Investigating the correlation between sentiment and stock-price fluctuations

By

Jørgensen, Oliver L. - Exam no.68876

Paget, Marc D. - Exam no. 68831

Tækker, Tobias L. - Exam no. 68973

&

Utzon, Bjørn A. - Exam no. 69234

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF HUM-TEK BACHELOR AT ROSKILDE UNIVERSITY

Bachelor of Hum-Tek

Roskilde University

June, 2022

SUPERVISOR

Jens Classen

ABSTRACT

This paper, seeks to examine the correlation between stock price and public sentiment expressed through social media. Through twitter scraping and pre-processing, sentiment can be extracted from text. The paper will be based on a heuristic approach to natural language processing. Furthermore, the paper will rely on the most common forms of sentiment analysis, using a rule-based and a machine-learning approach as a starting point and weigh these up against each other. Finally, we will continue with the best performing method, and weigh this up against real market data in a pursuit to find a correlation, should one exist. The paper found a sentiment-to-market accuracy 75%. And the accuracy score utilizing the rules-based approach of 72,72%.

Key Words:

Natural Language Processing, NLP, Sentiment Analysis, Tweepy, SpaCy, Web-Scraping, Stock-Prediction

Contents

1 Introduction

With technological advancements, social media has become a household item. And with good reason, as it is an excellent platform for sharing opinions with people out of one's vicinity. It means that every single user can share their opinion with other users globally in the blink of an eye. With more than 200 million daily Twitter users[7], This amounts to a massive abundance of data. It can be tiresome to sift through this information to find the opinions of a certain interest. How can this data be extracted for quantitative purposes? And could this information be reflective of the general public sentiment?

Investors typically base their decision on financial statements, such as prospective future earnings; however, recent studies show that emotions also influences investment decisions[5]. These emotions can be influenced by news or general public sentiment [16]. This brought forth the following hypothesis: is there a way to analyze public sentiment and correlate this to stock pricing. This paper seeks to investigate that possibility and try to find a correlation between stock price variations and change in public sentiment. In practice, we will collect Tweets from Twitter and develop an algorithm to determine sentiment. At last, we will examine the relationship between sentiment and stock-prices

The act of predicting the stock market has been studied by many researchers, while recent studies in behavioral economics have shown that emotions influences investment decisions[5][10]. Previous theories have speculated the possibility of stock movement as a random walk. However, researchers have observed a correlation between news article headlines and stock returns. Investing decisions, seem to be partly based on irrational behavior such as mood and public sentiment [16]. With the rise of Social media, we expect some of these platforms to have gained increasing importance in displaying public opinions. We believe this might support our hypothesis of using social media to determine public sentiment in combination with stock price fluctuation prediction.

This is not a new phenomenon, and it is the reason we believe this experiment of extracting public sentiment from social media

We have decided to use the public Twitter API to gather tweets and process them with Python to evaluate our hypothesis.

1.1 Motivation

This project started with a common interest in stock fluctuations. We found interest in how prominent Twitter users, like Elon Musk, could affect stock prices and cryptocurrencies by expressing themselves on Twitter. A bit of research gave us further motivation. According to a study by Xu Qianwen; Chang Victor and Hsu Ching-Hsien[12], using sentiment analysis to show sentimental impact on a stock's price is possible. They studied economic articles and specifically looked for rumors and speculations around a handful of different companies. After gathering sentiment, they compared their findings to how the stock market reacted to those speculations. Interestingly, the group found that after five days, the stock price on a given company went up, with positive rumors, and down for negative rumors, though the down stock price would bounce back after a while[12].

This inspired us to attempt to create our own sentiment analyzer by scouring the internet using Python. Relying on our own experiences of different social media, we chose to analyze Twitter. Later we will discuss the benefits and shortcomings of choosing Twitter. We tried to get a sense of the impact on the stock market based on tweets regarding a given company. Our interest in programming led us to attempt to develop our own solution. The solution should be able to scour Twitter for mentions of the company that we want to investigate.

From this scraped data, we wanted to attempt to extract the general public sentiment regarding a given stock, in an effort to examine the correlation between stock price movements and sentiment. Fluctuations were defined as the difference between a given stock's opening and closing price. We quickly realized that there are several different ways to analyze data for sentiment and therefore wanted to try different methods and compare them against each other. Even though we developed our own solution, the code is not the main focus. It was a tool we wanted to use, and a tool that made us consider implementation of different methods. Following our general motivation, we converged on the following research question and related working questions:

1.2 Research Question:

How can an algorithm score sentiment from Twitter, and how do these results correlate to fluctuations in the stock market.

