input and calculates an output via an activation function. Activation functions introduce nonlinear properties to the network. This helps the network learn any complex relationship between input and output. This output should then match the assigned class of the input data. Accordingly, the model optimizes itself by adjusting the weights until it can properly classify its inputs into the correct categories (Rojas, 1996, pp. 151 - 173). ANNs consist of three types of layers: an input layer, one or more hidden layers, and an output layer. Any model that has more than three hidden layers can be considered as a deep learning model (Chollet, 2018, p. 49). The benefits of ANNs seem almost limitless, as the technology has been steadily growing since 2012. Given that the core concepts of artificial neural networks have been around for 50 years it becomes visible that the main constraint of this technology was and still is computational power requirements. Another common problem is that ANNs have a tendency to overfit the learning data. However, given the strong research activity in this field many methods to fight such overfitting have been introduced such as weight-decay (L2 regularization), sparsity (L1 regularization) or dropout (Maslej-Kresnakova, Sarnovsky, Butka, & Machova, 2020, p. 5). However, opposite to decision trees, their inner process can become a black box, given a complex enough ANN.

Other types of more advanced deep learning models include convolutional neural networks (CNN) and recurrent neural networks (RNN), such as the Long Short-Term Memory (LSTM) and the gated recurrent network (GRU) (Maslej-Kresnakova, Sarnovsky, Butka, & Machova, 2020, p. 7). The idea behind recurrent neural networks is to make use of sequential information. In a traditional neural network, it is assumed that all inputs (e.g. words or sentences) and outputs are independent of each other. However, for many tasks this is not the best idea. An example can be made by trying to predict what word comes next in a given sentence e.g. “The sun is [predict]”. Predicting what word, denoted here as [predict], is clearly easier if the model knows that the words “The, sun and is” stand before the to be predicted word. Accordingly, these networks are called recurrent networks because they perform the same task for every element of a sequence, with the output being reliant on the previous computations. Hence, this is described as a “memory” of a model. Having such a memory makes RNNs extremely useful to solve problems for time-series data or text and audio data. However, this memory is considered to be “short-term memory”, which means that if a whole paragraph of text is processed, the information (the words) at the very start of the paragraph might no longer influence the prediction for the very end

of the paragraph. Hence, LSTM and GRU models were invented, together with a mechanism called attention, as a solution to this short-term memory problem (Cho, et al., 2014, p. 1724). They have internal mechanisms called gates that can regulate the flow of information. These gates are used for the model to decide which information from previous steps was important (e.g. the word “sun” was important but the word “is” not so much) and what part of the new information of the input should be added to the current step. Through this process, RNNs work extremely well with text-based or generally any sequential data where the order of the features (e.g. the words) matters. This includes time-series, for example stock prices which contain non-linear, and high noise characteristics. A demonstration of the LSTM model is given in figure 2.

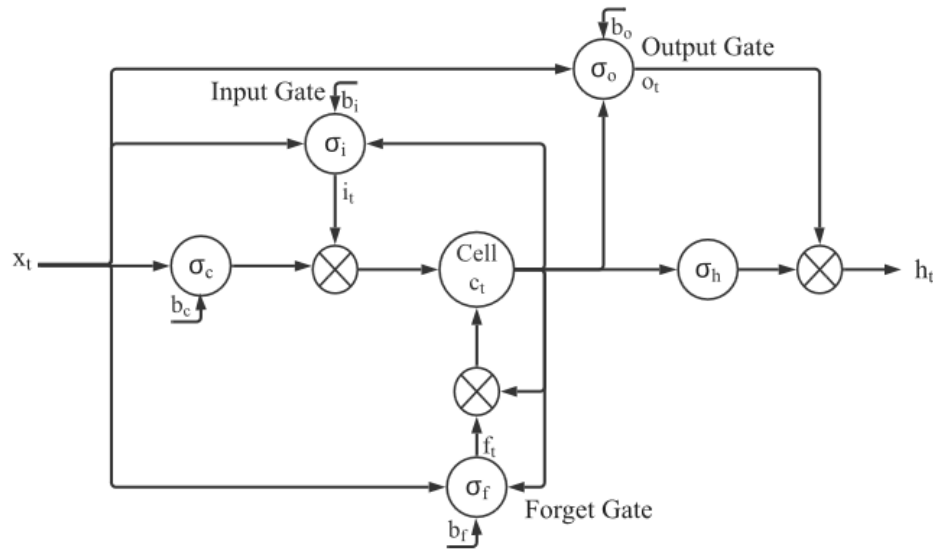


Figure 2 A LSTM model as per (Gao, Chai, & Liu, 2017)

In the model above, x_t is the input vector at time t , while h_t is the output vector. The cell c_t stores the state of the information, and is connected to the input gate, forget gate and output gate which in combination decide what stays in the networks memory and what information is forgotten. All gates are acted upon by $\sigma_i, \sigma_f, \sigma_o, \sigma_c$ and σ_h which are activation functions.

CNNs on the other hand are more specified on image recognition (Albawi, Mohammed, & Al-Zawi, 2017). CNNs capture the spatial features of an image. Spatial features refer to the arrangement of pixels and the relationship between them in an image. This helps in identifying the object accurately, the location of an object, as well as its relationship with other objects (displayed via pixels) in an image. While its largest performance factor lies in the image recognition, CNNs perform well with textual data too.

Having provided a basic overview of the different methods of classification within the NLP world, it becomes clear that the more modern neural network solutions provide a benefit for processing language over the older ML algorithms such as SVM, decision trees or the probability classifiers. Instead of just focusing on single words neural networks can take in the context of a whole sentence or paragraph, from start to finish, and thus can better decide on what to predict. Unsurprisingly, LSTM's and GRU's are also used in state-of-the-art deep learning applications like speech recognition, speech synthesis, natural language understanding, such as the Google Translate application (Schuster, Johnson, & Thorat, 2016). However, in 2017 a revolutionary paper named "Attention Is All You Need" was released, which introduced the Transformer (Vaswani, et al., 2017). This model claims to clearly beat the performance of RNNs and CNNs. Accordingly, the Transformer model will be analyzed in detail in section 2.2 as it seems to be a promising model for the sentiment classification task of this paper.

2.1.4 Analysis of Results

After deciding which of the possible algorithms to use for the classification task, the model can be run, and its results can be analyzed. For classifying text into sentiment categories, this paper focuses on these three categories: positive, neutral and negative. As an overview, a confusion matrix can be used. For the explanations of the below performance measures a 2 x 2 matrix makes more sense, as the display of True Positive or False Positive rates are calculated individually per class in the case of multi-class classification.

Table 2 A 2x2 confusion matrix (own creation)

		Predicted Class	
		True class	Negative class
Actual Class	True class	True Positive (TP)	False Negative (FN)
	Negative class	False Positive (FP)	True Negative (TN)

TP and TN both represent a correct classification into either of the two classes. FP represents a type I error and FN a type II error. In the case of a multi-class task, as described with the three categories positive, neutral and negative, the confusion matrix will be expanded by a column and

row, but the notation regarding what is considered a FP or FN will change depending on which class is evaluated.

Accuracy is defined as the ratio between the number of correctly classified samples and the overall number of samples (Chicco & Jurman, 2020, p. 2). Accordingly, if there are a total of 4 samples for every of the three classes (positive, neutral and negative), and 10 were correctly classified, the accuracy of the model would be $10/12 = 0.8333$ or 83.33 percent. However, with this accuracy it is not clear which class was predicted correctly and which was not – it simply indicates how many were correct or wrong. A better approach is given by the F1-Score.

The F1-Score is the harmonic average of precision and recall, where

$$precision = \frac{TP}{TP + FP} \quad (1)$$

$$recall = \frac{TP}{TP + FN} \quad (2)$$

And

$$F1\ score = 2 * \frac{precision * recall}{precision + recall} \quad (3)$$

Precision is a useful metric in cases where False Positives are a higher concern than False Negatives. Accordingly, Recall is a useful metric in cases where False Negatives are of higher concern than False Positives. The F1 score provides a way to express both of them in a balanced manner within a single score. However, when the dataset is unbalanced (the number of samples in one class is much larger than the number of samples in the other classes), the F1 score cannot be considered a reliable measure anymore, because it provides an overly optimistic estimation of the classifier ability on the majority class (Chicco & Jurman, 2020, p. 2). Despite the criticism, F1 remains one of the most widespread metrics among researchers (Chicco & Jurman, 2020, p. 3)

The Matthews correlation coefficient (MCC score) provides an effective solution for overcoming the class imbalance issue (Chicco & Jurman, 2020, p. 5). MCC only generates a high score if the predictor was able to correctly predict the majority of all data instances. It ranges in

the interval $[-1, +1]$, with extreme values -1 and $+1$ reached in case of perfect misclassification and perfect classification. It is given by Chicco & Jurman as follows (2020, p. 5):

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}} \quad (4)$$

However, even the MCC score has a weak spot, namely when either a whole row or column of a confusion matrix is zero. In that case, MCC is undefined. The decision whether to use MCC or F1-Score as main performance evaluation score therefore depends on if the classes in the dataset are balanced, or not.

With these tools a sentiment classifier can be analyzed on its performance. In terms of performance Roebuck (2011) makes a critical point: “The accuracy of a sentiment analysis system is, in principle, how well it agrees with human judgments. This is usually measured by variant measures based on precision and recall over the two target categories of negative and positive texts. However, according to research, human raters typically only agree about 80% of the time with the ratings made by other human raters. Thus, a program that achieves 70% accuracy in classifying sentiment is doing nearly as well as humans, even though such accuracy may not sound impressive. If a program were "right" 100% of the time, humans would still disagree with it about 20% of the time, since they disagree that much about any answer.”. Accordingly, the fact that not even humans consistently agree on all sentiment classification has to be kept in mind when building such classifier models using ML.

2.2 The Transformer

2.2.1 The Original Transformer Network

In 2017, Vaswani et. al. (2017) from the Google Brain team, published the paper “Attention is all you need” in which they introduce the Transformer as a model that can beat the state of the art performances of the previous LSTM models. To understand what the Transformer can do, a better description of the ideas behind encoders, decoders and attention is given, with the example of translating a text from one language into another. Before Bahdanau, Cho and Bengio (2015, pp. 1 - 15) introduced the attention mechanism in 2015, RNNs that used an encoder-decoder structure were the preferred models for translation (Cho, et al., 2014, p. 1724). The RNN Encoder-Decoder model consists of two RNNs. One RNN *encodes* a variable-length sequence of symbols (e.g. a sentence in French) into a fixed-length vector representation, and the other *decodes* the representation into another variable-length sequence of symbols (e.g. a sentence in English). Due to these two sequences, such models were also called seq2seq models. Each of the two RNNs, encoder and decoder, is a neural network that consists of a hidden state h and an optional output, given by y , which has a variable-length sequence $x = (x_1, \dots, x_T)$. At each time step t , the hidden state h_t of each of the RNNs is updated by (Cho, et al., 2014, p. 2)

$$h_t = f(h_{t-1}, x_t) \quad (5)$$

Where f is a non-linear activation function. This could range from a simple sigmoid function to a complex LSTM unit. The encoder reads each symbol of an input sequence (e.g. a sentence) sequentially. As it reads each symbol (word), the hidden state of the network changes as per equation 5. After reading the last symbol in the sequence (last word in the sentence), which is marked by an end of sequence symbol (e.g. a full-stop or exclamation mark), the hidden state of the network is a summary, denoted as c , of the complete input sequence. The decoder generates the output sequence by predicting the next symbol in the sequence y_t , based on the information in c and y_{t-1} . Therefore, the computation of the hidden state h for the decoder is updated by (Cho, et al., 2014, p. 2)

$$h_t = f(h_{t-1}, y_{t-1}, c) \quad (6)$$

The encoder and decoder of the proposed model are jointly trained to maximize the conditional probability of a target sequence given a source sequence. A visual guide for this process is given in figure 3 below.

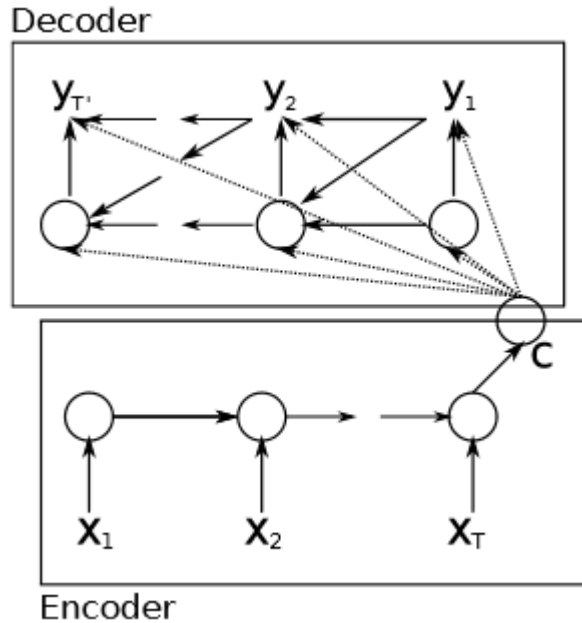


Figure 3 The RNN Encoder-Decoder (Cho, et al., 2014, p. 1725)

While this model was able to consider previous symbols as inputs for the current prediction, Cho et al. (2014) found its weakpoint to be very long sentences. They demonstrated that the encoder created worse summaries c when trying to interpret longer input sentences. This was accordingly called the long-range dependency problem of RNNs. To remedy this problem, Bahdanau, Cho and Bengio (2015) introduced the attention mechanism.

For this, they again made use of an encoder-decoder model. The previously described encoder reads an input sequence x by starting at the first symbol (word) x_1 and stopping at the last one x_t . The newly proposed encoder in Bahdanau, Cho, & Bengio (2015) is a bidirectional RNN (BiRNN), so that not only preceding words, but also following words can be considered. A BiRNN consists of forward and backward RNNs. The forward RNN reads the input sequence from x_1 to x_t and calculates a sequence of forward hidden states $\vec{h}_1, \dots, \vec{h}_{T_x}$. Accordingly, the backward RNN reads the sequence in the reverse order from x_t to x_1 , which results in a sequence of backwards hidden states $\overleftarrow{h}_1, \dots, \overleftarrow{h}_{T_x}$. By doing that each symbol (word) is given an annotation

h_j , which is made up of the concatenated information of the forward and the backward hidden states. The decoder can then again make use of this additional information via a summary c . The context summary c is calculated as a weighted sum of the different annotations:

$$c_i = \sum_{j=1}^{T_x} a_{ij} h_j \quad (7)$$

Where a_{ij} is the weight of each previously calculated annotation h_j . The approach of taking a weighted sum of all the annotations can be understood as computing an expected annotation, where the expectation is about how well an input word x_i and output word y_i align. The higher the probability of alignment is, the higher a_{ij} will be. This specific process implements the attention mechanism and allows the decoder to decide which parts of the input sentence it should pay special attention to.

With the basic description of the inner workings of encoders, decoders and the attention mechanism, the Transformer, as introduced by Vaswani et. al. (2017) can now be analyzed in closer detail. The Transformer is a model architecture that completely relies on the attention mechanism, without the dependency on RNNs. More specifically, it relies on self-attention. Self-attention, also called intra-attention, is the mechanism of relating different positions of a single sequence (sentence) in order to compute a representation of the sequence (Cheng, Dong, & Lapata, 2016, p. 558). In this form of attention, each embedding of a word has three different vectors corresponding to it, called Key (K), Query (Q), and Value (V). The output for a word is the weighted sum of the values, where the weight of each Value is given by a function of the Query with the corresponding Key. In other words, a Query is raised, which hits the Key of the input. The Key is compared with the current memory, which is the previous Value. In practice, this is done by computing the dot products of the Query with all Keys, divided by $\sqrt{d_k}$, where d_k is the dimension of Keys. Then a softmax function is applied to obtain the weights on the Values (Vaswani, et al., 2017, p. 4).

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (8)$$

Following the explanations of Aggarwal (2018) the softmax function, also known as softargmax, converts its input into a range between 0 and 1 and normalizes them, so that the sum of all outputs is equal to one as shown in figure 4. This essentially turns its inputs into probabilities, ranging from zero to one, which all together sum to one. The main point of softmax is that it is an exponential function and therefore enlarges differences. It is mathematically defined as (Aggarwal, 2018, pp. 14 - 15):

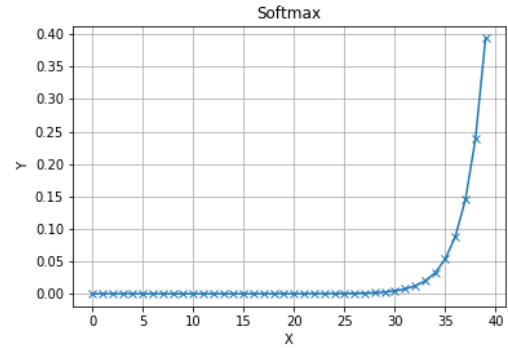


Figure 4 Softmax function (own creation)

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_i e^{x_i}} \tag{9}$$

Where x_i are the outputs of neuron that are acted upon the activation function softmax.

Together this is also called “Scaled Dot-Product Attention”. As a concrete example, consider the sentence “I enjoy working with Python”. To calculate the attention for the word “working”, the dot product of the Query vector of the embedding “working” to the Key vector of each of the previous words “I” and “enjoy” is calculated. These values are then divided by the dimension of the Keys d_k and lastly, the softmax operation is applied. This process is shown in figure 5 on the left side.

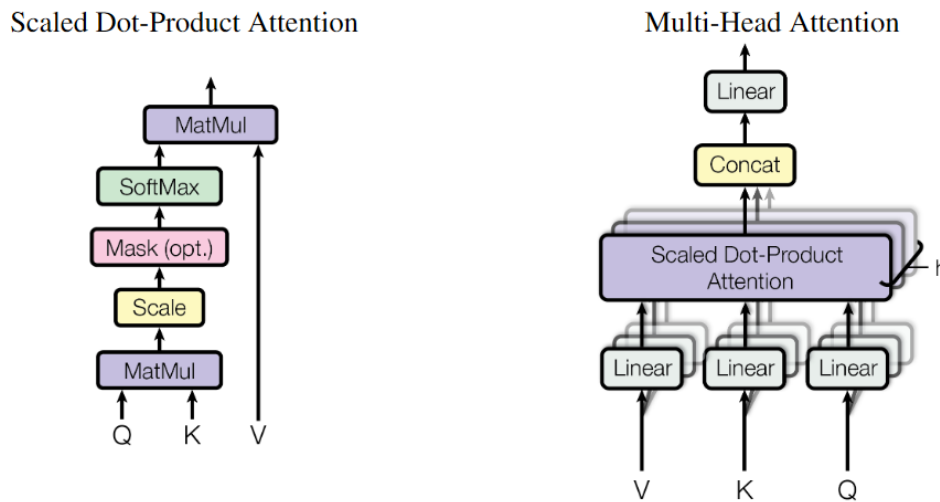


Figure 5 The Scaled Dot-Product Attention (left) and the Multi-Head Attention consisting of several attention layers running in parallel (right) (Vaswani, et al., 2017, p. 4)

Multi-Head Attention, as shown in figure 5 on the right, takes this concept a little further. It projects the Keys (K), Queries (Q), and Values (V) h times in parallel. Only after that, the results are concatenated and given as an output. Using multiple attention heads allows the model to attend to multiple different information points at the same time. Accordingly, this multi-headed attention is used in both the encoder and decoder of the Transformer model structure as shown in figure 6.

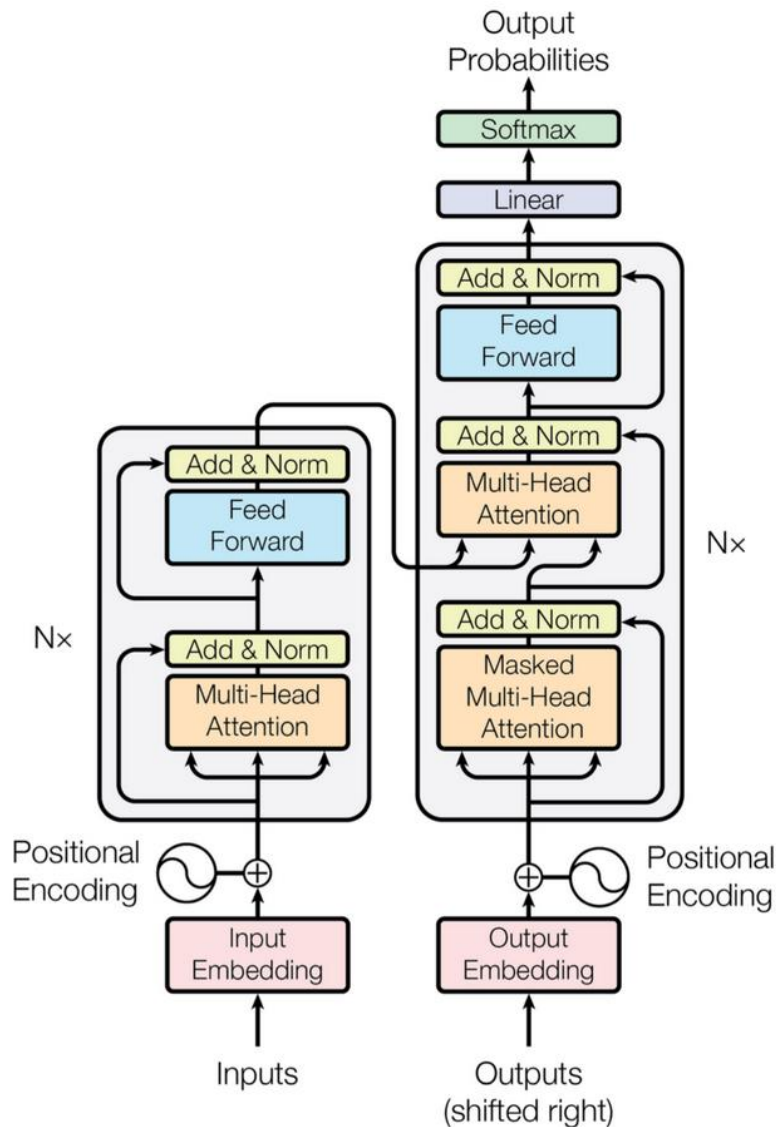


Figure 6 The Transformer architecture (Vaswani, et al., 2017, p. 3)

Consider the left side of figure 6. To start the process, the inputs are embedded. An embedding is a mapping of text or a discrete, categorical variable to a vector of continuous numbers. Then the model is given some sense of direction in terms of where the analyzed word stands in the sen-

tence. This piece of information is attached to every word and is called positional encoding. Again, considering the sentence “I enjoy working with Python”, the model knows that the embedded form of the word “I” is the first in the sentence, “enjoy” the second and so on. The summed information of the input embedding and positional encoding are then used in the encoder. Each encoder consists of $N = 6$ identical, stacked layers. Each layer has two sub-layers. The first sub-layer is the multi-head attention mechanism, which then forwards information into a simple feed-forward neural network (the second sub-layer). To pass the information from the multi-head attention stage to the feed-forward network it passes through an Add & Norm layer, which applies layer normalization. Training deep neural networks like the Transformer is computationally expensive. Thus, layer normalization, as introduced by Lei Ba, Kiros and Hinton (2016), normalizes the activations of a layer by its mean and standard deviation. This drastically reduces training time for such networks. Taking a closer look at the arrows in figure 6, one can see that after the positional encoding, information flows not only directly into the multi-headed attention mechanism, but also directly into the Add & Norm layer. This is called a residual connection, as introduced by He, Zhang, Ren & Sun (2015), and improves the networks performance. Creating better performance for a neural network cannot be simply accomplished by stacking more layers to increase complexity - with an increasing network depth, accuracy gets saturated at some point and then degrades rapidly. Such degradation is not caused by overfitting, and adding more layers after degradation just leads to higher training error (He, Zhang, Ren, & Sun, 2015, p. 770). Therefore, residual neural networks (ResNets) prevent this by adding so-called skip connections. A layer, here the multi-headed attention, is skipped so that both, original information and information after the multi-headed computation is fed into the Add & Norm Layer. After these steps for the first sublayer of the encoder, the information flows into the second sublayer of the encoder, which consists a feed-forward neural network and another normalization layer. The feed-forward neural network consists of two linear transformations with a ReLU activation in between. A Rectified Linear Unit (ReLU) as shown in figure 7 is composed out of two linear functions and returns 0 for

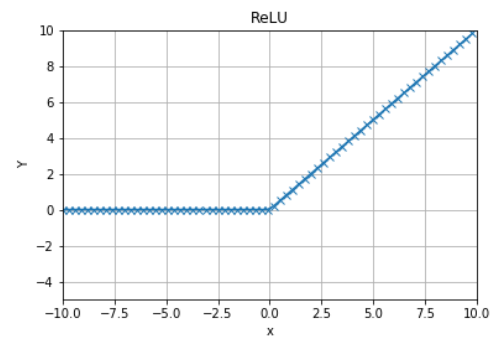


Figure 7 ReLU function (own creation)

all negative inputs and returns positive inputs unchanged (Goodfellow, Bengio, & Courville, 2016, pp. 191 - 195).

While the linear transformations in the network are the same across different positions, they use different parameters from layer to layer. So far, this covered the encoder side of the Transformer. After the information went through the encoder it is processed by the decoder, shown on the right side of figure 6. The decoder also consists of $N=6$ identical, stacked layers. Each layer includes three sub-layers. – a masked multi-attention head, a “normal” multi-attention head and again a feed-forward neural network. The normal multi-attention head and the feed-forward network work in the same way as explained for the encoder. Also, layer normalization and residual connections are used again in the same style. The masked multi-attention head is a modified version of the normal multi-attention head. Masking is required so that both the encoder and decoder can work in parallel and therefore the decoder does not have to wait on the results of the encoder before it can start working. Parallelization is useful since it allows the model to work faster. The exact process of masking is best explained with an example. Consider translating the English sentence “I like Python” into German – “Ich mag Python”. As always, these words are embedded, meaning they are represented as numbers in a vector. For simplicity, consider the number 1, 2 and 3 as the vector representations of the input sequence “I like python”. For the German version these numbers would be for example 10, 11, and 12 for “Ich mag Python”. In reality these vectors would be much longer given that there are normally much more words in any dataset. Also, there would be additional tokens that mark the start and end of the sequence. During the training, the network knows that the expected translation of “I like python” is “Ich mag Python”. Therefore, it can process the translation of all three words in parallel:

- Parallel Operation 1 with input 1, 2, 3 - Trying to predict 10
- Parallel Operation 2 with input 1, 2, 3 and 10 - Trying to predict 11
- Parallel Operation 3 with input 1, 2, 3, 10 and 11 – Trying to predict 12

This is where the masking comes in. The algorithm hides (masks) a part of the known output sequence (“Ich mag Python”) for each of the parallel operations. For the first operation it hides the entire output, for the second it hides the second and third output and for the third operation it hides the third output. This masking, combined with fact that the output embeddings are offset by

one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i (Vaswani, et al., 2017, p. 3). This concludes the working process of the decoder. As a result, the Transformer can be trained significantly faster than architectures based on recurrent or convolutional layers, at least for its original purpose of language translation. At its publication, this model achieved new state-of-the-art performance on both, English to German and English to French translation at a fraction of the training cost of previous algorithms (Vaswani, et al., 2017, p. 8). Since its publication in 2017, research has made strong improvements in NLP, using the Transformer as a foundation.

2.2.2 Bidirectional Encoder Representations from Transformers (BERT)

In 2019 Devlin, Chang, Lee & Toutanova (2019) from the Google AI Language team introduced a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. BERT is conceptually simple and obtains new state-of-the-art results on eleven natural language processing tasks, where performance increases lie between 4 and 7 percent in comparison to the original Transformer, indicating how powerful BERT is (Devlin, Chang, Lee, & Toutanova, 2019, p. 4171). In the Transformer, every token (word) could only attend to previous tokens in the self-attention layer. This is sub-optimal for sentence-level tasks and also potentially harmful for certain token-level tasks, such as question answering. BERT alleviates these problems by introducing two novelties: a masked language model (MLM) pre-training objective and a “next sentence prediction” task that jointly trains text-pair representations. Apart from these two methods, the basis of BERT is built upon the previously explained Transformer. However, if the task of BERT is not language translation, only an encoder and no decoder is required.

The MLM randomly selects and masks some tokens (words) of the input. The objective is then to predict the original vocabulary id of the masked token, based only on its surrounding context. Unlike a left-to-right language model, this MLM process enables the representation to fuse the left and the right context together, thus the name bidirectional in BERT. The MLM is also called *Cloze* task in literature (Taylor, 1953). For example, the model would randomly mask a word in the sentence “The child went to the store and bought a pair of shoes”, so that it would have to fill in the blank based on the context surrounding the masked word, e.g. “The child went to the store and bought a ____ of shoes”. This blank is denoted as a [MASK] token. BERT gives every word,

that is not at the beginning or end of a sentence a 15 percent chance to be masked. However, this random masking approach leads to the problem that the model only tries to predict a token when the [MASK] token is present in the input. Hence, out of the normally 15 percent of tokens selected for masking, BERT applies the following (Devlin, Chang, Lee, & Toutanova, 2019, p. 4183):

- 80 percent of the tokens are replaced with the token [MASK].
- 10 percent of the time tokens are replaced with a random token.
- 10 percent of the time tokens are left unchanged. The purpose of this is to bias the representation towards the actual observed word.

Using this procedure, the model does not know which words it will be asked to predict, or which have been replaced by random words, thus it is forced to keep a distributional contextual representation of every input token. While the Transformer was predicting the word that was next in the sequence order, MLM makes the model look in both directions for its prediction of the masked word. It therefore takes both the previous and next tokens into account at the same time. However, this results in a model that converges much more slowly than left-to-right or right-to-left models.

The second novelty, the next sentence prediction, allows the model to understand the relationship between two sentences, thus providing the model overall with more context. Such potential relationships between sentences are required for tasks such as question answering. During training, BERT gets sentences in pairs of two as inputs. This is done in a manner that 50 percent of the time, the second sentence comes after the first one and the other 50 percent of the time, the second sentence is a random sentence drawn from the overall text. It then is required to predict whether the second sentence is independent from the first sentence or not. An example of this is given in figure 8.

<p>Sentence A = The man went to the store. Sentence B = He bought a gallon of milk. Label = IsNextSentence</p>		<p>Sentence A = The man went to the store. Sentence B = Penguins are flightless. Label = NotNextSentence</p>
---	--	---

Figure 8 Predicting if the second sentence is in context to the first one (Devlin & Chang, 2018)

Having explained the core ideas behind the benefits of BERT, the model can be looked at in more detail. The actual architecture of BERT is almost identical to the original Transformer, as

previously explained in this paper. BERT simply uses different amounts of layers (12), hidden states (768) and self-attention heads (12). The input representation is able to represent both, a single sentence and a pair of sentences in a one-token sequence. For BERT, a sentence can be any span of contiguous text, rather than a normal sentence. Accordingly, a sequence may also contain one or multiple such “sentences”. Before processing data, BERT uses token embeddings, segment embeddings and position embeddings. Token embeddings add a [CLS] token to the input word tokens that stand at the beginning of a sentence. Additionally, a [SEP] token is implemented at the end of every sentence. With these tokens, the model knows when a sentence begins and ends. With segment embeddings, every single sentence is given another token (A or B) to indicate to which original sentence the token belongs. Lastly, with positional embeddings, each token is provided with a number that indicates its position in the sentence. For any given token, its input representation is constructed by summing the corresponding token, segment, and position embeddings as shown in figure 9 below.

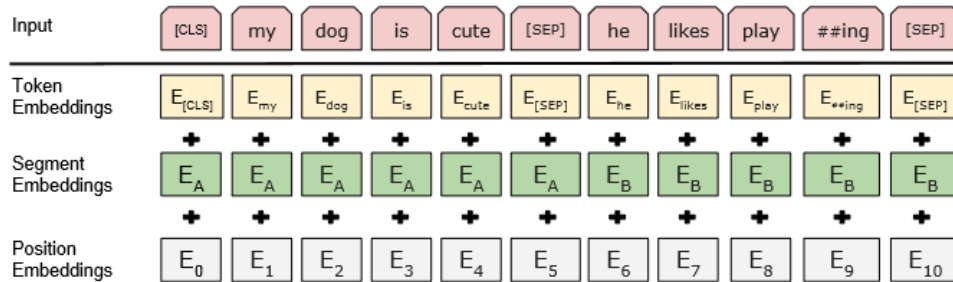


Figure 9 BERT input representation shown as a sum of token, segment and position embeddings (Devlin, Chang, Lee, & Toutanova, 2019, p. 4174)

The BERT implementation has two steps: pre-training and fine-tuning. During pre-training, the model is trained on unlabeled data over different pre-training tasks, which therefore represents an unsupervised learning approach. After pre-training, fine-tuning works by initializing the BERT model with the pre-trained parameters, which then are fine-tuned using labeled data from specific downstream tasks (e.g. question answering). With that, BERT can be applied to many different (downstream) language processing tasks, as long as there is labeled data for fine-tuning available. This means that anyone can take the large pre-trained basis of BERT and fine-tune it onto any specialized task, as long as they have a respective training dataset for it. This process is shown in figure 10.

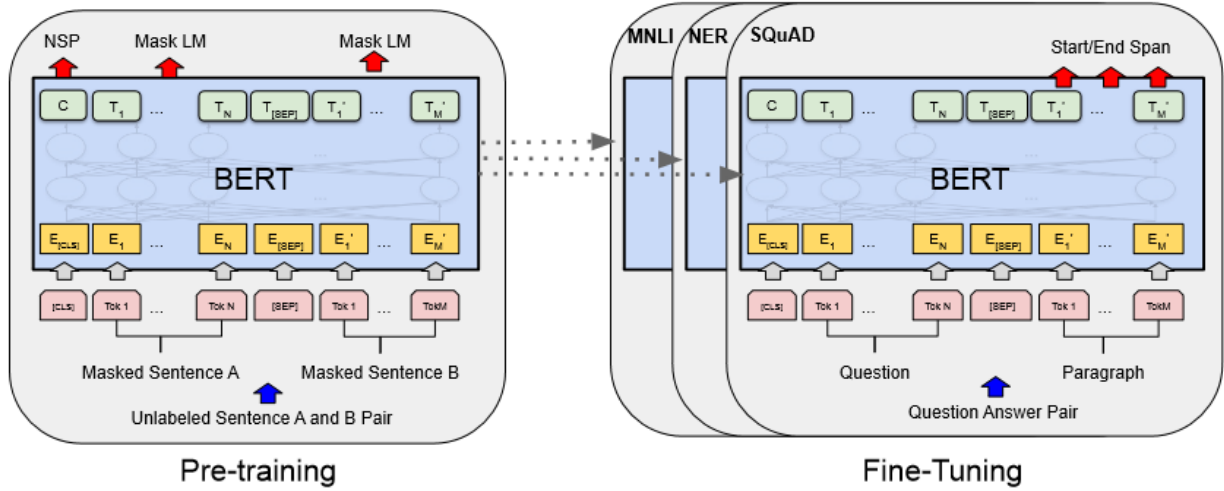


Figure 10 The pre-training and fine-tuning procedures for BERT (Devlin, Chang, Lee, & Toutanova, 2019, p. 4173)

The model was pre-trained using the BooksCorpus (800 million words) from (Zhu, et al., 2015) and English Wikipedia (2,500 million words). The pre-training took 4 days to complete (Devlin, Chang, Lee, & Toutanova, 2019, p. 4183). Compared to pre-training, fine-tuning is relatively inexpensive and therefore BERT can be fine-tuned to work on different downstream tasks (such as classification), even when the user does not have access to high-performance GPUs. The original paper by Devlin, Chang, Lee & Toutanova (2019) describes the fine-tuning for a question answering task. However, for the purpose of this paper the fine-tuning for a classification task is more relevant as described by Chi, Qiue, Xu & Huang (2019, pp. 194-206). The additional steps required to turn BERT into a classification model is straightforward. As mentioned, the first token of a sequence is always marked with [CLS]. For a classification task, the final hidden state h of the [CLS] token is taken as a representation of the whole sentence. Then, a softmax classifier is added on top of BERT, to predict the probability of the label c :

$$p(c|h) = \text{softmax}(Wh) \tag{10}$$

Where W is the task-specific parameter matrix. All BERT parameters from pre-training as well as W are then jointly fine-tuned by maximizing the log-probability of the correct label (Chi, Qiue, Xu, & Huang, 2019, p. 195). With this process, a new record on both binary and multi-class sentiment classification on the IMDb and Yelp dataset were achieved (Chi, Qiue, Xu, & Huang, 2019, p. 198). The optimal hyperparameter values for fine-tuning are task-specific, but the fol-

lowing range of possible values seem to work well across all tasks (Devlin, Chang, Lee, & Toutanova, 2019, p. 4184):

- **Batch size:** 16, 32
- **Learning rate (Adam):** 5e-5, 3e-5, 2e-5
- **Number of epochs:** 2, 3, 4

It was also observed that large data sets (e.g., 100k+ labeled training examples) were far less sensitive to hyperparameter choice than small datasets. As mentioned, fine-tuning is quite fast, so it is reasonable to simply run an exhaustive search over the above-mentioned parameters and choose the model that performs best on the result of the hyperparameter optimization. However, to facilitate this process a distilled version of BERT, named DistilBERT, was introduced in 2019 (Sanh, Debut, Chaumond, & Wolf, 2019). They have shown that it is possible to reduce the size of the BERT model by 40 percent, while retaining 97 percent of its performance and making it 60 percent faster (Sanh, Debut, Chaumond, & Wolf, 2019, p. 1). This trade-off is more than worth it for the personal use of such algorithms.

Given the complexity, state-of-the-art performance in different NLP tasks and bidirectionality of the Transformer algorithms, especially BERT, it is assumed that this model would be best for sentiment classification. This is also supported by a comprehensive study where two datasets were analyzed on sentiment using a plethora of techniques ranging from lexicon-based approaches to Transformers, including BERT (Mishev K. , Gjorgjevikj, Vodenska, Chitkushev, & Trajanov, 2020). The first dataset in this study consisted of financial news headlines and the other of randomly selected financial news sentences. The performance of lexical-based models averages around an MCC score of 0.327, while ML classifiers and deep neural networks achieve MCC around 0.667. BERT models and its adaptations (DistilBERT and similar) achieve an MCC score ranging from 0.786 to 0.895, thus clearly beating the non-transformer models. Having learned that BERT provides the best basis for a sentiment classification with record performance, it will be chosen for the sentiment project in this paper.

2.3 Complexity

The second NLP task for this paper consists of analyzing text complexity. Compared to the previously introduced deep learning models, the process is simpler and much more approximative.

The original motives behind analyzing text complexity are plentiful and range from creating simple text for emergency instructions to assessing how hard a textbook should be for students in a certain school grade (Allington, McCuiston, & Billen, 2015, pp. 491 - 501). The motivation to have texts that are easily readable lies in the fact that easier reading improves comprehension, retention of information and reading speed (DuBay, 2007, p. 5). Text complexity is defined as an objective approximation to relative text difficulty which can be based on factual criteria (Göpferich, Jakobsen, & Mees, 2009, p. 63). Therefore, it is not the individual perception of text difficulty, as for example a text about machine learning is easy to understand for data scientists, while a child would struggle with it. Multiple methods exist to measure text complexity, whereby readability indices are the most used (Göpferich, Jakobsen, & Mees, 2009, p. 63). Klare (1963) suggested early on that word difficulty (i.e. via count of syllables) and word count are the most important criteria in assessing text readability. Accordingly, readability indices were created that focus on such criteria. With readability indices, one can give a text a certain score to measure how hard, or easy it is to read. However, qualitative factors such as prior knowledge of the person reading the text, or its rhetorical structure are not easily quantifiable. Hence, readability scores are approximations. Their short comings are that they do not account for grammar, spelling, tone or the previously mentioned qualitative aspects. Therefore, a text that is riddled with grammatical issues might not receive the same readability score as a text with perfect grammar. Nonetheless, their theoretical and statistical validity was proven numerous times since their inception (DuBay, 2007, p. 6). There is a plethora of reading scores, where some are specialized on technical writing (e.g. in a computer science magazine), while some can be used for all textual purposes. The problem for the analysis task of this paper is that almost none of the readability scores have been empirically tested for the German language as the research in this area predominantly focuses on English. Hence, for the purpose of this paper the Flesch readability test (Flesch, 1948) and the Wiener Sachttextformel (Bamberger & Vanecek, 1984) were chosen as measurement tools. They were selected due to their displayed usage in literature and also their functionality for the German language.

2.3.1 Flesch reading-ease score

The Flesch reading-ease score, as introduced by Richard Flesch (1948), counts the total numbers of syllables, words and sentences within a text and ranges them within a score from 0 to 100. The

lower the score is, the harder the text is to read. The formula for the Flesch reading-ease score is given by (Flesch, 1948):

$$Flesch\ score = 206.835 - 1.015 \left(\frac{total\ words}{total\ sentences} \right) - 84.6 \left(\frac{total\ syllables}{total\ words} \right) \quad (11)$$

The maximum score of this formula is 121.22, however there is no limit on how low the score can be. A negative score would also be a valid result. To better interpret the scores an overview in the table below was created by Flesch.

Table 3 Flesch reading-ease score assessment as per (Flesch, 2016)

Score	School Level (US)	Assessment
100.00–90.00	5th grade	Very easy to read. Easily understood by an average 11-year-old student.
90.0–80.0	6th grade	Easy to read. Conversational English for consumers.
80.0–70.0	7th grade	Fairly easy to read.
70.0–60.0	8th & 9th grade	Plain English. Easily understood by 13- to 15-year-old students.
60.0–50.0	10th to 12th grade	Fairly difficult to read.
50.0–30.0	College	Difficult to read.
30.0–10.0	College graduate	Very difficult to read. Best understood by university graduates.
10.0–0.0	Professional	Extremely difficult to read. Best understood by university graduates.

To reach a very high score, a text would have to consist of exclusively one-syllable words. This method has a correlation of 0.91 with comprehension as measured by reading tests and is also applied by the US department of defense as readability testing standard for its documents and forms (Si & Callan, 2001, p. 574). On average, German words tend to be longer than English words while sentence length is similar. Accordingly, the formula for a German Flesch implementation was adapted by a Swiss researcher as follows (Amstad, 1978):

$$Flesch\ score_{German} = 180 - \left(\frac{total\ words}{total\ sentences} \right) - 58.5 \left(\frac{total\ syllables}{total\ words} \right) \quad (12)$$

The interpretation for this adapted formula remains the same as given in table 3.

2.3.2 Wiener Sachtextformel

The Wiener Sachtextformel was created by Richard Bamberger and Erich Vanecek in 1984 (Bamberger & Vanecek, 1984) as a measure to assess the readability of German texts. Its original purpose was to assess for which school level a given text was suitable. However, it also finds its usage in the analysis of political speeches and scientific texts (Kulgemeyer & Starauschek, 2013). There are four different formulas for the Wiener Sachtextformel, ranging from simple to more complicated. As all formulas can be implemented in a Python program, only the most complicated formula is considered, for the purpose of maximum accuracy. The first Wiener Sachtextformel (WST) is given by (Bamberger & Vanecek, 1984):

$$WST = 0.1935 * MS + 0.1672 * SL + 0.1297 * IW - 0.0327 * ES - 0.875 \quad (13)$$

Where

MS = percentage of words with more than three syllables

SL = the average sentence length

IW = percentage of words with more than six letters

ES = percentage of words with one syllable

As this formula was explicitly created for the German language, no adaptation is necessary as compared to the Flesch score. In terms of interpretation the results range on a scale from 4 to 15, where 4 is the easiest in terms of understandability. It therefore is set-up opposite to the Flesch score, where easier texts have higher scores. However, there is no clear grading scale with descriptions as for the Flesch score.

These explanations conclude the theoretical process behind sentiment analysis and text complexity in terms of NLP.

3 Results in Literature

Before conducting the analysis for the Swiss small and mid cap companies, it deems necessary to establish the current state of findings in literature with regard to this topic. Accordingly, results of current literature will be presented to two main topics on a global scale: the findings on implications of sentiment for stock price movements and the findings on implications of text complexity for stock price movements.

An analysis of the current literature for sentiment analysis in finance shows that most papers use lexicon-based approaches that focus on large word corpora, while newer methods such as ML-based classifiers are rarely used (Git Hiew, et al., 2019, p. 1). Wisniewski and Yekini (2014) have analyzed a total of 1'262 annual reports of 209 UK listed firms on their sentiment using a lexical-based approach between 2006 and 2012. They measured the frequency of words associated with praise, concreteness and activity to forecast future stock performance. Praise words consist of words such as successful, intelligent, admirable or beneficial. Words for the concreteness category are for example payments, factors, estate, savings or finance. For the activity category, words such as completion, launch, achieving or strengthens will increase the value of activity, while words like shutdown, standstills, constrained and puzzled will lead to its decrease. They have found that these three indicators have Pearson correlations to stock price movements between 0.0708 and 0.08027, while an increase in the frequency of any of the three categories would lead to future stock price increases of up to 3 percent. Additionally, they found that their concreteness category displays a correlation of -0.1926 to earnings surprises. However, as a critique point, their earnings surprise is calculated against a benchmark of random-walk stock price movement (Wisniewski & Yekini, 2014, p. 15), which does not align with the traditional definition of an earnings surprise, where the actual earnings are compared to the earnings estimates made by skilled financial analysts (Pinto, Henry, Robinson, & Stowe, 2010). Another lexical-based approach was conducted by Uhr, Zenkert and Fathi (2014). Using keyword spotting and word frequencies multiple short- and long-term moving averages of sentiment levels were created. These sentiment levels were based on a news corpus containing 918'427 finance related news, researcher comments and analyst ratings from July 2000 to February 2014. All news were written in German and in relation to the 30 stocks within the German DAX. With this information a trading strategy was created, where a buy signal for stocks is triggered if the short-term

moving average crosses or notes over long-term moving average. Accordingly, a sell signal is created in the opposite situation. It was found that short-term changes in sentiment were noisy, while longer-term sentiment trends provided a better base for a trading strategy. As a result, the trading strategy, which also allowed for short position, has shown that document-level sentiment analysis provides valuable input in downturns, while trading on sentence level sentiment analysis performs best during rally phases (Uhr, Zenkert, & Fathi, 2014, p. 7). Instead of financial news, other researchers also focus on data from Twitter. Bollen, Mao & Zeng (2010) have analyzed over 9 million tweets from February to December in 2008 with two mood tracking tools. One of the tools explicitly reads mood states from expressions such as “I feel” and “I don’t feel”, while the other focuses on more traditional text polarity (positive or negative) using word frequencies. Mood is split up into six dimensions (calm, alert, sure, vital, kind and happy). They find, using a granger causality analysis, that calm and happy moods are prognostic for the development of the American Dow Jones Industrial Average Index (DJIA), however no such indication is found for the positive and negative sentiment analysis. They additionally find that they can forecast the direction of the stock price movement within the DJIA with an accuracy of 86.7 percent with the created model (Bollen, Mao, & Zeng, 2010, p. 8). Additional research for American stocks is conducted by Tetlock (2007). He finds that pessimistic words within columns of the Wallstreet Journal predict negative stock price movements between 1984 and 1998. This paper was the first to find that news media content can predict movements in broad indicators of stock market activity. The analysis is conducted using word frequencies, which are then classified into sentiment using Harvard’s psychosocial sentiment corpus (H4N). Together with Saar-Tsechansky and Macskassy (Tetlock, Saar-Tsechansky, & Macskassy, 2008) this methodology was also applied to the Dow Jones News Service which contained news stories about individual S&P 500 firms from 1980 to 2004. It was found that the fraction of negative words within these news articles forecast low firm earnings, especially if the news articles focus on company fundamentals. Additionally, they found that the firm’s stock prices briefly underreact to the information that is embedded in the negative words. However, these results are strongly criticized by Loughran & McDonald (2011). They provide evidence based on 50’155 annual reports of American firms between 1994 and 2008 that the H4N corpus, as applied by Tetlock, Tsechansky & Macskassy, substantially misclassifies words when taken in the context of finance. Certain misclassified words, such as “taxes” or “liabilities”, simply add noise to the measurement of tone, as they are

not inherently positive or negative in the financial world and therefore reduce the meaningfulness of the estimated regression coefficients in Tetlock, Saar-Tsechansky & Macskassy (2008). Moreover, Loughran & McDonald find that 73.8 percent of the negative words according to the H4N corpus cannot be considered negative in a financial context (Loughran & McDonald, 2011, p. 35). A more modern approach is taken by Git Hiew et al. (2019). They conduct sentiment analysis for three companies listed on the Hong Kong stock exchange using 117'029 Weibo post between the years 2016 and 2018. Weibo can be described as the Chinese version of Twitter. The classification is done with a BERT algorithm that achieves an F1-Score of 78.5, which also beats other tested models (transformer, CNNs and dictionary-based approach). However, they did not find a clear connection between today's sentiment index and tomorrow's stock return for any of the three analyzed companies. The maximum correlation between sentiment and stock performance was 0.0710 (Git Hiew, et al., 2019, p. 5). Early in 2021 Ching-Ru & Hsien-Tsung also made use of BERT to create sentiment classifications (negative and positive) from financial, political, and international news content related to six stocks listed on the Taiwanese stock exchange between 2015 and 2020 (Ching-Ru & Hsien-Tsung, 2021). These sentiment scores were then, together with past stock information such as closing price and volume, used as an input for an LSTM for the purpose of stock price prediction. The experimental results found that the combination of sentiment data together with historical stock information increases the stock price prediction accuracy of the LSTM, as compared to a model that would exclusively use stock data.

The research conducted in terms of text complexity within this area is less fruitful. The main findings come from Li (2008), which analyze the readability of annual reports of American listed companies using the Fog index (a score system similar to the Flesch reading-ease score) between 1994 and 2004. The overall report length was analyzed as well. The findings show that firms with lower earnings tend to file annual reports that are more difficult to read, while an increase (decrease) in earnings from the previous year results in annual reports that are easier (more difficult) to read. You and Zhang (2009) find unusual trading volumes and stock-price movements around the filing dates of American annual reports. They also find that investor response to annual reports is sluggish, as most relevant information (earnings, dividends, sales growth etc.) has already been previously released in earnings announcements. In that regard, they find that this underreaction is even more drastic for firms with more complex annual reports. Complexity was

assessed using a word count method, where a higher number of complex words results in a higher complexity of the overall text.

To summarize, it can be stated that most research for the sentiment of financial texts is based upon a lexical approach, where most results are found in connection to some sort of “tone” analysis, given by the frequency of appearance of certain words. The maximum timeframe analyzed in the mentioned papers is for the effect of one year, meaning that if a text is released today, only its potential effect on stock performance in 1 year is assessed. Although most researchers rely on lexical-based methods, there are clear critiques for such approaches. Modern approaches, such as BERT, also have mixed findings for connections between stock performance and sentiment. While Chinese tweets do not seem to have any effect on stock prices, the sentiment of news articles seems to increase stock price forecast accuracy for a handful of Taiwanese stocks. In terms of complexity, research indicates that less complex texts might lead to more investment activity and also that the reporting for positive performance is written in easier language than in the case of negative earnings. For this literature review, no results for Swiss companies were found. Another identified trend in literature is that research is almost exclusively conducted for large cap companies.

4 Model Methodology

This chapter describes the steps and methods in creating the sentiment and complexity models for this paper. Note that this only introduces the relevant information for the creation and performance evaluation of the two models. The data gathering for the actual news articles and earnings reports is covered in section 5.

4.1 Complexity Model

The implementation of the Flesch score and Wiener Sachtextformel in code rely on libraries that are specialized on natural language processing. They help the user in defining what a sentence, a syllable, a word, or even a single character is. Both formulas require the number of words, sentences and syllables for the calculation of their respective complexity score. No additional dataset or similar is required, as only the formulas must be implemented in code.

4.1.1 Implementation of the Flesch Score

Due to the popularity of the Flesch Score it can be directly imported and implemented in its German version from the textstat library for Python. Therefore, no coding of the function by hand is required. Accordingly, its implementation only takes a few lines of code as shown in figure 11.

```
import textstat
#textstat documentation under: https://pypi.org/project/textstat/
text = "Heute ist ein guter Tag"

textstat.set_lang('de') #setting language to german

def apply_flesch(text): #write function so that i can apply it to a pandas dataframe
    flesch_score = textstat.flesch_reading_ease(text)
    return flesch_score
```

Figure 11 Code required for the textstat implementation of the Flesch score (own creation)

The Flesch score function from text stat is used in a function, so that each single sentence in a given data frame can be evaluated. With that, the Python implementation of the Flesch score is concluded.

4.1.2 Implementation of the Wiener Sachtextformel

The Wiener Sachtextformel was implemented using the Pyphen and *re* (Regular expression operations) libraries. The implementation for the various required Python functions was largely in-

spired by Pablo Theissen (Theissen, 2017). An example of the required function is shown in figure 12 where the counting methods for sentences and syllables are defined using the *re* library.

```

6 import pyphen
7 import re
8 from collections import Counter, defaultdict
9
10 #Demotext to test functionality
11 demotext = "Die Unternehmen zeigten allesamt Verluste von bis 2 Millionen Euro
12
13 #function counts the nr. of sentences
14 def count_sentences_german(text):
15     #Count number of sentences by counting end-dots but avoid counting
16     #abbreviations like »u.a.«
17     text = re.sub("[^a-zA-ZäöüÄÖÜ\.\?! ]", " ", text)
18     return len(re.findall(
19         "[a-zäöüß]{3}[\.\?!][\n\s]", text)) + 1
20
21 #function counts the nr. of syllables
22 def count_syllables_german(text):
23     """ Count number of syllables:
24     1. Clean up text
25     2. Split text into separate words separated by a space
26     3. Hyphenate every single word with pyphen
27     4. Count number of words + number of hyphens
28     """
29     dic = pyphen.Pyphen(lang='de_DE')
30
31     # Option to filter out certain characters from the text
32     # text = re.sub("[--\.,;'\(\)»«</>0-9]", " ", text)
33     text = re.sub("[^a-zA-ZäöüÄÖÜß ]", " ", text) # special chars/numbers
34     # text = re.sub("[A-ZÄÖÜ]+s", " ", text) # One-character words
35     # text = re.sub("[a-zäöüß]+s", " ", text) # One-character words
36     text = re.sub("\s+", " ", text)
37

```

Figure 12 A code snippet of the implementation for the Wiener Sachtextformel (own creation)

To count the number of syllables, additional counting functions for the number of words and number of characters are programmed. With these, the required inputs for the Wiener Sachtextformel calculation, as described in chapter 2.3.2, are defined and the respective formula can be implemented to create the score for the Wiener Sachtextformel. Lastly, another function is written so that the calculations can be applied on any sentence in a given data frame. Accordingly, apart from small changes for the Wiener Sachtextformel regarding Umlaute (ä, ö, ü), the implementation of these calculations is straight forward.

The written code can take any German sentence as an input and calculate a score for each, the Flesch score and Wiener Sachtextformel, as per the equations given in section 2.3. Apart from defining and applying the formulas in code, there is no model training or similar required as compared to the much more complex sentiment model.

4.2 Sentiment Model

The description follows the traditional ML process of gathering, cleaning and preparing the required data, followed by training the models and lastly testing them to validate the results. The basis for the created model in this paper is a distilled version of BERT (DistilBERT). It is trained

to be able to predict the sentiment of German sentences, where the sentiment can be either negative, neutral or positive. This sentiment range can be reflected with any choice of integers, such as [0, 1, 2] or [-1, 0, 1]. The task at hand is therefore to be interpreted as a supervised multi-label classification problem.

4.2.1 Data Gathering

For the fine-tuning of the DistilBERT model, additional data was required to allow the model to “specialize” itself on the sentiment analysis of financial texts. While there are several large, publicly available sentiment-annotated datasets, they are mostly related to products or movies and in English. In the case of finance and economic texts, annotated collections are a scarce resource, and many are reserved for proprietary use only. The finance world has its own jargon and therefore financial texts contain specific words that would not normally be related to any sentiment. Examples are bullish and bearish, which reflect either a positive or negative outlook on the market and could therefore also be interpreted as positive and negative sentiment respectively. The model must be faced with such texts in a training phase before conducting predictions on previously unseen financial sentences. The necessity and resulting advantage in better model performance is demonstrated by Araci (2019) with the FinBERT model. The training dataset from that paper, named financial phrase databank, is freely available and thus was also used for the purpose of this paper. It consists of roughly 5’000 sentences, representing the headlines of news articles of multiple English-speaking newspapers, that were categorized by up to eight annotators into one of the three categories: positive, neutral or negative. The annotators were either the researchers or master’s students with a finance, accounting or economics background. The categorization is based from the view of an investor, i.e. if the text would have a positive, neutral or negative effect on the stock price of a company. The corpus consists of English news on all listed companies in OMX Helsinki (Malo, Sinha, Korhonen, J., & Takala, 2014). The dataset is available in four possible configurations depending on the percentage of agreement of annotators:

Table 4 Available datasets from the Financial phrase databank as per (Malo, Sinha, Korhonen, J., & Takala, 2014)

Dataset Name	Description	Sentence count
sentences_50agree	Number of instances with $\geq 50\%$ annotator agreement	4’846
sentences_66agree	Number of instances with $\geq 66\%$ annotator agreement	4’217
sentences_75agree	Number of instances with $\geq 75\%$ annotator agreement	3’453
sentences_allagree	Number of instances with 100% annotator agreement	2’264

For the training of the DistilBERT model, a total of 3'453 sentences where 75% of annotators agreed on the categorization was taken, as it reflects the best trade-off between number of sentences and categorization quality. These differences in annotator agreement display again, that even humans do not fully agree with their own classifications, as already mentioned in chapter 2.1.4. The otherwise unprocessed English sentences were then translated into German using the DeepL translation service (DeepL, 2021). An example of one of the original sentences together with its German translation is given below:

«Valmet Automotive reports net sales of EUR 85mn and operating profit of EUR 8mn. »

«Valmet Automotive meldet einen Nettoumsatz von EUR 85 Mio. und einen Betriebsgewinn von EUR 8 Mio.»

In its original form, the financial phrase databank has labeled the three categories with the strings 'negative', 'neutral', and 'positive'. To apply these to the model they were translated into the corresponding integers {'negative' : 0, 'neutral' : 1, 'positive' : 2}. The dataset is imbalanced with regard to its classes, where the negative class appears 420 times, the neutral class 2'146, and the positive class 887 times.

4.2.2 Data pre-processing

For the data pre-processing, it was tested if the model would respond to any lowercasing or stopword removal. Given that the dataset consists of articles written by journalists, which is to be interpreted as a high-quality text, there is no cleaning in terms of spelling mistakes or similar required.

For lowercasing, it can be stated that BERT models, or also their distilled versions called DistilBERT, generally exist in cased or uncased version. Cased and uncased relate to the case-sensitivity of the model. If a model was only trained on texts in lowercase, meaning that all capital letters are exchanged to lowercase letters (e.g. E is turned into e), it is considered to be an uncased model. If capitalization was left in the training data, it is therefore called a cased model. Using an uncased model reduces vocabulary size, however some distinctions might be lost (e.g. "Apple" the company vs "apple" the fruit is a commonly used example). Depending on the language, capitalization can therefore have important impacts on word meaning. This is especially true for German, where nouns, names or substantiated verbs or adjectives are written with a capi-

talized first letter. Accordingly, a cased model was chosen for the model of this paper. Given that the model was trained on non-lowercased data, it would be reasonable that financial phrase data-bank used for fine-tuning also wouldn't require any lowercasing. The same reasoning applies to the removal of stopwords.

To confirm this, the effects of lowercasing and stopword removal on model performance were tested in two ways: In one try the stopwords of the input data (the financial phrase training set of the 3'451 news articles used for fine-tuning) were removed and then the model was trained on that data. The model's sentiment classification predictions were then applied on the previously unseen news articles, which also had their stopwords removed. The second try did not apply any pre-processing to the training data set and only removed the stopwords from the testing data. The same process was conducted separately for lowercasing. As a benchmark for comparison a model with absolutely no pre-processing, whether on the training, nor on the testing data was used.

It showed that the first and second method resulted in almost no performance differences between each other. For lowercasing, this process resulted in an accuracy score that was on average 2% worse compared to the completely unprocessed data, while the application of stopword removal together with lowercasing even lead to an average decrease in performance of 3%. These result support the exclusion of lowercasing and stopword removal. Accordingly, no other pre-processing methods were tested as the model seems to perform better on natural language with all its details included. This would also make sense given that the underlying German DistilBERT model is trained on the unprocessed, raw language German Wikipedia dump (6GB of raw txt files), the OpenLegalData dump (2.4 GB) and news articles (deepset.ai, 2019). The overall effect of lowercasing and stopwords is displayed below in table 5.

Table 5 Pre-processing performance comparison (own creation)

Model Pre-processing	MCC	Eval_loss	F1-Score
No pre-processing	0.8708	0.3097	0.9305
Average Lowercasing	0.8501	0.3476	0.9189
Average Lowercasing & Stopword removal	0.8196	0.3611	0.9071

4.2.3 Model architecture

This section covers the model choice, its implementation in Python and the tuning of the relevant hyperparameters. As mentioned above, the basis for the created model in this paper is a distilled, cased version of the open source German BERT model that was created and trained from scratch by deepset (deepset.ai, 2019). The training for the original German BERT model on the above-mentioned data inputs took roughly nine days and the researchers displayed state of the art performance on multiple datasets as shown in figure 13.

Model	germEval18Fine	germEval18Coarse	germEval14	CONLL03	10kGNAD
multilingual cased	0.441	0.71	0.834	0.792	0.888
multilingual uncased	0.461	0.731	0.823	0.786	0.901
German BERT cased (ours)	0.488	0.747	0.84	0.804	0.905

Figure 13 German BERT performance (deepset.ai, 2019)

As laid out in section 4.2.1 and 4.2.2, the cased, distilled version was then trained on additional financial texts to improve its performance. The concept of taking a pre-trained model and adding more information to it, allowing it to specialize itself, is called Transfer Learning. It can be generalized by describing it as a method that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem (West, Venutra, & Warnick, 2007). Accordingly, with this model the author tries to keep the German DistilBERT's capabilities of understanding and interpreting normal German texts, while also finetuning its understanding for financial texts. The ease with which one can finetune BERT is one of its main benefits and was accordingly mentioned in section 2.2.2.

The training for the model is implemented with the simpletransformers library in Python. The code for this project was written in the Google Colab environment. The platform is similar to a Jupyter Notebook but allows the user to access external computational resources, namely GPUs, for free. This increased the calculation speed for this model roughly 13-fold. However, even with this drastic increase in speed, a German BERT model took 21 minutes to calculate, which is also why the faster distilled version was chosen. Before feeding the data into the model, the 3'451 sentences from the financial phrase databank are split randomly into a training and testing dataset with a 80 / 20 split. The training data is the further split into actual training data and validation

data using stratified 5-fold cross validation. Stratified cross validation splits the training data set into five different combinations of training and validation datasets, where the respective representation of the three different classes is taken into account. This step is therefore crucial for the evaluation of the model's ability to generalize on previously unseen data, especially given the previously mentioned class imbalance of the dataset. The previously split-off testing dataset, containing 20 percent of the original data, is not affected by this cross-validation split. The model is defined as shown in figure 14.

```
#RUN WITH NO OPTIMIZATION

#CV split with stratification
for train_index, test_index in kf.split(X, y):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    train_df = pd.DataFrame(X_train, columns = ['text'])
    train_df['labels'] = pd.DataFrame(y_train)

    val_df = pd.DataFrame(X_test, columns = ['text'])
    val_df['labels'] = pd.DataFrame(y_test)

    # Create a ClassificationModel
    model = ClassificationModel(
        'distilbert', #'bert'
        'distilbert-base-german-cased', #'bert-base-german-dbmdz-cased'
        use_cuda=True,
        num_labels=3,
        args=model_args,
        #sweep_config=wandb.config,
    )

    # Train the model
    model.train_model(train_df, eval_df=val_df)

    # Evaluate the model
    result, model_outputs, wrong_predictions = model.eval_model(val_df, acc=sklearn.metrics.accuracy_score)

    results.append(result['acc'])
```

Figure 14 The DistilBERT model (own creation)

To find the best hyperparameters for the model, Bayesian hyperparameter optimization is applied. Bayesian optimization is a method for finding the minimum of a function, which in this case is defined as the valuation loss. It differs from other procedures in that it constructs a probabilistic model for $f(x)$ and then exploits this model to make decisions about where in X to next evaluate the function, while integrating out uncertainty. The essential philosophy is to use all of the information available from previous evaluations of $f(x)$. This results in a procedure that can find the minimum of difficult non-convex functions with relatively few evaluations, at the cost of performing more computation to determine the next point to try (Snoek, Larochelle, & Adams,

2012, p. 2952). This computational power requirement is again why DistilBERT was chosen over BERT. Using this optimization method, a total of 53 models were trained, all using constantly adapting hyperparameters. The hyperparameters for the DistilBERT model consist of the number of epochs, batch size and the learning rate. The range for the epochs was between [3, 4, 5, 6], while the learning rate was allowed to fluctuate between a minimum of 0.00002 and a maximum of 0.0004. Batch size was kept constant at 16 and 8 for training and testing runs. Poorly performing optimization runs were stopped after six continuously bad runs. A bad run is when the to be minimized value, which in this case is the valuation loss, does no longer decrease. Out of these 53 models, the hyperparameters of the best performing model, as measured by the highest Matthews correlation coefficient (MCC), were selected for the training of the final model. The hyperparameters that lead to the best result were 5 epochs with a learning rate of 2.9141049808550304e-05 (0.000029). The figure 15 highlights the most important parameters for the trained models. The learning rate was identified as the most important parameter, with a correlation to MCC of -0.905. Accordingly, the lower the learning rate is, the higher the MCC. The correlation score for the total runtime was 0.657, indicating that the model actually improved the longer it was able to train itself. Lastly, the number of epochs was less important for the overall model, but nonetheless had a correlation with MCC of 0.441.

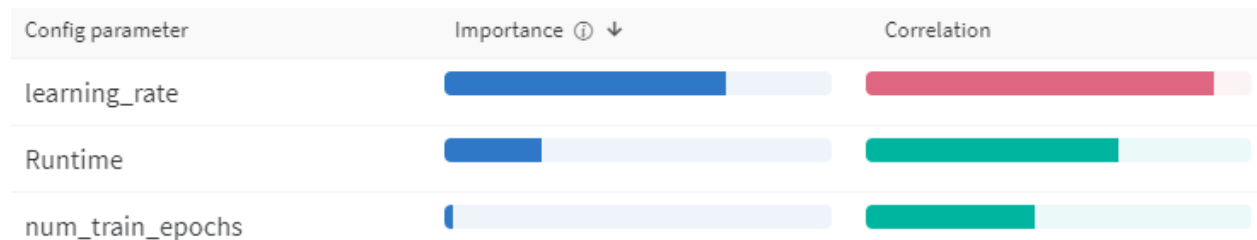


Figure 15 Most important parameters for a high MCC (own creation)

These relationships are also made visible in the below figure 16. Each curve, colored from purple to orange, shows which learning rate, with which number of training epochs lead to which overall MCC score for a single model. It clearly depicts that the combination of low learning rates and higher number of epochs, namely, 5, lead to the best results.

Analysis of Sentiment and Complexity of News and Earnings Reports of Swiss SMEs and their Relevance for Stock Returns

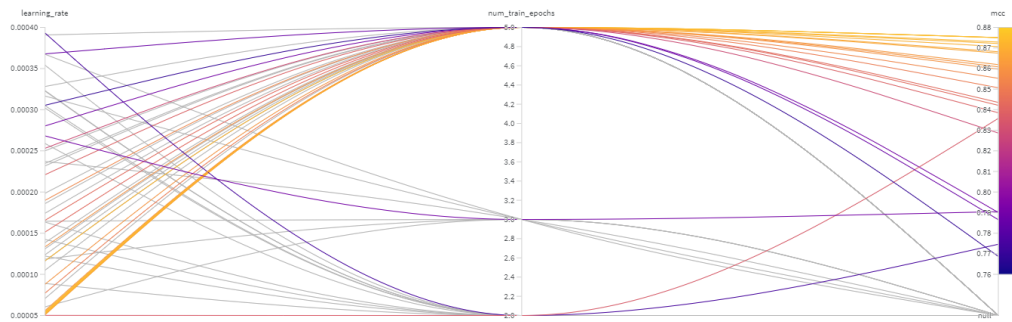


Figure 16 Connections between learning rate, number of epochs and MCC (own creation)

The Bayesian optimization clearly registers these relationships as shown in figure 17, which depicts a selection of the first 23 models. The y-axis in the graph is the learning rate and the x-axis is the training steps. A training step is one gradient update, where 16 examples, based on the batch size, are processed. All models consistently decrease the learning rate as this relates to finding better results. For a few models, one can also see either the different run times based on number of epochs or an early stopping due to bad model performance. This effect also shows for the other models; however, they were excluded from the graphic due to cluttering.

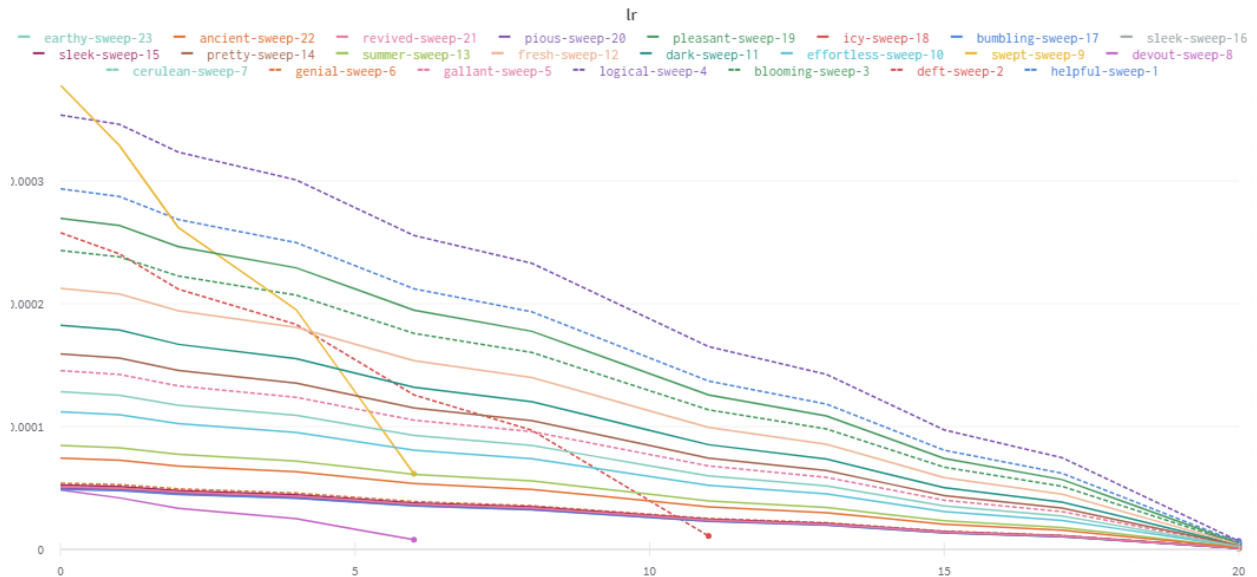


Figure 17 A constant decrease of learning rates for the first 23 models (own creation)

The training for a single model using the optimal parameters takes around 12 minutes thanks to the additional GPU provided by Google Colab. Having assigned the best hyperparameters to the model it can be trained and evaluated.

4.2.4 Model evaluation

The results from evaluating the model on the 5 stratified cross validation folds were averaged to receive a single score. The model was first tested on the validation dataset and then finally on the previously unseen testing dataset. The average accuracy on the validation dataset was at 0.9, indicating that 90 percent of all classifications were correct, with a very low standard deviation of 0.008. This indicates a very stable performance on all different folds. These results are displayed in figure 18.

```
Results per Fold:
Fold-1: 0.9023508137432188
Fold-2: 0.8987341772151899
Fold-3: 0.9112318840579711
Fold-4: 0.9239130434782609
Fold-5: 0.9130434782608695

5-fold CV accuracy result:

Mean: 0.909854679351102, Standard deviation:0.00882809492517318
```

Figure 18 Cross validation results (own creation)

However, for the actual evaluation of the model performance, the results on the testing dataset are more interesting and are accordingly displayed in more detail. The model achieves the following scores as shown in figure 19.

```
{'f1': 0.9131693198263386,
 'mcc': 0.8388243230892787,
 'eval_loss': 0.2787486110578409}
```

Figure 19 Results on the testing dataset (own creation)

These are to be interpreted as a very satisfying result. However, for the specific task of applying the trained model to news articles and earnings reports it is important to minimize the number of grave misclassifications in the model. For this task, a misclassification where a positive sentence is classified as a neutral sentence, is less grave than if it would be misclassified as a negative sentence, as this represents the complete opposite of the sentiment. This also applies vice versa. Accordingly, the differences between the values for precision and recall are important. They are

best interpreted in combination with a confusion matrix, where one should focus on the top right and bottom left corner of the matrix. Both are shown in figure 20 and 21 below. As comparison in terms of performance, the baseline performance of predicting only the most represented class, the neutral sentences, would be 62 % accuracy.

	precision	recall	f1-score	support
negative	0.90	0.79	0.84	98
neutral	0.95	0.95	0.95	429
positive	0.82	0.89	0.86	164
accuracy			0.91	691
macro avg	0.89	0.88	0.88	691
weighted avg	0.91	0.91	0.91	691

Figure 20 Classification report for precision, recall and F1-Score (own creation)

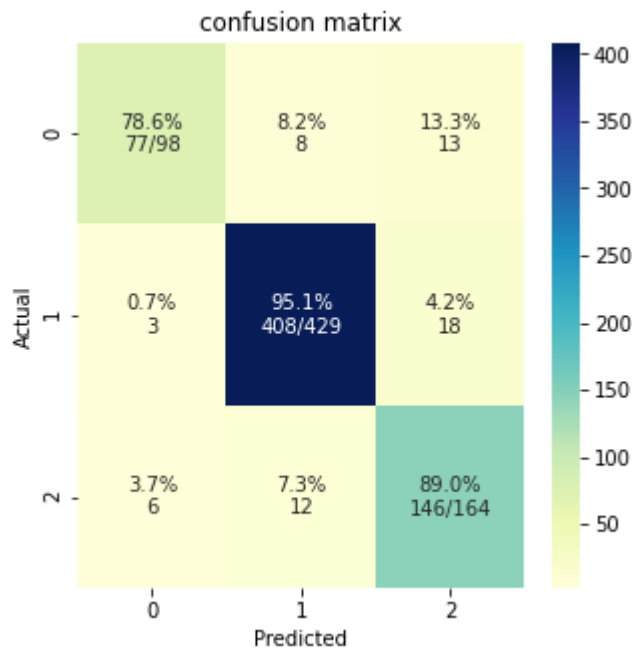


Figure 21 The confusion matrix for the model with labels 0 : negative, 1 : neutral, 2 : positive (own creation)

Precision can be seen as a measure of quality, while recall can be interpreted as a measure of quantity. For the negative class, the model identifies 78.6 % of all available negative sentences, which represents its recall. For the precision of the negative class, 90 % of predicted negative sentences where actually part of the negative class. For the positive class, 89 % of all available

positive sentences were identified, while the model tends to classify more sentences into the positive category than it should, with a precision score of 0.82. On an F1-score level, this leads to similar results for the model's prediction abilities on the negative and positive class. The neutral sentences are identified in a balanced way, with both precision and recall standing at 0.95. Overall, the model predicts very little positive sentences as negatives (3.7 %), but the amount of actual negative sentences predicted as positives is higher (13.3 %). Accordingly, the overall performance on the sentences with negative sentiment was the worst, but still within a very useable range of accuracy. It is assumed that this mainly lies in the initially unbalanced classes of the financial phrase databank. Another reason might be that sentences with negative sentiment could be more complex to interpret in general. This assumption will be analyzed in more detail in chapter 5 of this paper. Nonetheless, the overall result is very good and on par with the performance of papers in literature. The model is therefore accepted for further use on the to be introduced earnings reports and news articles.

Finally, to display the output of the trained model, it was provided with four, self-created, new sentences that were then classified. These are displayed in figure 22. Each of the four sentences was assigned an integer, ranging from [0, 1, 2] and meaning negative, neutral and positive. The first sentence "*Der Umsatz ist gesunken, jedoch wurde dabei ein grosser Gewinn von 500 Mio. erzielt*", was classified as negative. The sentence contains both positive (grosser Gewinn erzielt) and negative (Umsatz ist gesunken) aspects and is therefore not easy to classify, especially from the view of an investor. One could however argue that profit (Gewinn) would be more important than sales (Umsatz) from a financial standpoint and thus the sentence should have been interpreted as a positive. This displays again that even for humans, a clear classification is not always simple, given that one sentence can contain both, negative and positive sentiment. The other three, simpler sentences were all classified correctly into their respective categories, at least based on how the author would interpret them. Note that while the sentence "*Das Wetter ist schön*" could in normal circumstances be interpreted as something positive, it hardly has any impact on the financial world and thus it was (correctly) classified as a neutral sentence. The third and fourth sentence contained either clearly negative or clearly positive sentiment.

Natural Language Processing in Finance:

Analysis of Sentiment and Complexity of News and Earnings Reports of Swiss SMEs and their Relevance for Stock Returns

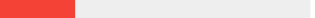

```
predicted = model.predict(["Der Umsatz ist gesunken, jedoch wurde dabei ein grosser Gewinn von 500 Mio. erzielt",  
                           "Das Wetter ist schön",  
                           "Im Vergleich zum Vorjahr wurden starke Verluste erlitten",  
                           "Das Unternehmen hat einen starken Gewinn erzielt"])  
  
predicted[0]  
  
INFO:simpletransformers.classification.classification_utils: Converting to features started. Cache is not used.  
25%  1/4 [00:00<00:00, 11.51it/s]  
100%  1/1 [00:00<00:00, 20.24it/s]  
[0, 1, 0, 2]
```

Figure 22 Using the trained model to classify four sentences (own creation)

This trained model can now be used to categorize any German sentence into one of the three sentiment categories with a useful accuracy.

5 Analysis

This chapter introduces the process for gathering, cleaning and evaluating news articles and earnings reports for Swiss companies on their sentiment and complexity and therefore follows the traditional ML process as described in section 2.1.

5.1 Data gathering

As described in the introduction to this paper, the focus will only be set on Swiss exchange-listed small- and mid cap companies. There are no official capital sizes given for Swiss companies by which they could be separated into these segments, however the SIX Swiss Exchange, which governs the infrastructure of the Swiss financial exchange, creates indices for the largest companies. The 20 largest Swiss stocks are aggregated in the Swiss Market Index (SMI). They are not in scope for this paper. The 30 largest companies after the SMI constituents are considered mid caps and are part of the SMI MID Index (SMIM) (SIX Swiss Exchange, 2021). After those, total 50 largest Swiss companies, every company that has a smaller free-float market capital is considered to be a small cap.

For this paper, 15 companies were selected as shown in table 6. The selection criteria for a company was as follows:

- 1) The company must be publicly listed on the Swiss stock exchange since 2016. The general time frame for this review is 1st of June 2016 to 30th May 2021.
- 2) The company must be available to trade in terms of daily liquidity (volume) and its trading data must be available on Yahoo Finance.
- 3) The company must complete its reporting in German and must publish regular updates to its economic performance, namely earnings reports, e.g. in the form of quarterly and annual performance updates. While these publications must be available in German, additional availability of the texts in other languages is not a criterion for exclusion. The text of the reports must be copyable, as it showed that certain companies publish their reports as pictures within an PDF.
- 4) The news articles about the company must be freely accessible and available for the given timeframe, without any paywalls or similar constraints. The reports must be published before the opening of the stock exchange (9 AM Swiss time).

Table 6 Selected companies

Company	Ticker	Market capital in Mio. CHF	Category
Straumann	STMN	22'432.00	Mid cap
Sonova	SOON	19'631.00	Mid cap
VAT Group	VACN	8'314.00	Mid cap
Tecan	TECN	5'150.00	Mid cap
OC Oerlikon	OERL	3'457.00	Mid cap
Bucher Industries	BUCN	5'133.00	Small cap
EMMI	EMMN	4'962.00	Small cap
Belimo Holding	BEAN	4'791.00	Small cap
Sulzer	SUN	3'942.00	Small cap
Bell Holding	BELL	1'786.00	Small cap
Burckhardt	BCHN	1'263.00	Small cap
Arbonia	ARBN	1'198.00	Small cap
Zehnder Group AG	ZEHN	972.00	Small cap
Feintool	FTON	310.00	Small cap
Calida	CALN	297.00	Small cap

The split-up into the small- and mid cap categories was made as per 04.06.2021 and is subject to change over time, as market capitalizations change if the stock price changes. It must be noted that the market capital in table 6 reflects the overall market capital of the company, while the classification of SIX regarding mid and small cap only relates to free-floating market capital. Therefore, it is possible for a small cap to have a higher overall market capital than a mid cap company. With over 217 companies listed on Swiss stock exchange (SIX Swiss Exchange, 2021), the amount of selected stocks might seem rather small. However, mainly the second and third criteria as defined above lead to the exclusion of many companies. Most firms had to be excluded due to their lackluster reporting quality, as not many companies created useful and informative media releases. Also, given by the fact that most Swiss companies have a foreign focus, their reporting and media releases are usually created in the English language. Swiss firms are required to create their reporting in at least one of the countries languages, while English is allowed as additional reporting language. Hence, any firm located in the French or Italian speaking region of Switzerland is unlikely to create media releases in German. Lastly, the gathering of data had to be done by hand, which is an extremely time intensive process. Thus, the selection

was also limited in that aspect. All media releases were downloaded from the respective investor relations websites of the corresponding companies. An average of 14 media releases per company was analyzed. Only media releases that stood in connection to an earnings report (e.g. the release of either annual earnings or quarterly earnings) were considered. Accordingly, the media releases were split-up into two categories: annual earnings reports and quarterly earnings reports.

For all of the above-named companies, news articles created by Swiss media between 1st of January 2016 and 30th of May 2021 were gathered manually from the website “cash.ch”. Cash is one of the leading websites for Swiss financial news and averages 4.63 million visitors per day (Similarweb, 2021). Its news platform covers most publicly listed Swiss companies, especially around earnings dates. The data for their news articles is gathered from different services, such as AWP, which is the leading financial news agency in Switzerland (AWP, 2021). However, AWP does not offer a free access to historical news articles, hence cash.ch provides a clear benefit to users by making access to older news articles possible. That is also where other evaluated news providers such as nzz.ch, handelszeitung.ch, bilanz.ch, fuw.ch, finanzen.ch or finews.ch lack accessibility or broad firm coverage. Therefore, these websites were not considered for this paper. An average of 30 news articles per company was gathered from cash.ch and put into a usable .txt format. A limit of three news articles for one company on a specific day was set. Only material news articles that were relevant for stock price development were selected (e.g. the lay-off of 1’000 employees or the start of a new CEO). An overview of the gathered news articles and earnings reports is given below in table 7.

Table 7 Overview of gathered news and earnings reports

Size	Aggregated Type	Detailed Type	Count of reports	Aggregated count of reports
Mid cap	Earnings report	Annual Earnings reports	26	66
		Quarterly Earnings reports	40	
	News	News	156	156
Small cap	Earnings report	Annual Earnings reports	64	141
		Quarterly Earnings reports	77	
	News	News	307	307
Sum				670

As mentioned, this process was conducted manually as the website's search results are cluttered with unimportant information for this review and therefore human input for the selection of articles was required. Therefore, no web scraping bot could be programmed to conduct this selection automatically.

The financial data for this paper stems exclusively from Yahoo Finance. The data is gathered via the `yfinance` library for Python, which is a downloader for the market data of the Yahoo! Finance platform (`yfinance`, 2021). The library allows to directly access all of Yahoo Finance's publicly available data such as historical stock performance, balance sheets or dividends.

5.2 Data pre-processing

The gathered news articles and media releases consist of multiple sentence in an aggregated format. To make use of them in a model, they were tokenized into single sentences using the NLTK tokenizer function, which has a special implementation for the German language. This is done as both the sentiment and text complexity work on a sentence level. Accordingly, the 670 news articles and earnings report were split into 20'557 sentences, averaging a count of roughly 30 sentences per text.

For both, earnings reports and news articles, white space removal was applied, where 14'421 unnecessary white spaces were removed. Numbers were not removed from the text, as it showed that the sentiment model is able to classify numbers into a sentiment categorization if there was a plus or minus in front of the number. To make use of this effect, numbers were left in the dataset.

It was identified that the trained DistilBERT model had the same results in predicting sentences when sentences were either written normally (e.g. "*Umsatzsteigerung in den ersten 9 Monaten 2018 um 11%*")., or in lowercase (e.g. "*umsatzsteigerung in den ersten 9 monaten 2018 um 11%*")., or were written without stopwords and numbers (e.g. "*umsatzsteigerung ersten monaten.*"). However, the model had different predictions if a sentence was completely capitalized (e.g. "*UMSATZSTEIGERUNG IN DEN ERSTEN 9 MONATEN 2018 UM 11%*"). Accordingly, fully capitalized sentences were removed from the gathered news articles and media releases. An example of this effect is given in figure 23, where the unprocessed text leads to a positive sentiment classification (`array([2])`), while the capitalized text is categorized as a neutral text (`array([1])`), which is incorrect.


```

model.predict(['Umsatzsteigerung in den ersten 9 Monaten 2018 um 11%.'])[0]
INFO:simpletransformers.classification.classification_utils: Converting to features started. Cache is not used.
100% ██████████ 1/1 [00:00<00:00, 9.87it/s]
100% ██████████ 1/1 [00:00<00:00, 20.77it/s]
array([2])

model.predict(['UMSATZSTEIGERUNG IN DEN ERSTEN 9 MONATEN 2018 UM 11%'])[0]
INFO:simpletransformers.classification.classification_utils: Converting to features started. Cache is not used.
100% ██████████ 1/1 [00:00<00:00, 11.40it/s]
100% ██████████ 1/1 [00:00<00:00, 18.81it/s]
array([1])

```

Figure 23 The difference made in prediction in case of capitalization (own creation)

Apart from tokenization, white space removal and removal of full capitalization in sentences, no further pre-processing was conducted, given the insights from section 4.2.2.

5.3 Evaluation

After cleaning the gathered data, the complexity formulas were applied to every sentence, while also a sentiment prediction was created using the trained DistilBERT model. Accordingly, the data will now be separately analyzed on complexity and sentiment.

5.3.1 Complexity

Both, the Flesch score and Wiener Sachtextformel, were applied on all sentences and then aggregated back into report format to receive an average Flesch score and Wiener score for each of the news articles and earnings reports. They were then analyzed on overall measured complexity in different constellations such as complexity in small- and mid caps, over time, differences between earnings reports and news articles and lastly on the difference in complexity between sentiment classification. An aggregated overview of all datapoints is displayed in the appendix in section 8.1.

While the tokenization algorithm generally works very well, a few outliers were removed from the dataset, where the splitting into sentences did not work correctly. Reason being was usually the presence of quotation marks in the text. Other reasons for outliers are simply very short sentences. For example, the sentence *“Auch die ZKB und die UBS werten die Zahlen ähnlich.“* has a Flesch score of 93.95, while the sentence *“Dort waren Restrukturierungsmassnahmen angesagt.“* has a score of -28.76. The reason for the difference is that while both sentences are short, the second sentence contains one very long word (*Restrukturierungsmassnahmen*) which

heavily influences the overall complexity of the sentence due to the way that the formula works. However, opposite to the few mistakes in tokenization, these outliers were not removed as they were separated correctly and simply fall into the normal range of language diversity in terms of length and complexity. This however highlights one of the key sensitivities of the Flesch formula. The same analysis was conducted for the Wiener score, where the same sensitivity towards short sentences with long or complex words was identified. Additionally, the correlation of the Flesch and Wiener scores over all evaluated sentences was calculated to be -0.8667 . The correlation is negative due to the fact that a higher score in the Flesch formula indicates an easily understandable text, while for the Wiener score this would be represented by a low score. The correlation displays that both scores generally rate most sentences similarly from easy to hard to understand.

Overall, the text complexity for the Flesch score, where a lower number indicates higher text complexity, ranged from 10.5 to 63.5 on a sentence level and 10.6 to 63.5 on an aggregated report level. The average Flesch score per sentence was 40.1, whereas the median is at 40.9. Per the grading scale displayed in chapter 2.3.1 this score is in the third hardest category in terms of understanding and requires the understanding capabilities of college students and is therefore difficult to read. The histogram in figure 24 is negatively skewed to the left and displays a heavy left-tail, which represents the previously discussed outliers.

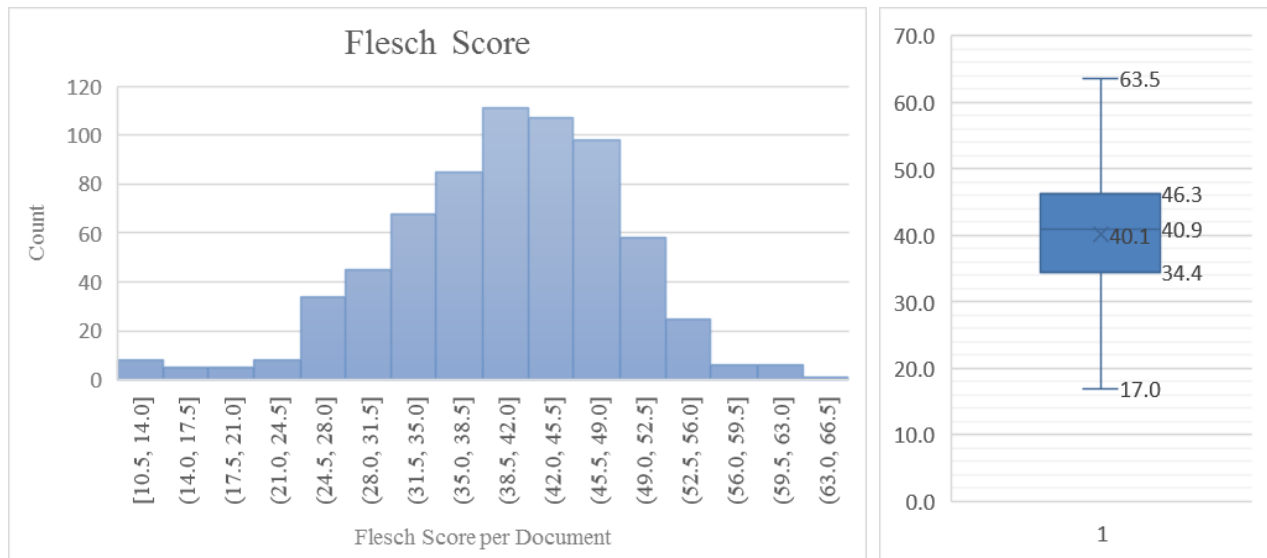


Figure 24 Distribution of Flesch scores going from hard to easy (left to right) in terms of understanding (own creation)

The overall text complexity for the Wiener score, where a higher number indicates higher text complexity, ranged from -7.2478 to 31.627 on a sentence level and 7.3 to 16.8 on an aggregated report level. The average Wiener score per aggregated document was 11.9, whereas the median was 11.8. In the absence of a grading scale for the Wiener Sachtextformel this is nonetheless to be interpreted as a general difficult rating in terms of reading ease, given that the highest possible score is at 14 and above. The histogram for the Wiener scores is given in Figure 25.

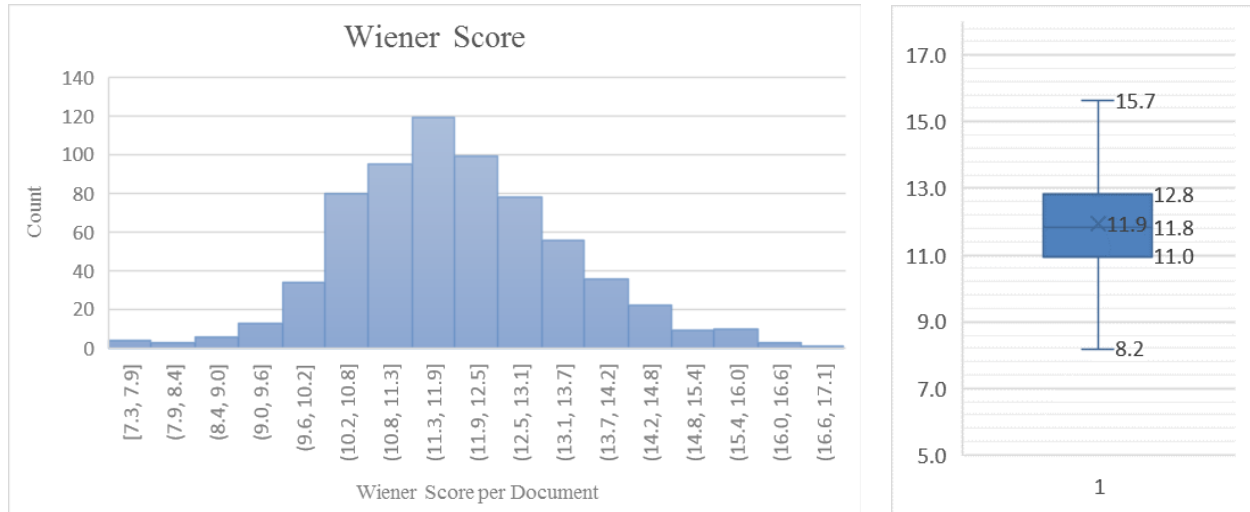


Figure 25 Distribution of Wiener scores going from easy to hard (left to right) in terms of understanding (own creation)

In terms of the difference between complexity for small and mid caps it can be said that textual reportings for small caps tend to be easier to understand than for mid caps. However, the difference between the two segments is very small, as displayed in table 8.

Table 8 Differences in complexity between small and mid caps (own creation)

Small or Mid cap	Wiener Score	Flesch Score
Mid cap	12.05945982	37.93408859
Small cap	11.95134227	39.70648962
Average	11.99361209	39.01354867

With these scores, both fall into the same category of difficulty in terms of complexity and can be classified as “hard” to understand. However, given that the textual database consists of news articles and earnings reports, it can be analyzed if there is a difference between the two mediums. It shows that news articles with a Flesch score of 43.7 and Wiener score of 11.4 are on average easier to understand than the earnings reports with a Flesch score of 35.5 and Wiener score of

12.4. The benefit of the Flesch score is that due to its rating scale ranging from 0 (very complex) to 100 (very easy), a change in one point can be interpreted as a percentage change in difficulty. In that regard, it can be said that the analyzed earnings reports were roughly 8 percent harder to understand than the news articles, based on the Flesch score. These results are displayed in table 9 below.

Table 9 Differences in complexity between earnings reports and news articles (own creation)

Earning or News	Flesch Score	Wiener Score
Earnings	35.51281475	12.40074877
News	43.7260751	11.44554348
Average	39.01354867	11.99361209

To further expand on the differences in complexity for earnings reports and news articles, it also shows that the complexity for earnings reports has increased on average over the past 5 years, especially in 2021 where complexity scores are at a 5-year high. This effect is mostly visible for small caps, whereas mid caps remained more stable. This effect is highlighted in the below table 10, where the Flesch score for small caps displays a somewhat consistent reduction from 37 down to around 33. In terms of Wiener score, this represents an increase from class 12 to class 13, thus increasing complexity by one level.

Table 10 Development of complexity over the past 5 years (own creation)

Score	Small or Mid Cap	Earnings or News	2016	2017	2018	2019	2020	2021
Flesch	Mid Cap	Aggregated Earnings and News	35.530	38.006	37.351	37.673	39.152	38.905
		Earnings	35.745	33.317	34.914	35.845	34.785	35.191
		News	32.321	43.972	42.346	42.390	44.049	43.814
	Small Cap	Aggregated Earnings and News	37.604	42.320	40.609	39.600	37.994	37.448
		Earnings	36.580	37.488	36.927	36.849	33.722	33.162
		News	44.220	46.166	43.922	45.308	41.678	42.615
Wiener	Mid Cap	Aggregated Earnings and News	12.330	12.000	12.104	12.080	11.996	11.941
		Earnings	12.277	12.611	12.332	12.183	12.424	12.287
		News	13.120	11.222	11.637	11.813	11.516	11.483
	Small Cap	Aggregated Earnings and News	12.264	11.468	11.770	11.973	12.267	12.436
		Earnings	12.339	12.103	12.226	12.303	12.859	13.001
		News	11.774	10.964	11.359	11.288	11.756	11.756

While a change in complexity of earning reports deems interesting, the fluctuating values for the news articles are less surprising. Given that all articles come from the same source it is expected that cash.ch has some sort of writing minimum standard, which would also affect text complexity. While it is possible to identify slight trends in the data, all changes are generally very small and in the range of a few percentage points. Consider the big picture this would indicate that there is not a lot of variation and change in text complexity, be it over the past 5 years, for small- and mid caps or for news articles and earnings reports.

Lastly, it shall be evaluated if there is any identifiable connection between the sentiment of a text and its complexity. In chapter 4.2.4 it was identified that the overall performance of the sentiment model was worse for sentences with negative sentiment. Accordingly, it was assumed that this could be due to higher complexity of negative sentences. It shows for both, small- and mid caps, that neutral sentences are easier to understand (less complex) than sentences with negative or positive sentiment. For the mid caps, positive sentences were more complex than negative sentences. This effect is opposite for small caps, where negative sentences were more complex than positive ones. However, considering the overall difference in the given numbers in table 11, the differences are at maximum a few percent, e.g. the Flesch score for small caps is 38.65 for negative sentences and 39.46 for positive sentences. This would indicate that the negative sentences were roughly one percent more complex and therefore very slightly harder to understand. Due to these low differences, no clear trend for negative and positive sentences can be identified and accordingly the estimation made in chapter 4.2.4 cannot be confirmed.

Table 11 Comparison of complexity per sentiment class

Score	Small or Mid Cap	Negative Sentiment	Neutral Sentiment	Positive Sentiment
Flesch	Mid Cap	37.56197507	40.98635389	36.57940917
	Small Cap	38.65730805	40.86471349	39.46592811
Wiener	Mid Cap	12.10412758	11.52858598	12.29947394
	Small Cap	12.12782387	11.73339994	12.00311862

To summarize the complexity analysis it can be said that while differences and trends between complexity over the years, between small and mid caps and news and earnings reports can be identified, they are infinitesimal and therefore do not seem bring any additional insight into the

interaction of sentiment and complexity. While even very small differences in complexity could be statistically significant, the idea that a text with mostly negative sentiment, for example an earnings report, would be harder to understand is unrealistic, as such small differences in complexity are not really noticeable for human readers, or at least not for the author during a self-assessment. Also, further analysis with text complexity in terms of its potential of effect on volume or earnings, as suggested by Li (2008), seems unrealistic, considering that the interquartile range (IQR) given by the box-whisker plots in figures 21 and 22 is 11.9 for the Flesch score and 1.8 for the Wiener score and therefore quite the low. The low, or narrow, IQR also supports the statement that actual differences in complexity between the analyzed texts are hardly noticeable to human readers, as the maximum difference in text complexity for most of the Swiss news articles and earnings reports is 12 percent, based on the IQR of the Flesch score. As such differences in text complexity are not really noticeable, no further evaluation in terms of stock price connection was made. Having realized that the human ability to accurately estimate small differences in text complexity in a thorough self-test is very limited, it would be unrealistic that such differences would be priced into the valuation of a stock. Therefore, a focus will be set on the sentiment variables which can be better evaluated by humans.

5.3.2 Sentiment

This section will first establish the actual accuracy of the created DistilBERT model on the created classifications for every sentence of the gathered news articles and earnings reports. Then the connections between stock performance around the publication date of the gathered news articles and earnings reports are analyzed. Lastly, multiple ML models are created to analyze the potential connection between sentiment and stock performance from another, non-linear point of view.

5.3.2.1 Sample Testing

So far, the sentiment model using DistilBERT was only tested on the financial phrase dataset. Given that the sentences gathered from the news articles and earnings reports represent a complete out of sample application of the model, the result are evaluated again. This was done by manually evaluating a sample of 377 of the total 20'557 sentences. The sample size calculations were made with a confidence level of 0.95 and a margin of error of 5% using the following calculation (Taherdoost, 2018):

$$Sample\ size = \frac{\frac{z^2 * p (1 - p)}{e^2}}{1 + \left(\frac{z^2 * p (1 - p)}{e^2 N}\right)} \quad (14)$$

Where z is the z-score, e the margin of error, p is 0.5 and N the population of 20'557. This results in a sample size of 377. To be able to make a valid statement for the models out of sample performance for all classes, an equal sample for all three classes (negative, neutral and positive) was analyzed. Accordingly, 126 negative sentences, 126 neutral sentences and 125 positive sentences were picked randomly and evaluated on their sentiment by hand. The predicted sentiment of the sentences was hidden during this procedure to not influence the evaluation and thus eliminate a potential bias for the rater. For the negative class, 115 out of 126 (90.2%) of sentences were classified correctly. For the neutral class, also 115 out of 126 (90.2%) of sentences were classified correctly. For the positive class, 108 out of 125 (86.4%) of sentences were classified correctly. This results in an average classification accuracy of 0.896 over all three classes. In that regard, a correct classification means that the model's output was the same as the rating given by the human rater. Given the sampling, it can be stated that with a confidence level of 0.95 it is expected that the model classifies $89.6\% \pm 5\%$ ($84.6\% - 94.6\%$) of all sentences correctly. This

performance is very slightly worse than for the financial phrase dataset used for training, but still useful for the outlaid tasks.

5.3.2.2 Statistical Analysis

Knowing that the sentiment classifications are accurate to a useful degree, the overall distribution of classifications can be analyzed to get an overview. As mentioned previously, a total of 20'557 sentences were classified. The results in table 12 show that the sentiment for all firms was generally positive over the past 5 years, at least based on sentence classification count. For both small and mid cap companies, roughly 50 to 60 percent of all reported sentences belong to the positive category, while the neutral sentences are almost twice as much represented as negative ones. This distribution would also reflect the distribution of the financial phrase databank, which had a similar distribution in terms of frequency of positive, neutral and negative sentiment on a sentence level.

Table 12 Aggregated sentiment per company over 5 years

Cap	Ticker	Negative	Neutral	Positive	Total
Mid Cap	OERL	447	786	1'537	2'770
	SOON	254	479	1'061	1'794
	STMN	112	356	619	1'087
	TECN	76	358	869	1'303
	VACN	154	259	670	1'083
	Total Mid Cap	1'043	2'238	4'756	8'037
Small Cap	SUN	265	441	753	1'459
	ARBN	133	297	457	887
	BCHN	140	382	486	1'008
	BEAN	86	228	491	805
	BELL	147	319	573	1'039
	BUCN	625	353	1'429	2'407
	CALN	93	290	589	972
	EMMN	245	381	963	1'589
	FTON	187	319	564	1'070
	ZEHN	241	393	650	1'284
	Total Small Cap	2'162	3'403	6'955	12'520
Grand Total Count		3'205	5'641	11'711	20'557

The sentiment over time seems to fluctuate as well, which would be expected given the pandemic, which clearly affected the year 2020 and still is affecting 2021. Hence this should also be reflected in both, news and earnings reports. Accordingly, this expectation is represented in table 13, which lists a percentage-based view of sentence count for all three classifications over the years. The years 2016 and 2021 both do not have the same amount of data as the years in between. Reason being is that news articles published before 2017 become harder to find and also the year 2021 is not yet over. A clear increase in negative sentiment for both, news and earnings reports can be seen in the years 2020 and 2021. The table also shows that news tend to have a higher amount of neutral sentences in their text structure as compared to earnings reports. This might be due to the fact that earnings reports are less descriptive and relate more to facts (e.g. ‘net income increased by 15% yoy’), as the company can describe these results with more color in the actual annual report. As news articles do not need to be short and focused on earnings publication days, although that would be useful, they might be more descriptive in that field. Thus, it would be reasonable for news to contain more sentences with neutral sentiment.

Table 13 Sentiment over time (own creation)

Type	Years	Negative Sentiment	Neutral Sentiment	Positive Sentiment
News	2016	26%	39%	35%
	2017	15%	29%	56%
	2018	13%	33%	54%
	2019	16%	35%	49%
	2020	21%	32%	46%
	2021	17%	36%	47%
	Average over the years	17%	33%	51%
	Earnings Reports	2016	16%	27%
	2017	13%	21%	66%
	2018	8%	22%	69%
	2019	13%	24%	62%
	2020	22%	24%	54%
	2021	18%	25%	57%
	Average over the years	15%	24%	62%

The evaluated sentences were then aggregated back into a combined format, so that one sentiment evaluation for each of the gathered news articles and earnings reports could be created. This aggregation was based on majority voting, meaning that the amount of negative, neutral and positive sentences in any given text document was counted, whereas the majority class would determine the overall sentiment of the document. The total amount of 670 gathered news articles and earnings reports was then cut to 662, as 8 articles were published on Saturdays and Sundays, which are non-trading days. Given that no trading took place at the time of publication, the impact of the text on the stock at the time of publication cannot be measured. The 662 texts contain an aggregated amount of 53 negative texts, 187 neutral texts and 422 positive text. The distribution of the sentiment for news articles and the annual and quarterly earnings reports is given below in table 14.

Table 14 Overview of sentiment distribution

Text	Negative Sentiment	Neutral Sentiment	Positive Sentiment	Total
Annual Earnings report	3	2	84	89
Quarterly Earnings report	6	9	102	117
News	44	176	236	456
Grand Total	53	187	422	662

The distribution of the aggregated sentiment classification again shows that negative texts are rather underrepresented, especially for earnings reports. Out of a total of 206 earnings reports, only 9 contained an aggregated negative sentiment. This means that out of 15 companies, only 6 reported an overall negative sentiment in either a quarterly or annual earnings report. These companies were ARBN, BEAN, BUCN, FTON, SUN and ZEHN. While it would not be expected that a company actively describes its own performance as negative, even though that is the case, the pandemic has led to earnings decreases for many firms in the sample. Examples are OC Oerlikon and Sulzer, which recorded a decrease in EBIT of 55% and 38% respectively from 2019 to 2020. The fact that both of these firms have no aggregated negative sentiment in the corresponding earnings report for that time period is surprising. A closer analysis shows that the earnings reports for these firms contain a lot of “fluff” in the sense that while financial performance decreased, a lot of positivity is introduced into the text by describing strategies and hopes for better performance in the future. However, while there only are three annual earnings reports with an

aggregated negative sentiment, the stock return at the publication day of these earnings reports was negative for all three companies, ranging from -1 to -3 percent. Accordingly, these potential connections can be evaluated further. On the other hand, that was not the case for the quarterly earnings reports. In terms of statistical analysis, the correlations between every of the 15 companies and the stock performance of the days surrounding the publication of a news article or earnings report were analyzed. For that, the publication day is written as T+0, where the previous day is noted as T-1, and the day after T+0 is written as T+1. Accordingly, stock performance in percent was analyzed for the range of T-1, T+0, T+1, T+2 and T+3. Additionally, regression models were created to identify potential direct, linear influence of sentiment on stock performance. The independent variable in that case is the sentiment, which was changed from its original notation {'negative' : 0, 'neutral' : 1, 'positive' : 2} to {'negative' : -1, 'neutral' : 0, 'positive' : 1}. Given that sentiment is categorical variable it was used in regression using two dummy variables as follows:

Table 15 Dummy notation (own creation)

Sentiment	Variable	Dummy_1 (negative)	Dummy_2 (positive)
Negative		-1	1
Neutral		0	0
Positive		1	0

Accordingly, the dependent variable is the stock performance on the respectively selected day. This resulted in the 5 following regression models that were tested:

$$\begin{aligned}
 y_{t-1} &= \alpha + \beta_1 Dummy_{1,t+0} + \beta_2 Dummy_{2,t+0} + \varepsilon_i \\
 y_{t+0} &= \alpha + \beta_1 Dummy_{1,t+0} + \beta_2 Dummy_{2,t+0} + \varepsilon_i \\
 y_{t+1} &= \alpha + \beta_1 Dummy_{1,t+0} + \beta_2 Dummy_{2,t+0} + \varepsilon_i \\
 y_{t+2} &= \alpha + \beta_1 Dummy_{1,t+0} + \beta_2 Dummy_{2,t+0} + \varepsilon_i \\
 y_{t+3} &= \alpha + \beta_1 Dummy_{1,t+0} + \beta_2 Dummy_{2,t+0} + \varepsilon_i
 \end{aligned}
 \tag{15}$$

Where y is the performance of the stock price at the given date, α the intercept, β the slope and $Dummy$ the two dummy variables to indicate a negative, neutral or positive sentiment and ε_i the error term. The four regressions for T+0 to T+3 analyze if there is a linear connection between

either negative, neutral or positive sentiment with stock performance on the day of publication (possible as news articles and earnings reports were published before market opening) or the following three days. The regression for T-1 however takes into account that stock performance could be influenced by the assumption of positive or negative reporting on the following day. Hence an example would be that investors would already buy a given stock today, which would increase its price, if they expect a very positive earnings report to be published tomorrow. The results of the correlation analysis of the sentiment for mid caps and stock performance based on adjusted closing price development is displayed in table 16. As already mentioned, a total of 8 news articles had to be removed from the original dataset of 670 news articles and earnings reports, as they were published on a non-trading day.

Table 16 Correlation of sentiment and stock performance for mid caps (own creation)

Company	Cap	Type	Correlation				
			T+0	T-1	T+1	T+2	T+3
OERL	Mid Cap	News	0.116	0.285	0.215	0.306	0.280
		Earnings Report	-0.085	0.429	-0.010	0.180	0.057
		Overall	0.046	0.289	0.176	0.254	0.208
SOON	Mid Cap	News	0.130	-0.028	-0.228	-0.319	-0.246
		Earnings Report*	Nan	Nan	Nan	Nan	Nan
		Overall	-0.039	-0.081	-0.196	-0.146	-0.079
STMN	Mid Cap	News	0.168	-0.290	-0.264	-0.182	-0.280
		Earnings Report*	Nan	Nan	Nan	Nan	Nan
		Overall	0.151	-0.143	-0.189	-0.157	-0.270
TECN	Mid Cap	News	0.410	-0.172	-0.137	0.092	0.102
		Earnings Report	-0.033	-0.268	-0.085	-0.095	-0.019
		Overall	0.220	-0.210	-0.095	0.056	0.072
VAT	Mid Cap	News	-0.138	-0.130	-0.149	-0.162	-0.337
		Earnings Report*	Nan	Nan	Nan	Nan	Nan
		Overall	-0.019	-0.106	-0.145	-0.021	-0.171

Correlations above 0.3 or below -0.3 were highlighted in bold. Nan values could not be calculated due to the fact that the evaluated sentiment for these earnings reports was always positive, which is an interesting finding in itself. The results of the correlation analysis of the sentiment for small caps and stock performance based on adjusted closing price development is displayed in table 17, where again correlation above 0.3 or below -0.3 were highlighted in bold. Again, Nan

indicates correlations that could not be calculated since the evaluated sentiment for the earnings reports were exclusively positive. For both, small and mid caps, larger correlations were found. Especially for the small cap companies Belimo (BEAN), Bell (BELL) and FeinTool (FTON) which display correlations up to 0.702. Larger correlations can be found for all timeframes (T-1 to T+3), however in most cases the connection exists between T+0 with 8 large connections, T+2 with 7 large connections and +3 with also 8 large connections.

Table 17 Correlation of sentiment and stock performance for small caps (own creation)

Company	Cap	Type	Correlation				
			T+0	T-1	T+1	T+2	T+3
ARBN	Small Cap	News	0.117	-0.168	-0.113	-0.158	-0.139
		Earnings Report	-0.714	-0.027	0.275	0.104	-0.091
		Overall	-0.154	-0.116	0.029	-0.060	-0.123
BCHN	Small Cap	News	-0.082	-0.212	0.359	-0.013	0.072
		Earnings Report*	Nan	Nan	Nan	Nan	Nan
		Overall	-0.090	-0.216	0.342	-0.093	0.013
BEAN	Small Cap	News	0.127	-0.344	0.098	0.554	0.678
		Earnings Report	-0.024	0.050	0.125	0.238	-0.065
		Overall	0.036	-0.255	0.070	0.442	0.377
BELL	Small Cap	News	0.379	0.540	0.230	-0.213	0.114
		Earnings Report	0.240	-0.273	-0.038	0.625	0.702
		Overall	0.270	0.272	0.112	0.072	0.338
BUCN	Small Cap	News	0.371	-0.066	0.169	0.110	0.248
		Earnings Report	0.002	-0.220	0.116	-0.250	-0.101
		Overall	0.268	-0.119	0.126	0.017	0.164
CALN	Small Cap	News	-0.111	-0.370	-0.130	-0.232	-0.102
		Earnings Report	-0.370	-0.130	-0.232	-0.102	-0.095
		Overall	-0.011	-0.243	-0.068	-0.091	-0.016
EMMN	Small Cap	News	0.234	-0.247	-0.155	0.282	0.247
		Earnings Report*	Nan	Nan	Nan	Nan	Nan
		Overall	0.181	-0.241	-0.164	0.219	0.210
FTON	Small Cap	News	0.336	0.148	0.121	-0.370	-0.441
		Earnings Report	0.040	0.228	0.126	-0.123	-0.359
		Overall	0.258	0.156	0.113	-0.324	-0.417
SUN	Small Cap	News	0.112	0.155	-0.149	0.051	-0.059
		Earnings Report	-0.039	0.516	0.477	-0.224	-0.090
		Overall	0.079	0.197	-0.070	-0.013	-0.060
ZEHN	Small Cap	News	0.553	-0.360	-0.118	-0.112	0.038
		Earnings Report	0.036	0.023	0.097	0.101	0.134
		Overall	0.344	-0.227	-0.061	-0.067	0.067

Strong correlations can be found mostly for small caps, but for all three sections (news, earnings reports and overall). While correlation does not imply causation in any form, it is interesting to find such large values, given the many economical inputs that would theoretically flow into the performance of any given stock. However, note that above correlations are Pearson correlation coefficients and not correlation ratios, also called Eta, which are normally used for correlation calculations of categorical and discrete numbers. The correlation coefficient was chosen due to its ease in implementation and as correlation itself provides no basis for a trading model the scores were only calculated out of interest and for informative purposes.

A closer, more relevant review is provided with the below regression results. For each of the 15 companies, five regressions as per equation 15 above were created, leading to a total of 45 conducted regressions. The analysis has found statistically significant results (significance level with $\alpha=0.05$) for either single or all variables for Burckhardt Compression Holding AG (BCHN), Bell Food (BELL), Bucher Industries (BUCN), Straumann (STMN) and Zehnder (ZEHN). The corresponding results are displayed in the appendix in table 22 to 31. This shows that there are linear connections between the sentiment of news articles and earnings reports of a company and the stock price surrounding the publication date of said textual data, at least for a handful of companies. Accordingly, one of the regressions for BELL in table 18 below is examined more closely to analyze its results.

Table 18 Regression results for BELL at T+2

SUMMARY		Company: BELL				
OUTPUT	T+2					
<hr/>		<hr/>				
<i>Regression Statistics</i>		<i>ANOVA</i>				
Multiple R	0.50808047					
R Square	0.25814577	<i>df</i>	<i>SS</i>	<i>MS</i>	<i>F</i>	<i>Significance F</i>
Adjusted R Square	0.21028420	Regression	2	0.0034	0.0017	5.3936
Standard Error	0.01788093	Residual	31	0.0099	0.0003	0.0098
Observations	34	Total	33	0.0134		

	<i>Coefficients</i>	<i>Standard Error</i>	<i>t Stat</i>	<i>P-value</i>	<i>Lower 95%</i>	<i>Upper 95%</i>	<i>Lower 95.0%</i>	<i>Upper 95.0%</i>
Intercept	-0.015	0.005	-2.845	0.008	-0.025	-0.004	-0.025	-0.004
dummy sentiment 1	0.039	0.014	2.846	0.008	0.011	0.067	0.011	0.067
dummy sentiment 2	0.016	0.007	2.415	0.022	0.002	0.029	0.002	0.029

A multiple linear regression was calculated for BELL to predict stock returns two days into the future (T+2) based on the sentiment of news articles and earnings reports of the same stock published on day T+0. A significant regression equation was found ($F(2, 31) = 5.3936, p < 0.0098$), with an R^2 of 0.25. The stock return is equal to $-0.015 + 0.039$ (Dummy Sentiment 1) + 0.016 (Dummy Sentiment 2), where Dummy Sentiment 1 and Dummy Sentiment 2 are coded as [1, 0] for negative sentiment, [0, 0] for neutral sentiment and [0, 1] for positive sentiment. The stock return at T+2 increases 0.039 if there is negative sentiment present and also increases 0.016 if there is positive sentiment present. Both, negative and positive sentiment represented via Dummy Sentiment 1 and Dummy Sentiment 2 were significant predictors of stock return in T+2.

Accordingly, the regression equation would predict a positive stock return of $-0.015 + 0.039 * 1 + 0.016 * 0 = 0.024$ or 2.4% within two days after publication of an article or earnings report with negative sentiment. In the same way, the model would predict a positive stock return of 0.001 or 0.1% within two days after publication of an article or earnings report. The publication of a neutral text would result in negative stock return of -0.015 or - 1.5 %. In all cases where both Dummy variable 1 and Dummy variable 2 as well as the overall regression was statistically significant with $\alpha=0.05$, the coefficient for Dummy variable 1 (negative sentiment) was larger than for Dummy variable 2 (positive sentiment), therefore indicating that negative news would always lead to larger, positive stock returns than positive media coverage or company own reportings. One possible explanation for this would be that if negative news are published, the stock price would temporarily fall and then investors would pick the stock back up on the second day after the initial publication. Given the complexity of stock price determination and the influence of many different financial variables into price making it comes surprising that such linear effects exist. However, this complexity is confirmed by the regressions as, while being significant, the R^2 value for the discussed BELL regression was 0.25, while the overall highest identi-

fied R^2 value was at 0.37. Furthermore, only 5 out of the total 15 companies showed significant variables or overall significant regression results. While the observed linear connections are surprising to see, one cannot interpret too much prediction strength into sentiment alone on stock return development. However, the sentiment definitely seems to have an effect on stock returns, although that effect might be opposite of what one would expect, i.e. negative news leading to higher stock returns.

5.3.2.3 *Non-linear Analysis*

While there seems to be a linear connection between sentiment and the return of some stocks in the Swiss small- and mid cap segment, it is also of interest to find out if there are non-linear effects between the two variables. This can be achieved by combining both, historical stock prices and sentiment indicators into a ML model with the goal of predicting the future stock price. Such a model was successfully created by Ching-Ru and Hsien-Tsung in March 2021 (2021). They have found that when sentimental information from three stocks listed on the Taiwan Stock Exchange is combined with its historical stock price is combined as input for a LSTM, as described in section 2.1.3, the forecast accuracy of the future stock price increases. The study measures an average improvement of 12.05% as compared to a model without sentiment, which only used historical stock information.

To test if such an effect is also visible for the Swiss small- and mid cap stocks based on the sentiment of the gathered news articles and earnings reports, a LSTM model in the style of Ching-Ru and Hsien-Tsung (2021) is created. However, the task is slightly altered given the difference in datasets between the cited paper and the gathered news articles and earnings reports. For each of the Swiss stocks the adjusted closing prices at the publication date of a text (T+0) and the past 20 trading days (T-1 to T-20) are gathered. Together with the sentiment of the text at T+0, the model shall then predict the stock price at the publication date T+0. Given that all gathered news articles and earnings reports in the dataset were published before the opening of the Swiss stock exchange (9 AM), they can be evaluated before market opening and thus their effect can be analyzed directly on the same day of publication.

In terms of data pre-processing the stock prices were standardized, by centering and scaling them to have a mean of 0 and standard deviation of 1. Standardization of a dataset is a common re-

quirement for many machine learning estimators, as they might behave badly if the individual features do not more or less look like standard normally distributed data (Fan, 2021).

$$\text{Standardized value} = \frac{x - \mu}{\sigma} \quad (16)$$

Where x is the variable, μ the mean of the training samples and σ the standard deviation of the training samples. Training and testing data were standardized separately, to prevent any data spill from the training dataset onto the testing data.

The first feature for the LSTM model is the collected sequence of 20 of the adjusted closing prices for every stock, representing the development of the stock price in the past 20 days (T-1 to T-20). The second feature for the model is the sentiment of the text (news article or earnings report) on the day of publication T+0, which is first again transformed into dummy variable format. Accordingly, negative sentiment is depicted as [1, 0], neutral sentiment as [0, 0] and positive sentiment as [0, 1]. To match with the number of data from the stock returns (20), the sentiment dummies are padded, meaning that the array in which they are stored is filled with zeroes. This is required for the matrix calculation concerning both features, the stock returns and sentiment. These two features are then formatted into a 3D matrix with the shape (662, 20, 2). The number 662 represent the 662 news articles and earnings reports. The number 20 represents 20 timesteps, meaning that the model has a lookback of 20 days in the past. Lastly, the number 2 represents the number of features, which are the direction of past stock returns and the sentiment. This input data was then used to train two LSTM models, using the layout as shown in figure 26, where one time both features were used, and the other model only used the stock return development as an input. Therefore, the input shape of the second model with only one feature was (662, 20, 1).

```
# Initializing the Neural Network based on LSTM
model = Sequential()

# Adding 1st LSTM layer
model.add(LSTM(units=512, return_sequences=True, input_shape=(20,1)))
# Adding 2nd LSTM layer
model.add(LSTM(units=1024, return_sequences=True))
# Adding Dropout
model.add(Dropout(0.2))

# Output layer
model.add(Dense(units=1, activation = 'linear'))

# Compiling the Neural Network
model.compile(optimizer = Adam(learning_rate=0.01), loss='mean_squared_error', metrics = ['accuracy'])
```

Figure 26 LSTM configuration (own creation)

The models were created with a supervised task in mind, meaning that the output of the model should predict the stock price at time $T+0$. The model performance is measured using the mean squared error as a loss function, where a lower error would indicate better model performance. However, while training the models it showed that both do not converge, independent of model complexity and applied regularization methods. This non-converge means that the results of Ching-Ru and Hsien-Tsung (2021) cannot be confirmed in the same manner for the gathered data. Furthermore, assuming correct specification of the model parameters, the non-converge of the LSTM models could be due to the following reasons. Firstly, it might be that the models do not find relationships between both features and the stock price at time $T+0$. While this might be the case for sentiment, it is rather unlikely that there would be no identifiable relationship between a stock price of today and the stock price of the same company of the past 20 days. However, given that both models do not converge, no clear testimonial about this can be made. Another reason for non-convergence might be that the padding for the sentiment data is confusing the network as it transforms a matrix of e.g. $[0, 1]$ into $[0, 1, 0, 0, 0, 0, \dots, 0]$ therefore adding many zeroes as inputs. Another explanation might be that the lookback for the stock prices was not large enough, meaning that more than 20 days are required to find a potential pattern. However, a longer lookback would also again lead to larger padding for the sentiment. Therefore, such a methodology simply seems not fitting for the given dataset. Lastly, the overall reason for non-convergence might also be due to the training set being too small for the given task. Given the non-convergence, a closer analysis of results makes no sense and the LSTM description was therefore held brief.

Keeping these findings in mind, another model architecture was created to try and find connections between sentiment and stock data. Using feed-forward neural networks, again two models were created. The task for these models was in the style of supervised binary classification. Accordingly, instead of predicting the stock price as a number, the output of the neural networks shall predict the direction (positive or negative) of the stock return on day $T+0$. As inputs, the past 20 stock returns together with the two dummy variables for sentiment are taken as a single feature each. In terms of data pre-processing, the daily stock return for all 15 companies was calculated and transformed into $[-1, 0, 1]$ depending on whether the return was negative (below zero), zero, or positive (above zero). As every stock development was taken as a single feature, and

not a sequence, the dummy variables for sentiment did not require any padding for any matrix calculation. All input data was again standardized as previously described. The output was accordingly also transformed into a dummy variable, where 1 is used to indicate positive stock return and 0 represents a negative stock return. If the stock return on any given day was exactly zero (no movement), it is also represented as 0. With this, the dataset contains 286 cases where the daily return at time $T+0$ was negative (class 0) and 376 cases where the daily return at time $T+0$ was positive. The output variable is therefore relatively balanced with a 43 to 57 ratio. The neural networks for both models were initialized using the following architecture as shown in figure 27.

```
# Initializing the Neural Network
model = Sequential([
    tf.keras.layers.Dense(22, input_shape = (X_train.shape[1],)), #only indicate nr. of columns

    tf.keras.layers.Dense(1024, activation=tf.nn.relu),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(128, activation=tf.nn.relu),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(1, activation=tf.nn.sigmoid)
])

# Compiling the Neural Network
model.compile(optimizer = Adam(learning_rate=0.01), loss='binary_crossentropy', metrics = ['accuracy'])
```

Figure 27 Neural Network architecture (own creation)

Each model was trained using the following hyperparameters:

- Epochs: 200
- Batch size: 256
- Hidden layers: 2 layers with 1024 and 128 neurons each
- Activation functions: ReLU for hidden layers and Sigmoid for output layer
- Loss function: Binary cross entropy
- Regularization: Dropout with value = 0.5 and L2 regularization
- Optimizer: Adam

The final hyperparameters were the result of the iterative testing of many different combinations. Using these specifications for both models, one with stock and sentiment data and the other exclusively with stock data, can be compared in their performance. Given that these models follow a classification task, just as in the DistilBERT model, they can be evaluated using the same

measures (F1-Scores and MCC) as initially described in section 2.1.4. The baseline performance in terms of accuracy for both models is a prediction accuracy of 59.6 percent. That is the amount of positive daily returns (class 1) in the testing dataset. Both trained models beat this naïve benchmark with accuracy scores of 0.71 for the model with sentiment and 0.63 for the model without sentiment. This shows that there is a gap of eight percentage points in performance between the two models. Knowing that both models produce useful results, they can be analyzed more closely. In figure 28, the range of predictions for both models is displayed, where the model with sentiment data is shown on the left. The range of predictions has two large accumulations of prediction values that are either very close to zero or to one, indicating that the model is quite sure about most of the classifications, whereas a few stragglers are in between with values from 0.2 to 0.8. The model without sentiment however displays higher uncertainty in that it has one larger accumulation close to zero, whereas most predicted values are around 0.6. Given that the naïve cutoff for this classification lies at 0.5 (values below are categorized as class 0 which are negative stock developments, and values above are categorized as class 1), this is rather close to this decision border. While building the model without sentiment data, L2 regularization had to be applied on top of strong dropout layers to prevent the model from only predicting the dominant class (class 1). This behavior could be stopped with the strong regularization but as a result the negative class (class 0) was only predicted 16 times. The model with sentiment did not profit from the L2 regularization on top of the dropout layers and thus the L2 weight penalty was not applied.

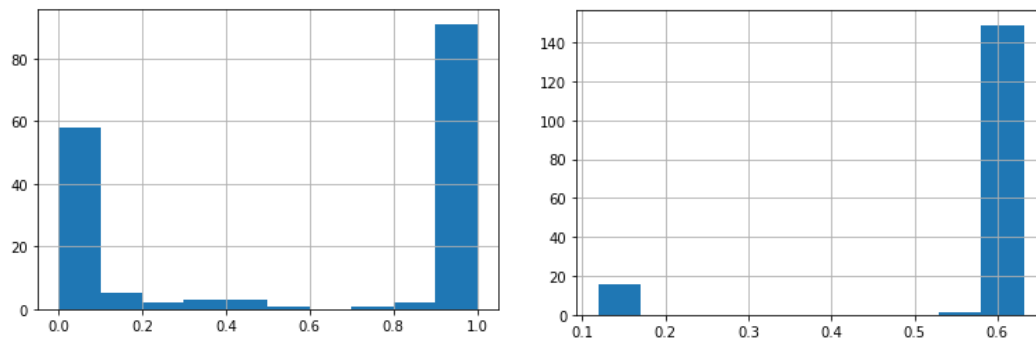


Figure 28 Prediction range of the model with sentiment (left) and without (right) (own creation)

Using a confusion matrix as shown in figure 29, one can see the large amount of misclassification for the negative class of the model without sentiment. The misclassifications for the model with sentiment on the left-side of the graph are more balanced.

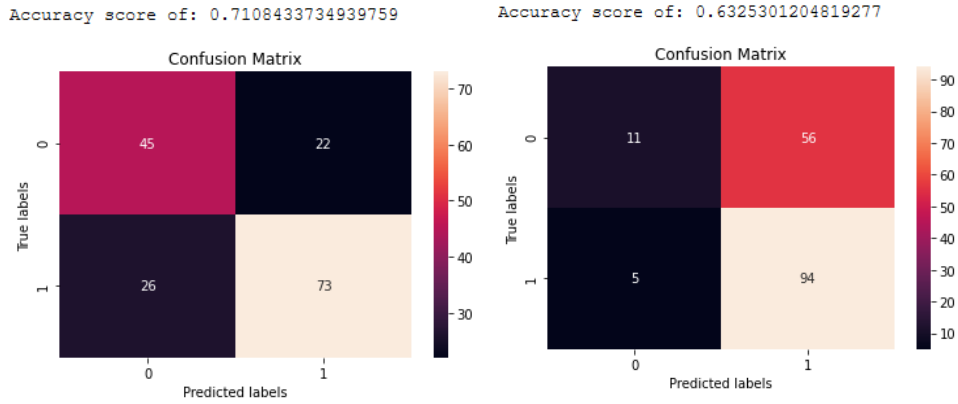


Figure 29 Confusion matrix for the model with sentiment (left) and without (right) (own creation)

Accordingly, with its accuracy of 0.71, the model with sentiment could analyze the sentiment of a news article or earnings report on any given day before market opening and predict the direction of the stock price development with an accuracy of around 70 percent. However, both performances cannot really be considered satisfactory from a prediction standpoint, given the F1-scores in figure 30. Both models perform worse in predicting the negative class, especially the model without any sentiment data. This is displayed with an F1-score of 0.65 for the negative class in the model with sentiment, while the model without sentiment has an F1-score of 0.27 for the negative class. Overall F1-scores for the models are 0.71 and 0.56, respectively. Given the weaker performance in the negative class, which is also slightly underrepresented in the training dataset, the MCC for both models is at 0.40 and 0.18 respectively.

	precision	recall	f1-score	support		precision	recall	f1-score	support
0	0.63	0.67	0.65	67	0	0.69	0.16	0.27	67
1	0.77	0.74	0.75	99	1	0.63	0.95	0.76	99
accuracy			0.71	166	accuracy			0.63	166
macro avg	0.70	0.70	0.70	166	macro avg	0.66	0.56	0.51	166
weighted avg	0.71	0.71	0.71	166	weighted avg	0.65	0.63	0.56	166

Figure 30 Precision, Recall and F1-score for the model with sentiment (left) and without (right) (own creation)

While these performances are not useful from an industry standpoint for actual stock return prediction, this is not bothersome, given that the reason for creating the model was to find out if the

addition of sentiment to a stock return prediction model based on historical stocks can actually provide any benefit. To summarize it shows that the neural network is able to make use of the two dummy variables containing the sentiment in terms of pattern recognition for the provided data and thus there seems to be a certain relationship between the sentiment of the gathered news articles and earnings reports and the stock returns of the respective Swiss small and mid cap companies. This finding lies in a similar realm as the results of Ching-Ru and Hsien-Tsung (2021), which stated that adding sentiment to the model increased prediction performance by roughly 12 percent.

5.4 Discussion

This section provides an overview and summary of the findings of the applied DistilBERT model and complexity formulas on the gathered news articles and earnings reports.

The strength of the applied complexity formulas, the Flesch score and the Wiener Sachtextformel, is that they are quick to implement and are also used in practice, for example as an instruction manual standard of the US military. However, by applying them onto the gathered news articles and earnings reports it showed that they both are very sensitive in their rating regarding short sentences with longer words. The results showed that both, news articles and earnings reports, in the dataset are considered to be difficult to understand and require the reading level of a university student, at least per the grading scale of the Flesch score. Both scores create similar results, as their scores have a correlation coefficient of -0.867 . It was found that texts for small cap companies are easier to understand than for mid caps. Furthermore, an analysis of text complexity levels of earnings reports of the past 5 years has shown that complexity increased in a clear trend, where the highest values were recorded in 2020 and 2021. This effect did not show for news articles. Reason being could lie in the global pandemic and its negative influence on the financial world. Lastly, the proposed connection between sentences with negative sentiment having a higher complexity could not be confirmed, given that the difference in complexity scores between the sentiment ratings negative, neutral and positive is negligible. All findings are observed critically as the differences in complexity scores were small so that the observed effects only exist in the range of a few percent regarding complexity. As the scores are all rather close together, it was determined in a self-test by the author that complexity differences of a few percentage points are not noticeable for human readers. Given this lack of volatility in the complexity scores

and the difficulty of noticing smaller differences in complexity no further analysis in terms of the effect of complexity on stock returns or volume was conducted.

The sentiment classifications created using the trained DistilBERT model have been tested in a sample size testing and it showed that the classification accuracy with a range of 84.6 % – 94.6 % correct predictions was slightly worse than in the training scenario. However, this out of sample performance is still very adequate for the given task. It was found that only 15 percent of all categorized sentences were negative, while 56 percent were positive. This means that the reporting in both earnings reports and news articles over the past 5 years was on average very positive for the selected 15 Swiss companies. However, the amount of negative sentiment slightly increased in the years 2020 and 2021, which again could be due to the global pandemic. In combination with complexity it can therefore be stated that earnings reports became harder to understand (higher complexity) and also contained more negative sentiment. An impactful finding was that of a total of 206 earnings reports for the past 5 years, only 9 reports, made up of 3 annual and 6 quarterly earnings reports, could be classified as containing a majority of negative sentiment on an aggregated level. While the stock return at the time of publication of the three earnings reports was then also negative, this was not the case for the 6 quarterly earnings reports. This finding is essentially the same for the total 15 neutral earnings reports on an aggregated level. Given that the earnings and situations of the respective companies were not always overwhelmingly positive, especially in the years 2020 and 2021, this shows that earnings reports are written with a strong positive bias. An understandable explanation for this finding would be that companies simply do not word their reporting in a negative way on an aggregated level, although the financial results were not positive. The evaluated news articles show more fluctuation in their sentiment. Based on this, news articles would seem to be a better information source for sentiment, however also here the potential bias of the journalist that writes the article must be considered. The correlations between stock returns on publication days of news articles and earnings reports $T+0$, as well as for the previous and next three days was analyzed ($T-1$, $T+1$, $T+2$ and $T+3$). For some companies, no strong correlations were found, while a handful of firms, such as the Bell Food Group displayed very large positive correlations of up to 0.7, while other firms such as Calida only displayed negative correlations. In that sense, a negative correlation would indicate that a news article or earnings report with positive sentiment would lead to negative stock returns

for that company. Additionally, a total of 45 regressions were conducted, where exclusively two dummy variables, representing the sentiment, were used to evaluate their predictive ability on stock returns, either at publication date or T-1, T+1, T+2 and T+3. Significant results with a confidence level of 0.95 were found, either for just one variable, or even for all variables and on regression level. However, only a handful of the total 15 companies displayed such statistically significant results. Nonetheless, this shows that there is a linear connection between sentiment ratings and stock returns for a few companies that are listed on the Swiss stock exchange.

To also analyze the potential non-linear connection between sentiment and stock returns, two neural network models were created. The task of the models was to predict the direction of the development of the stock return at the publication day (T+0) of an earnings report or news article. One model was provided with sentiment data at publication date and the direction of the stock returns of the past 20 days (T-1 to T-20), while the other model was only provided with the stock data and no sentiment. The binary classification models showed that the model that was provided with sentiment data was able to predict the direction of the stock return development with an accuracy of 71 percent. This was 8 percentage points better than the performance of the second model which only used stock data. This shows that there are also non-linear connections between sentiment and stock returns. Hence it is concluded that sentiment is a beneficial variable in a stock return prediction model. Similar results were already displayed for American and Asian stocks in literature, however so far, no research for Swiss companies, especially not for small and mid caps, existed. This means that for the task of stock return prediction in the Swiss market, one could try and combine further potentially influential variables together with sentiment and previous stock returns to create a better forecasting model. However, this would require more data, which exactly describes the main problem for evaluating Swiss small- and mid caps – the lack of data. During the creation of this paper it was identified that reporting standards of Swiss companies in terms of their earnings reports is sub-par. While the reportings of American companies are all aggregated and publicly displayed with a strict layout on the SEC's EDGAR, this does not exist for Switzerland. Therefore, Swiss companies can, and are, based on the authors opinion, less disciplined regarding the quality of their earnings reports. Examples of bad quality reporting include reports in PDF format that contain exclusively pictures of text instead of actual text, missing reports for given years, hidden behind dead links, or simply publishing

annual earnings reports without any detailed indication regarding the publication date. This fact in combination with the finding that earnings reports very seldomly display negative sentiment on an aggregated level makes them conflicted data sources in terms of sentiment analysis. As already mentioned, the evaluated news articles are therefore seen as a better data source. Accordingly, it would be of interest if news articles alone, or earnings reports alone, also would lead to the same improvement of stock prediction as displayed by the neural networks. This would however again require more data, which in turn highlights the small amount of information available for Swiss small- and mid caps.

6 Conclusion

This master thesis presented the ML process for NLP problems, analyzed multiple algorithms and also implemented own models for both text complexity and text sentiment. These were then applied to self-gathered news articles and earnings reports of Swiss small- and mid caps that are listed on the SIX Swiss stock exchange. Chapter 2 has presented this ML process for natural language processing problems based on the steps of data collection, data pre-processing, its classification and lastly on how to analyze and interpret the results. Multiple ML algorithms for NLP were evaluated based on their theoretical descriptions and performance on state-of-the-art evaluation dataset, ranging from Naïve Bayes to the Transformer. This leads to the answer of the first research question of this thesis “*What are transformer models, and do they outperform other NLP techniques?*”. Transformer models were introduced as a modern evolution of neural networks that make use of encoders, decoders and the attention mechanism. As an improvement to the Transformer, Bidirectional Encoder Representations from Transformers (BERT) were introduced and evaluated. BERT improves the Transformer model by introducing a masked language model (MLM) pre-training objective. Unlike a left-to-right language model, this MLM process enables the representation to fuse the left and the right context of a sentence. Given this bidirectionality and BERT’s state-of-the-art performance on multiple NLP tasks in literature, BERT, as a form of Transformer, is identified as being able to clearly outperform other, older NLP algorithms such as Naïve Bayes. This was confirmed by the implementation of an own DistilBERT language processing model to analyze the sentiment of news articles and earnings reports of Swiss small and mid cap companies between the 1st of June 2016 and 31st of May 2021. The model achieved a classification accuracy of roughly 0.9 on out of sample data.

Using this model and a Python implementation of the Flesch Score adaptation for German and the Wiener Sachtextformel, the text complexity and sentiment of 670 news articles and earnings reports of 5 Mid cap and 10 Small caps were analyzed. This leads to the second question of this paper “*Does text complexity have any effect on text sentiment?*”. While findings in literature have shown that firms with lower earnings tend to file annual reports that are more difficult to read, no literature regarding their effect on sentiment was found. The empirical testing using the above-mentioned earnings reports and news articles led to the result that these text documents are generally hard to understand and require the reading level of university students. However, for

each document type the calculated complexity scores were very close together with a low volatility. In terms of the effect of complexity on sentiment, the average complexity of negative, neutral and positive sentences was all within 3 percentage points of each other, indicating that no clear connection between sentiment and complexity exists. Therefore, the assumption that a negative earnings report being more complex to understand could not be confirmed. To answer the last question of this thesis “ *Are there any connections between sentiment, complexity, and news and earnings reports for Swiss small- and mid caps and their corresponding stock price movements?*”. Given the above described findings for complexity, only the topic of sentiment will be further expanded on. On an aggregated report level, where the sentiment of all evaluated sentences was added back together to make up their original news article or earnings report, it showed that of a total of 206 earnings reports, only 9 were negative. Given that the financial performance of the 15 companies in the sample was not always overwhelmingly positive, especially not in the pandemic affected year 2020 and 2021, this would indicate a positivity bias in the writing of the analyzed earning reports. As an explanation for this bias, it is suggested that Swiss companies are relatively free in the language of their own reporting as accordingly it would be understandable that negative performance would be hidden behind a certain amount of “fluff” in the text. News articles seem to be a more balanced source of sentiment indicators. With regard to the connection between sentiment and stock returns of the analyzed companies a total of 45 regressions were made, which analyzed the effect of sentiment on stock returns a day before publication of the document, on the publication day, and up to three days after publication (T-1, T+0, T+1, T+2 and T+3). For a minority of the 15 stocks, significant results with a confidence level of 0.95 were found, together with strong correlations between sentiment and stock returns. While this displays a linear connection between sentiment and stock returns for Swiss small- and mid caps, the effect was only visible for a few companies, while other showed no such connections. Furthermore, two feed-forward neural network models were trained on the task of predicting the stock return at the day of publication, one time using the sentiment of the article or reports published that day and the stock returns of the past 20 day, and the other time using only the stock returns. The prediction of the stock return on publication day is possible as the gathered news articles and earnings reports were published before market opening and therefore can have a direct impact on the whole trading day. It showed that the model with sentiment data was able to predict the direction of the stock return (positive or negative) with an accuracy of 0.71. This beat

the performance of the model with only stock data by 8 percentage points, as it was only able to predict 63 percent of all stock returns correctly. Overall, the created sentiment scores therefore seem to have an impact on stock price development of the analyzed 15 Swiss small and mid cap companies. It can therefore be used as a variable in Swiss stock prediction models as it seems to be one of potentially many variables that influence price making. As an outlook for such a model it is suggested that more Swiss companies can be analyzed to consolidate the findings of this paper. However, this leads to one of the key problems of Swiss small- and mid caps: lack of public information. Accordingly, further testing for the impact of sentiment on the Swiss stock market for companies with small and middle-sized market capital would require large efforts in data gathering. Should the non-linear effect found via the created neural networks also show for more of the total 216 listed Swiss companies, a more complex stock return prediction model could be created, based on the findings of this paper. Additionally, as done in other papers, sentiment could be split into more specific sub-groups instead of the negative, neutral and positive classes.

7 References

- Abdullah, Q. A., Manjula, B., & Lakshman Naik, R. (2019). A Study of Sentiment Analysis: Concepts, Techniques, and Challenges. *Proceedings of International Conference on Computational Intelligence and Data Engineering* (pp. 147 - 164). Singapore: Springer.
- Agarwal, A., Xie, B., Vosha, I., Rambow, O., & Passonneau, R. (2011). Sentiment Analysis of Twitter Data. *Proceedings of the Workshop on Language in Social Media*, 30 - 38.
- Aggarwal, C. C. (2018). Neural Networks and Deep Learning: A Textbook. In C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook* (pp. 14-15). Yorktown Heights, NY: Springer Verlag.
- Aggarwal, C. C. (2018, January 01). *Neural Networks and Deep Learning: Presentation Slides*. Retrieved from charuaggarwal.net: <http://www.charuaggarwal.net/AllSlides.pdf>
- Alam, S., & Yao, N. (2018). The impact of preprocessing steps on the accuracy of machine learning algorithms in sentiment analysis. *Computational and Mathematical Organization Theory*, 319 – 335.
- Albawi, S., Mohammed, T., & Al-Zawi, S. (2017). Understanding of a Convolutional Neural Network. *ICET 2017* (pp. 2 - 7). Antalya: IEEE.
- Allington, R., McCuiston, K., & Billen, M. (2015). What Research Says About Text Complexity and Learning to Read. *The Reading Teacher*, 491 - 501.
- Amstad, T. (1978). *Wie verständlich sind unsere Zeitungen?* Zurich: University Zurich.
- Angiani, G., Ferrari, L., Fontanini, T., Fornacciari, P., Iotti, E., Magliani, F., & Manciardì, S. (2016). A Comparison between Preprocessing Techniques for Sentiment Analysis in Twitter. *Proceedings of the 2nd International Workshop on Knowledge Discovery on the WEB (KDWEB 2016)* (pp. 8 - 10). Cagliari: CEUR-WS.
- Araci, D. (2019). *FinBERT: Financial Sentiment Analysis with Pre-Trained Language Models*. Amsterdam: Prosus AI.

Analysis of Sentiment and Complexity of News and Earnings Reports of Swiss SMEs and their Relevance for Stock Returns

- Auria, L., & Moro, R. (2008). *Support Vector Machines (SVM) as a technique for solvency analysis*. Berlin: Deutsches Institut für Wirtschaftsforschung (DIW).
- AWP. (2021, 06 13). *About AWP*. Retrieved from AWP Finanznachrichten: <https://www.awp.ch/index.php/de/about-awp.html>
- Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *3rd International Conference on Learning Representations, ICLR 2015*, (pp. 1 - 15). San Diego.
- Bamberger, R., & Vanecek, E. (1984). *Lesen-Verstehen-Lernen-Schreiben: Die Schwierigkeitsstufen von Texten in deutscher Sprache*. Wien: Volk Sauerlaender.
- Bollen, J., Mao, H., & Zeng, X.-J. (2010). Twitter mood predicts the stock market. *Journal of Computational Science*, 1 - 8.
- Burnard, L. (2021, 05 17). *British National Corpus*. Retrieved from <http://www.natcorp.ox.ac.uk/corpus/index.xml>:
<http://www.natcorp.ox.ac.uk/corpus/index.xml>
- CFA Institute. (2019). *Investment Professional of the future*. CFA Institute.
- Cheng, J., Dong, L., & Lapata, M. (2016). Long Short-Term Memory-Networks for Machine Reading. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (pp. 551–561). Texas: Association for Computational Linguistics.
- Chi, S., Qiue, X., Xu, Y., & Huang, X. (2019). How to Fine-Tune BERT for Text Classification. *Chinese Computational Linguistics*, 194 - 206.
- Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, 1 - 13.
- Ching-Ru, K., & Hsien-Tsung, C. (2021). LSTM-based sentiment analysis for stock price forecast. *Peer Journal for Computer Science*, 1-23.
- Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical

Analysis of Sentiment and Complexity of News and Earnings Reports of Swiss SMEs and their Relevance for Stock Returns

Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1724 - 1734). Doha: Association for Computational Linguistics.

Chollet, F. (2018). Deep Learning with Python. In F. Chollet, *Deep Learning with Python* (pp. 49 - 51). Shelter Island, NY: Manning.

Chowdhury, G. (2005). Natural Language Processing. *Annual Review of Information Science and Technology*, 51.

D'Andrea, A., Ferri, F., Grifoni, P., & Guzzo, T. (2015). Approaches, Tools and Applications for Sentiment Analysis Implementation. *International Journal of computer Applications*, 26 - 33.

DeepL. (2021, 05 03). *DeepL Translator*. Retrieved from DeepL: <https://www.deepl.com/en/publisher/>

deepset.ai. (2019, 06 14). *German-bert*. Retrieved from Open Sourcing German BERT: <https://deepset.ai/german-bert>

Deshpande, A., & Fleisig, E. (2020). *Sentiment Analysis for Reinforcement Learning*. Princeton: University of Princeton.

Devlin, J., & Chang, M.-W. (2018, 11 2). *Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing* . Retrieved from Google AI Blog: <https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html>

Devlin, J., Chang, M.-W., Lee, K., & 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 (Long and Short Papers)* (pp. 4171–4186). Minneapolis: Google AI Language.

DuBay, W. H. (2007). *Smart Language: Readers, Readability, and the Grading of Text*. Costa Mesa: Impact Information.

- Fama, E. (1970). Efficient Capital Markets: A Review of Theory and Empirical Work. *Journal of Finance*, 383-417.
- Fan, T. (2021, 06 17). *Standard Scaler*. Retrieved from scikit-learn: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn-preprocessing-standardscaler>
- Flesch, R. (1948). A new readability yardstick. *Journal of Applied Psychology*, 221 -233.
- Flesch, R. (2016, 07 12). *How to write plain English*. Retrieved from University of Canterbury: https://web.archive.org/web/20160712094308/http://www.mang.canterbury.ac.nz/writing_guide/writing/flesch.shtml
- Gao, T., Chai, Y., & Liu, Y. (2017). Applying long short term memory neural networks for predicting stock closing price. *2nd IEEE international conference on recent trends in electronics, information & communication technology (RTEICT)* (pp. 575-578). IEEE.
- Git Hiew, Z., Huang, X., Mou, H., Li, D., Wu, Q., & Xu, Y. (2019). BERT-based Financial Sentiment Index and LSTM-based Stock Return Predictability. *Conference on Neural Information Processing Systems* (pp. 1 - 10). Vancouver: NeurIPS.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. In I. Goodfellow, Y. Bengio, & A. Courville, *Deep Learning* (pp. 204-221). Cambridge: MIT Press.
- Göpferich, S., Jakobsen, A. L., & Mees, I. M. (2009). *Behind the Mind: Methods, Models and Results in Translation Process Research*. Copenhagen: Samfundslitteratur Press.
- Haddi, E., Liu, X., & Shi, Y. (2013). The Role of Text Pre-processing in Sentiment Anlaysis. *Procedia Computer Science*, 26 - 32.
- Hand, D., & Yu, K. (2001). Idiot's Bayes - Not so Stupid after all? *International Statistical Review* , 385-398.
- Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The Elements of Statistical Learning*. New York: Springer.

- Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning. In T. Hastie, R. Tibshirani, & J. Friedman, *The Elements of Statistical Learning* (pp. 485- 490). Stanford, CA: Springer-Verlag.
- He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770-778). IEEE.
- Hutto, C., & Gilbert, E. (2014). *VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text*. Association for the Advancement of Artificial Intelligence.
- Kanis, J., & Skorkovska, L. (2010). Comparison of Different Lemmatization Approaches through the Means of Information Retrieval Performance. *International Conference on Text, Speech and Dialogue* (pp. 93 - 100). Berlin: Springer.
- Kaplan, A., & Haenlein, M. (2019). Siri, Siri, in my hand: Who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence. *Business Horizons*, 15- 25.
- Kibriya, A. M., Frank, E., Pfahringer, B., & Holmes, G. (2004). Multinomial Naive Bayes for Text Categorization Revisited. In G. Webb, & X. Yu, *AI 2004: Advances in Artificial Intelligence* (pp. 488-499). Berlin: Springer.
- Kim, S.-M., & Hovy, E. (2004). Determining the Sentiment of Opinions. *Proceedings of the 20th International Conference on Computational Linguistics* (pp. 1 - 7). Geneva: Association for Computational Linguistics.
- Klare, G. (1963). *The Measurement of Readability*. Iowa: Iowa State University Press.
- Koza, J., Bennett III, F., Andre, D., & Keane, M. (1996). Automated Design of Both the Topology and Sizing of Analog Electrical Circuits using Genetic Programming. *Artificial Intelligence in Design*, 151 - 170.
- Krotov, V., & Tennyson, M. (2018). Scraping Financial Data from the Web Using the R Language. *Journal of Emerging Technologies in Accounting*.

- Krotov, V., & Tennyson, M. (2018). Scraping Financial Data from the Web Using the R Language. *Journal of Emerging Technologies in Accounting*, 61-73.
- Krouska, A., Troussas, C., & Virvou, M. (2016). The effect of preprocessing techniques on Twitter Sentiment Analysis. *2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA)* (pp. 1 - 5). Chalkidiki: IISA.
- Kulgemeyer, C., & Starauschek, E. (2013). Analyse der Verständlichkeit naturwissenschaftlicher Fachtexte. *Methoden in der naturwissenschaftsdidaktischen Forschung*, 241 - 253.
- Langley, P. (2011). The changing science of Machine Learning. *Machine Learning*, 275-279.
- Lei Ba, J., Kiros, J. R., & Hinton, G. (2016). Layer Normalization.
- Li, F. (2008). Annual report readability, current earnings, and earnings persistence. *Journal of Accounting and Economics*, 221-247.
- Liddy, E. (2001). Natural Language Processing. In E. Liddy, *Encyclopedia of Library and Information Science* (p. 2). New York: Marcel Decker Inc.
- Loughran, T., & McDonald, B. (2011). When is a Liability not a Liability? Textual Analysis, Dictionaries, and 10-Ks. *Journal of Finance*, 35 - 65.
- Malo, P., Sinha, A., Korhonen, P., J., W., & Takala, P. (2014). Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, 1-65.
- Maslej-Kresnakova, V., Sarnovsky, M., Butka, P., & Machova, K. (2020). Comparison of Deep Learning Models and Various Text Pre-Processing Techniques for the Toxic Comments Classification. *Applied Science*, 1 - 26.
- Mat Zin, H., Mustapha, N., Murad, M., & Sharef, N. (2017). The effects of pre-processing strategies in sentiment analysis of online movie reviews. *The 2nd International Conference on Applied Science and Technology 2017* (pp. 1 - 8). AIP Publishin.
- Melo, F. (2013). Area under the ROC Curve. In W. Dubitzky, O. Wolkenhauer, K. Cho, & H. Yokota, *Encyclopedia of Systems Biology* (p. 5). New York: Springer.

Analysis of Sentiment and Complexity of News and Earnings Reports of Swiss SMEs and their Relevance for Stock Returns

- Mishev, K., Gjorgjevikj, A., Vodenska, I., Chitkushev, L., & Trajanov, D. (2020). Evaluation of Sentiment Analysis in Finance: From Lexicons to Transformers. *IEEE Access*, 1 - 2.
- Mishev, K., Gjorgjevikj, A., Vodenska, I., Chitkushev, L., & Trajanov, D. (2020). Evaluation of Sentiment Analysis in Finance: From Lexicons to Transformers. *IEEE Access*, 131662 - 131682.
- Mitchell, R. (2018). *Web Scraping with Python: Collecting More Data from the Modern Web*. Sebastopol: O'Reilly Media.
- Mitchell, T. (1997). *Machine Learning*. New York: McGraw-Hill Education.
- Nicholson, C. (2021, 05 20). *attention mechanism memory network*. Retrieved from pathmind: <https://wiki.pathmind.com/attention-mechanism-memory-network>
- Oppenlaender, J. (2021, 05 15). *German stopwords*. Retrieved from github: https://github.com/solariz/german_stopwords
- Osborne, M. (2002). Using Maximum Entropy for Sentence Extractions. *Proceedings of the Workshop on Automatic Summarization* (pp. 1 - 8). Philadelphia: Association for Computational Linguistics.
- Oxford Lexico. (2021, 05 14). *lexiko*. Retrieved from lexiko: https://www.lexico.com/definition/sentiment_analysis
- Pennebaker, J., Chung, C., Ireland, M., Gonazles, A., & Booth, R. (2007). *The Development and Psychometric Properties of LIWC2007*. Austin: LIWC.net. Retrieved from The development and psychometric properties of LIWC2007: liwc.net
- Pinto, J., Henry, E., Robinson, T., & Stowe, J. (2010). *Equity Asset Valuation*. John Wiley & Sons.
- Rai, A., & Borah, S. (2021). Study of Various Methods for Tokenization. In M. J., M. S., & R. A., *Applications of Internet of Things. Lecture Notes in Networks and Systems* (pp. 193 - 200). Singapore: Springer.

Analysis of Sentiment and Complexity of News and Earnings Reports of Swiss SMEs and their Relevance for Stock Returns

- Refenes, A. N., Zapranis, A., & Franics, G. (1994). Stock performance modeling using neural networks: a comparative study with regression models. *Neural Networks*, 357 - 388.
- Roebuck, K. (2011). *Sentiment Analysis: High-Impact Strategies - What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors*. Lightning Source.
- Rojas, R. (1996). Neural Networks: A Systematic Introduction. In R. Rojas, *Neural Networks: A Systematic Introduction* (pp. 151-173). Berlin: Springer-Verlag.
- Russell, S., & Norvig, P. (2003). Artificial Intelligence: A modern approach. In S. Russell, & P. Norvig, *Artificial Intelligence: A modern approach* (pp. 650-651). New Jersey: Prentice Hall.
- Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *Conference on Neural Information Processing Systems* (pp. 1 - 5). Vancouver: NeurIPS.
- Schuster, M., Johnson, M., & Thorat, N. (2016, 11 22). *Zero-Shot Translation with Google's Multilingual Neural Machine Translation System*. Retrieved from Google AI Blog: <https://ai.googleblog.com/2016/11/zero-shot-translation-with-googles.html>
- Shmilovici, A. (2005). Support Vector Machines. In A. Shmilovici, *Data Mining and Knowledge Discovery Handbook* (pp. 257 - 258). New York: Springer.
- Si, L., & Callan, J. (2001). A statistical model for scientific readability. *Proceedings of the tenth international conference on information and knowledge management* (pp. 574 - 576). CIKM.
- Similarweb. (2021, 06 04). *Website statistics*. Retrieved from Cash.ch website statistics: <https://www.similarweb.com/website/cash.ch/#overview>
- SIX Swiss Exchange. (2021, 06 04). *SMI MID*. Retrieved from The SMI MID: <https://www.six-group.com/de/products-services/the-swiss-stock-exchange/market-data/indices/equity-indices/smi-mid.html>

- SIX Swiss Exchange. (2021, 06 04). *SPI*. Retrieved from Swiss Performance Index: <https://www.six-group.com/de/products-services/the-swiss-stock-exchange/market-data/indices/equity-indices/spi.html>
- Snoek, J., Larochelle, H., & Adams, R. (2012). Practical Bayesian Optimization of Machine Learning Algorithms. *Proceedings of the 25th International Conference on Neural Information Processing Systems* (pp. 2951–2959). NIPS.
- Sutton, R. S., & Barto, A. G. (2017). Reinforcement Learning: An Introduction. In R. S. Sutton, & A. G. Barto, *Reinforcement Learning: An Introduction* (pp. 1-5). Cambridge, MA: The MIT Press.
- Taherdoost, H. (2018). Determining Sample Size; How to Calculate Survey Sample Size. *International Journal of Economics and Management Systems*, 2 - 4.
- Taspinar, A. (2016, 03 28). *Regression, Logistic Regression and Maximum Entropy*. Retrieved from ataspinar: <https://ataspinar.com/2016/03/28/regression-logistic-regression-and-maximum-entropy/>
- Taylor, W. (1953). Cloze procedure: A new tool for measuring readability. *Journalism Bulletin*, 415 - 433.
- Tetlock, P. (2007). Giving Content to Investor Sentiment: The Role of Media in the Stock Market. *The Journal of Finance*, 1139-1168.
- Tetlock, P., Saar-Tsechansky, M., & Macskassy, S. (2008). More than words: Quantifying language to measure firm's fundamentals. *Journal of Finance*, 1437-1467.
- Theissen, P. (2017, 09 04). *Wiener Sachtextformeln*. Retrieved from github.com: <https://github.com/pablotheissen>
- Tredinnick, L. (2017). Artificial intelligence and professional roles. *Business Information Review*, 37 - 41.
- U.S. Securities and Exchange Commission (SEC). (2021, 05 16). <https://sec-api.io/>. Retrieved from <https://sec-api.io/>: <https://sec-api.io/>

- Uhr, P., Zenkert, J., & Fathi, M. (2014). Sentiment Analysis in Financial Markets: A Framework to utilize the Human Ability of Word Association for analyzing Stock Market News Reports. *Intelligent Systems IEEE*, 1- 6.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., . . . Polosukhin, I. (2017). Attention Is All You Need. *31st Conference on Neural Information Processing System* (pp. 1 - 15). Long Beach: Google Brain.
- West, J., Venutra, D., & Warnick, S. (2007). *A Theoretical Foundation for Inductive Transfer*. Brigham: BYU College of Physical and Mathematical Sciences.
- Wisniewski, T., & Yekini, L. (2014). Predicting Stock Market Returns Based on the Content of Annual Report Narrative: A New Anomaly. *Munich Personal RePEc Archive (MPRA)*, 1 - 37.
- Yang, Z., Yang, D., Dyer, D., He, X., & Smola, A. H. (2016). Hierarchical attention networks for document classification. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1480- 1489.
- yfinance. (2021, 06 04). *pypi.org*. Retrieved from yfinance 0.1.59: <https://pypi.org/project/yfinance/>
- You, H., & Zhang, X.-j. (2009). Financial reporting complexity and investor underreaction to 10-K information. *Review of Accounting studies*, 559 -586.
- Zeroual, I., & Lakhouaja, A. (2018). Data science in light of natural language processing: An overview. *Procedia Computer Science* 127, 82 - 92.
- Zhu, Y., Krios, R., Zemel, R., Salakhutdinov, R., Torralba, A., & Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *Proceedings of the IEEE international conference on computer vision* (pp. 19 - 27). IEEE.

8 Appendix

8.1 Complexity Tables

Table 19 Mid cap complexity overview

		Flesch score					Wiener Score							
		2016	2017	2018	2019	2020	2021	2016	2017	2018	2019	2020	2021	
mid_cap	Mid cap Average	37.455518	37.266009	38.301341	40.569227	40.45019	41.762732	12.294627	12.501986	12.120305	11.807324	11.832431	11.959008	
	OERL	Company Average	40.533889	39.372667	40.456789	41.782316	41.376651	43.426547	12.253716	12.228783	12.300994	11.880095	11.562411	11.661342
		annual earnings report	36.446667	22.06	39.05	42.899375	63.519231	39.027857	12.347658	15.95359	10.856481	11.887817	7.2802259	13.559426
		news		40.87427	39.505522	39.441603	41.504845	47.633095		11.870573	12.468265	12.296677	11.529294	10.81878
		quarterly earnings report	42.5775	38.63661	44.730786	49.212338	33.483016	26.7925	12.206744	12.539424	12.113412	10.488914	13.122271	13.976072
	SOON	Company Average	36.463195	33.367069	32.508881	36.432282	39.843147	44.182251	12.320035	13.141225	12.906041	12.130853	11.844752	11.952141
		annual earnings report	43.099444	35.554898	38.473719	37.006774	34.221149		11.610608	12.416122	12.037102	11.684993	12.042963	
		news	32.990685	33.064196	25.9	38.752686	41.351668	44.182251	12.708779	13.355361	14.299063	11.912531	11.737824	11.952141
		quarterly earnings report	33.299455	32.996481	33.152925	26.576176	33.396984		12.640719	12.581514	12.381958	13.450002	12.501966	
	STMN	Company Average	37.213333	31.184339	44.111147	41.815029	42.516721	42.362554	11.983331	13.283355	10.861025	11.462509	11.84436	11.711488
		annual earnings report			46.603846	40.985		38.992857			10.698581	11.801483		11.527416
		news	37.213333	31.184339	44.763676	44.248729	43.834643	42.84394	11.983331	13.283355	11.204214	11.141122	11.694979	11.737784
		quarterly earnings report			39.008333	28.042857	29.3375				9.6507135	13.051858	13.338179	
	TECN	Company Average	38.923342	31.133758	30.231407	35.701969	33.099576	37.358825	11.578676	13.640566	12.853212	12.077361	12.599077	12.578828
		annual earnings report	38.671856	37.763056	35.734382	40.814155	35.69914	34.36551	11.615787	11.767547	12.163409	11.297823	12.156611	12.379326
		news		29.807898	25.574527	33.774968	33.766384	38.356597		14.01517	13.867721	12.325452	12.390662	12.645328
		quarterly earnings report	39.174828		39.122121	34.220844	26.499167		11.541565		10.661841	12.348963	14.292031	
	VACN	Company Average	28.746102	44.328324	40.100033	45.170488	42.726868	40.545091	14.395631	11.198686	11.612331	11.512104	11.660157	12.280039
		annual earnings report		45.425	45.790714	41.851111	59.59	38.33125		11.301638	11.256044	11.542617	8.5501613	11.452383
		news		46.954468	41.761667	47.575226	41.080107	44.011401		10.577707	11.991408	11.348071	12.032551	11.567411
quarterly earnings report		28.746102	37.214625	33.931424	40.818333	41.705728	32.36	14.395631	12.699657	11.032321	11.906927	11.539383	15.245581	

Table 20 Small cap complexity overview 1 out of 2

		Flesch score						Wiener Score						
		2016	2017	2018	2019	2020	2021	2016	2017	2018	2019	2020	2021	
Small cap	SUN	Company Average	40.342031	37.13672	38.572799	42.115747	46.103882	38.401085	12.08083	11.986872	11.776322	11.514309	10.904209	12.068827
		annual earnings report		40.525106	39.669905	41.131579		35.0752		11.621353	11.155075	12.31195		12.584709
		news			37.819155	42.31451	46.103882	38.770628			11.790618	11.435215	10.904209	12.011507
		quarterly earnings report	40.342031	33.748333	41.792464	41.812781			12.08083	12.352391	12.015468	11.431863		
	ARBN	Company Average	41.070942	42.451996	37.869022		44.209656	38.07102	12.695976	11.650929	12.979861		11.825733	12.444674
		annual earnings report	42.601	42.348941			59.328	49.571212	10.630048	11.987701			8.256036	10.291566
		news	37.099583	44.099256	37.869022		41.24887	35.195972	14.576679	11.288062	12.979861		12.582454	12.982951
		quarterly earnings report	47.4836	32.774545			43.895238		11.000499	13.154588			11.611824	
	BCHN	Company Average	31.831469	41.163489	33.717106	42.639076	33.792615	36.559221	13.216371	11.631931	12.940205	11.668228	12.756806	12.941341
		annual earnings report	23.262174	27.105952	27.555902	27.862097	27.065758		14.721735	14.485508	14.511498	14.426652	14.307381	
		news	46.974194	41.771326	35.269837	49.16911	34.396206	36.559221	10.599298	11.386872	12.574912	10.502445	12.596326	12.941341
		quarterly earnings report	25.258039	50.358333	33.667388	24.765882	35.690741		14.32808	10.738825	12.830084	14.738718	12.490072	
	BEAN	Company Average	39.087766	44.178799	41.30341	35.024851	37.744538	38.211182	12.080163	12.07812	12.381678	13.062345	12.348171	12.2895
		annual earnings report	31.919333	40.43	40.793824	36.017895	44.88125	39.830769	13.077133	12.802451	12.190755	13.409301	10.519593	12.267082
		news		46.868524	41.803851	32.664163	36.780722	37.67132		11.607708	12.458844	13.186776	12.649807	12.296973
		quarterly earnings report	42.671982	37.168696	40.05688	41.113871	33.499273		11.581677	13.235437	12.245643	12.342097	13.271841	
	BELL	Company Average	36.857861	37.242546	44.736186	42.438389	45.70273	45.668058	12.516829	11.768893	10.789851	11.250828	10.332904	10.990627
		annual earnings report	39.346415	44.171429	49.777619	39.249444	60.9575		11.77044	11.09023	9.8851295	11.568679	7.5839784	
		news		34.495172	42.92513	44.142813	47.447691	45.668058		12.443043	11.035076	11.787026	10.193725	10.990627
		quarterly earnings report	35.613583	37.899167	49.09256	42.328438	25.213077		12.890024	11.097001	10.274006	10.555704	13.499367	

Table 21 Small cap complexity overview 2 out of 2

		Flesch score						Wiener Score						
		2016	2017	2018	2019	2020	2021	2016	2017	2018	2019	2020	2021	
Small cap	BUCN	Company Average		39.053672	36.21519	40.580594	38.74811	39.310115		12.323141	12.802356	11.902675	12.383915	12.119417
		annual earnings report		36.880278	31.21522	39.648369	36.924086	30.877445		12.552103	13.097453	12.126395	13.056387	12.989576
		news		41.898816	39.037447	42.725189	39.627693	44.325456		12.032453	12.392278	11.713465	12.104909	11.448543
		quarterly earnings report		30.294324	32.022485	37.627753	37.086231	31.09875		13.215777	13.699167	12.068878	12.770238	13.733465
	CALN	Company Average	37.832946	40.154649	39.198541	40.867536	42.932562	45.297804	12.419576	11.939274	11.84808	11.593467	11.736045	11.408076
		annual earnings report			46.966857	39.035294	48.738261	46.166667			10.666285	12.083364	11.489123	10.812509
		news	45.691667	44.674355	38.682556	39.081351	42.168167	44.428942	10.870775	11.155055	12.001155	11.875162	11.856447	12.003642
		quarterly earnings report	33.903586	28.855382	35.042124	48.058333	47.064		13.193977	13.899821	11.958348	10.258484	10.417729	
	EMMN	Company Average		45.501655	44.47588	40.812516	39.547649	39.827287		10.797778	10.80958	11.633751	11.910255	11.558068
		annual earnings report		46.669444	48.971286	48.692857	44.091875	35.425		10.849307	10.438879	11.134853	11.033882	11.288637
		news		45.698965	43.924014	42.813238	38.605274	39.806193		10.888946	10.632297	11.404311	12.053611	11.780943
		quarterly earnings report		43.15	39.9	33.871262	42.458594	44.292857		10.199244	12.96925	12.227362	11.559982	11.158875
	FTON	Company Average	45.005634	40.647896	40.611042	38.850232	35.188025	41.08675	11.115938	11.817182	11.813562	12.059833	12.878338	11.376518
		annual earnings report	50.658824	49.113636	43.835	41.42381	38.060714	41.354286	10.142017	10.371322	12.777666	11.102932	12.298955	11.679774
		news		40.557203	43.884937	40.370316	34.613487	40.997571		11.829192	10.967255	11.701485	12.994215	11.275432
		quarterly earnings report	39.352444	38.279451	32.988598	36.472287			12.089859	12.239089	13.184806	12.737147		
	ZEHN	Company Average	39.564439	42.9806	41.544974	41.324688	40.205964	37.505599	11.565396	11.634223	11.92133	11.804053	11.698156	12.530762
		annual earnings report	46.238987	42.363769	37.909832	39.652244	39.560556	41.408333	10.183681	11.746055	12.796565	12.374828	11.965318	11.538908
		news	34.915551	42.737661	42.310186	42.394904	38.559695	35.471181	11.940681	11.672915	11.683243	11.408515	12.035209	13.607058
		quarterly earnings report	39.763629	46.643649	38.8675	40.38871	48.259583	33.768969	12.111253	11.023632	13.265995	12.244657	9.9142573	12.361875

8.2 Regression Results

Table 22 BUCHN regression T+1

SUMMARY OUT-
PUT T+1

Company BUCHN

<i>Regression Statistics</i>	
Multiple R	0.347166272
R Square	0.12052442
Adjusted R Square	0.071664666
Standard Error	0.02503109
Observations	39

<i>ANOVA</i>					
	<i>df</i>	<i>SS</i>	<i>MS</i>	<i>F</i>	<i>Significance F</i>
Regression	2	0.003091102	0.001545551	2.466742243	0.099089612
Residual	36	0.022555997	0.000626555		
Total	38	0.025647099			

	<i>Coefficients</i>	<i>Standard Error</i>	<i>t Stat</i>	<i>P-value</i>	<i>Lower 95%</i>	<i>Upper 95%</i>	<i>Lower 95.0%</i>	<i>Upper 95.0%</i>
Intercept	0.002439662	0.006070931	-0.40186	0.690163276	-0.014752081	0.009872757	0.014752081	0.009872757
dummy sentiment 1	0.006436027	0.025756779	-0.24988	0.804102883	-0.058673197	0.045801143	0.058673197	0.045801143
dummy sentiment 2	0.017387495	0.008166528	2.129117	0.040154659	0.000825009	0.033949982	0.000825009	0.033949982

Table 23 BELL regression T+0

SUMMARY OUT-PUT T+0

Company: BELL

<i>Regression Statistics</i>	
Multiple R	0.378112363
R Square	0.142968959
Adjusted R Square	0.087676634
Standard Error	0.048326025
Observations	34

<i>ANOVA</i>					
	<i>df</i>	<i>SS</i>	<i>MS</i>	<i>F</i>	<i>Significance F</i>
Regression	2	0.012077278	0.006038639	2.585692649	0.091505365
Residual	31	0.072397546	0.002335405		
Total	33	0.084474823			

	<i>Coefficients</i>	<i>Standard Error</i>	<i>t Stat</i>	<i>P-value</i>	<i>Lower 95%</i>	<i>Upper 95%</i>	<i>Lower 95.0%</i>	<i>Upper 95.0%</i>
Intercept	0.027054156	0.013950522	1.939293452	0.061613488	-0.055506432	0.001398121	0.055506432	0.001398121
dummy sentiment 1	0.032262352	0.036909611	0.874090805	0.388790316	-0.043015297	0.10754	0.043015297	0.10754
dummy sentiment 2	0.039962236	0.017646169	2.264640878	0.03067467	0.003972637	0.075951836	0.003972637	0.075951836

Table 24 BELL regression T-1

SUMMARY OUT-PUT T-1

Company: BELL

<i>Regression Statistics</i>	
Multiple R	0.50590642
R Square	0.255941306
Adjusted R Square	0.207937519
Standard Error	0.016050423
Observations	34

<i>ANOVA</i>					
	<i>df</i>	<i>SS</i>	<i>MS</i>	<i>F</i>	<i>Significance F</i>
Regression	2	0.002747058	0.001373529	5.331689929	0.010230846
Residual	31	0.007986098	0.000257616		
Total	33	0.010733157			

	<i>Coefficients</i>	<i>Standard Error</i>	<i>t Stat</i>	<i>P-value</i>	<i>Lower 95%</i>	<i>Upper 95%</i>	<i>Lower 95.0%</i>	<i>Upper 95.0%</i>
Intercept	0.005362393	0.004633358	1.157344939	0.255971516	-0.004087403	0.014812189	0.004087403	0.014812189
dummy sentiment 1	0.039324667	0.012258713	-3.2078952	0.003100541	-0.064326477	0.014322857	0.064326477	0.014322857
dummy sentiment 2	0.002202909	0.005860786	0.375872675	0.709571573	-0.014156061	0.009750242	0.014156061	0.009750242

Table 25 BELL regression T+2

SUMMARY OUT-
PUT T+2

Company: BELL

<i>Regression Statistics</i>	
Multiple R	0.508080477
R Square	0.258145771
Adjusted R Square	0.210284208
Standard Error	0.017880936
Observations	34

<i>ANOVA</i>					
	<i>df</i>	<i>SS</i>	<i>MS</i>	<i>F</i>	<i>Significance F</i>
Regression	2	0.003448964	0.001724482	5.393592566	0.009770976
Residual	31	0.009911564	0.000319728		
Total	33	0.013360528			

	<i>Coefficients</i>	<i>Standard Error</i>	<i>t Stat</i>	<i>P-value</i>	<i>Lower 95%</i>	<i>Upper 95%</i>	<i>Lower 95.0%</i>	<i>Upper 95.0%</i>
Intercept	0.014683397	0.005161782	2.844637499	0.007807773	-0.02521092	0.004155875	-0.02521092	0.004155875
dummy sentiment 1	0.038863143	0.01365679	2.845701085	0.007787211	0.011009936	0.066716351	0.011009936	0.066716351
dummy sentiment 2	0.015769735	0.006529195	2.415265026	0.021811307	0.002453355	0.029086116	0.002453355	0.029086116

Table 26 BELL regression T+3

SUMMARY OUT-PUT T+3

Company: BELL

<i>Regression Statistics</i>	
Multiple R	0.611569543
R Square	0.374017306
Adjusted R Square	0.333631326
Standard Error	0.026419811
Observations	34

<i>ANOVA</i>					
	<i>df</i>	<i>SS</i>	<i>MS</i>	<i>F</i>	<i>Significance F</i>
Regression	2	0.012928569	0.006464285	9.261067934	0.000702613
Residual	31	0.021638198	0.000698006		
Total	33	0.034566767			

	<i>Coefficients</i>	<i>Standard Error</i>	<i>t Stat</i>	<i>P-value</i>	<i>Lower 95%</i>	<i>Upper 95%</i>	<i>Lower 95.0%</i>	<i>Upper 95.0%</i>
Intercept	0.020098545	0.007626742	2.635272578	0.013009277	-0.035653389	0.004543701	0.035653389	0.004543701
dummy sentiment 1	0.049459616	0.020178464	2.451109117	0.020079616	0.008305368	0.090613864	0.008305368	0.090613864
dummy sentiment 2	0.039633704	0.009647151	4.108332614	0.000270078	0.019958211	0.059309198	0.019958211	0.059309198

Table 27 BUCN regression T-1

SUMMARY OUTPUT		T-1	Company	BUCN						
<i>Regression Statistics</i>		<i>ANOVA</i>								
Multiple R	0.305504847				<i>df</i>	<i>SS</i>	<i>MS</i>	<i>F</i>	<i>Significance F</i>	
R Square	0.093333212				Regression	2	0.001605723	0.000802861	2.933819316	0.061271663
Adjusted R Square	0.061520342				Residual	57	0.015598473	0.000273657		
Standard Error	0.016542594				Total	59	0.017204196			
Observations	60									
	<i>Coefficients</i>	<i>Standard Error</i>	<i>t Stat</i>	<i>P-value</i>	<i>Lower 95%</i>	<i>Upper 95%</i>	<i>Lower 95.0%</i>	<i>Upper 95.0%</i>		
Intercept	0.009779984	0.00584869	1.672166454	0.099972745	-0.021491784	0.001931817	0.021491784	0.001931817		
dummy sentiment 1	0.017989991	0.007433556	2.420105768	0.018726691	0.003104552	0.032875429	0.003104552	0.032875429		
dummy sentiment 2	0.010672925	0.006420595	1.662295482	0.101943895	-0.002184094	0.023529944	0.002184094	0.023529944		

Table 28 STMN regression T+0

SUMMARY OUTPUT		Company	STMN						
T+0									
<i>Regression Statistics</i>		<i>ANOVA</i>							
Multiple R	0.4213624			<i>df</i>	<i>SS</i>	<i>MS</i>	<i>F</i>	<i>Significance F</i>	
R Square	0.177546272	Regression		2	0.008547515	0.004273757	3.885729723	0.029648759	
Adjusted R Square	0.131854398	Residual		36	0.039594948	0.00109986			
Standard Error	0.033164132	Total		38	0.048142463				
Observations	39								
<hr/>									
	<i>Coefficients</i>	<i>Standard Error</i>	<i>t Stat</i>	<i>P-value</i>	<i>Lower 95%</i>	<i>Upper 95%</i>	<i>Lower 95.0%</i>	<i>Upper 95.0%</i>	
Intercept	0.010838922	0.008291033	1.307306579	0.199397048	-0.005976072	0.027653917	0.005976072	0.027653917	
dummy sentiment 1	0.095002831	0.034184805	2.779095289	0.008609109	-0.16433283	0.025672833	-0.16433283	-0.025672833	
dummy sentiment 2	-0.00323992	0.010896552	0.297334411	0.767920249	-0.025339152	0.018859312	0.025339152	0.018859312	

Table 29 STMN regression T+3

SUMMARY OUTPUT		T+3	Company	STMN						
<i>Regression Statistics</i>		<i>ANOVA</i>								
Multiple R	0.480337709									
R Square	0.230724315									
Adjusted R Square	0.187986777									
Standard Error	0.035310953									
Observations	39									
					<i>df</i>	<i>SS</i>	<i>MS</i>	<i>F</i>	<i>Significance F</i>	
					Regression	2	0.013462718	0.006731359	5.398633741	0.008901765
					Residual	36	0.044887083	0.001246863		
					Total	38	0.0583498			
	<i>Coefficients</i>	<i>Standard Error</i>	<i>t Stat</i>	<i>P-value</i>	<i>Lower 95%</i>	<i>Upper 95%</i>	<i>Lower 95.0%</i>	<i>Upper 95.0%</i>		
Intercept	0.019352591	0.008827738	2.192247943	0.034910158	0.001449108	0.037256074	0.001449108	0.037256074		
dummy sentiment 1	0.11396058	0.036397697	3.130983235	0.003449925	0.040142628	0.187778531	0.040142628	0.187778531		
dummy sentiment 2	-0.00466604	0.011601921	0.402178233	0.689930872	0.028195827	0.018863746	0.028195827	0.018863746		

Table 30 ZEHN regression T+0

SUMMARY OUT-
PUT

T+0

Company ZEHN

<i>Regression Statistics</i>	
Multiple R	0.373625767
R Square	0.139596214
Adjusted R Square	0.110429984
Standard Error	0.045935244
Observations	62

<i>ANOVA</i>					
	<i>df</i>	<i>SS</i>	<i>MS</i>	<i>F</i>	<i>Significance F</i>
Regression	2	0.020198327	0.010099164	4.786227557	0.011849997
Residual	59	0.124492754	0.002110047		
Total	61	0.144691081			

	<i>Coefficients</i>	<i>Standard Error</i>	<i>t Stat</i>	<i>P-value</i>	<i>Lower 95%</i>	<i>Upper 95%</i>	<i>Lower 95.0%</i>	<i>Upper 95.0%</i>
Intercept	0.00982929	0.009793427	1.003661912	0.319640259	-0.009767312	0.029425892	0.009767312	0.029425892
dummy sentiment 1	0.044229226	0.01896489	2.332163595	0.023124559	-0.082177883	0.006280569	0.082177883	-0.006280569
dummy sentiment 2	0.011939136	0.012722035	0.938461211	0.351833519	-0.013517597	0.037395869	0.013517597	0.037395869

Table 31 ZEHN regression T+2

SUMMARY OUT-
PUT

T+2

Company ZEHN

<i>Regression Statistics</i>	
Multiple R	0.329277436
R Square	0.10842363
Adjusted R Square	0.078200702
Standard Error	0.031295653
Observations	62

<i>ANOVA</i>					
	<i>df</i>	<i>SS</i>	<i>MS</i>	<i>F</i>	<i>Significance F</i>
Regression	2	0.007027251	0.003513625	3.58746282	0.033859241
Residual	59	0.057785657	0.000979418		
Total	61	0.064812908			

	<i>Coefficients</i>	<i>Standard Error</i>	<i>t Stat</i>	<i>P-value</i>	<i>Lower 95%</i>	<i>Upper 95%</i>	<i>Lower 95.0%</i>	<i>Upper 95.0%</i>
Intercept	0.021825304	0.006672256	3.271053338	0.00179214	0.008474151	0.035176457	0.008474151	0.035176457
dummy sentiment 1	-	-	-	-	-	-	-	-
dummy sentiment 1	0.027420268	0.012920768	2.122185673	0.038030458	-0.053274664	0.001565872	0.053274664	-0.001565872
dummy sentiment 2	-	-	-	-	-	-	-	-
dummy sentiment 2	0.020310828	0.008667514	2.343327864	0.02250259	-0.037654484	0.002967172	0.037654484	-0.002967172