1.2.1 Working Questions

1. What elements are needed to develop a program that can predict sentiment?
2. What is the best approach for determining sentiment?
3. How can we test the validity and accuracy of our score?
4. What can be concluded when the approach is compared to real-world data.

1.3 Scope of Constraints

In this paper, we will focus on implementing a sentiment analysis library in Python and use that library to detect sentiment in tweets. We have chosen only to include sentiment perspectives, which means this paper will contain no economic models or otherwise conventional economic stock prediction techniques. We have constrained ourselves from making our own machine-learning model, as well as training a preexisting model ourselves. This paper should be viewed as an experiment to see if it is possible to predict stock behavior solely based on public sentiment, using readily available tools on social media platforms.

1.4 Inclusion of mandatory perspectives

The following sections will outline how the required Roskilde University Hum-Tek objectives have been included throughout this project. The curriculum for Hum-Tek prescribes that the bachelor project must include key concepts from the dimensions Subjectivity Technology and Society, Technological Systems and Artefacts, and Design & Construction. These dimensions must be included, in some form, to demonstrate knowledge within the interdisciplinary field of humanism and technologies. The inclusion consists of descriptions of how design, social, and technical sciences are included in the project and the preliminary considerations of the implementations.

1.4.1 Technological Systems and Artifacts

The dimension of Technological Systems and Artifacts focuses on how technology works. This project will fulfill this requirement in the methodology section of the paper, focusing on how our system was built and optimized. Our implementation will be shown via diagrams and code-snippets.

1.4.2 Design & Construction

The dimension of Design & Construction is included in several distinct levels throughout this project. This paper was structured and planned on a project management level using Kanban, a tool used in agile development. The method of agile workflow ensures a more dynamic approach, where work is generated through an iterative process. This suits the project well, as the goals and strategies have developed concurrently with continuous learning and research.

1.4.3 Subjectivity, Technology, and Society

The dimension of Subjectivity, Technology, and Society is included in the project by considering the way language can express sentiment and how this, in turn, affects stock prices. We will consider the way people express themselves on social media and how to process this data reliably. In addition, we will be considering the consequences of subjectivity, both in our methodology and the subjectivity of tweets.

Our inclusion of these dimensions embraces the interdisciplinary spirit of Hum-Tek. The assumption is that humanities and technologies should be observed in a context. Our report tries to combine different approaches to bridge the gap on a complex problem.

2 Theory

I understand a fury in your words, But not the words.

- William Shakespeare, *Othello*

In the following chapter, we will describe all the tools used in the analysis of our results and discussion related to the thesis question. These descriptions will build a general understanding of theories relating to the field. The later methodology chapter will then examine our concrete use of the said theories.

2.1 Heuristics

We have chosen to use a heuristic approach to solve the thesis of this paper. This approach will support our choice of methods and theory, and this will function as our epistemic justification for the conclusions drawn later in the paper. Heuristics is an approach to problem-solving. The results of the heuristic approach are to approximate a behavior that is sufficient for a short-term goal. Although this may not result in rigorous proof of behavior, it serves as an intermediate result that can be evaluated[22]

In layman's terms, heuristics can be seen as a trial and error approach. Heuristic approaches are widely used in complex fields where an optimal solution is hard to prove. A heuristic approach is common in computer science as time complexity grows and computation becomes expensive[23]

The heuristic approach works well in an agile work environment, where solutions are built iteratively. Each iteration/solution might give new insights that spark ideas for improvements and functionality. On a more practical level, a heuristic approach is a good candidate when doing sentiment analysis, where the goal is to comprehend human emotion via computation. A fool-proof sentiment analyzer is a far reach, but we can approximate a good result by combining and evaluating different tools.

2.2 Agile workflow

The agile workflow approach is a set of principles and core values initially thought out in 2001 by 17 like-minded developers as a new way of prescribing software development[2]. They were meant as a way to deal with the rigid and inflexible software development processes that had dominated the landscape until that point. The principles and core values were worded as diametrical opposites of the existing principles, thereby identifying and addressing the problematic areas. Soon after the publication of the Agile Manifesto by the group of developers as mentioned above, the advantages of the outlined constituting principles and values became evident for practitioners beyond the realm of software development. This resulted in the widespread adoption of the method by numerous other industries [2].

Among the core values of this method is the iterative approach, where continuous delivery of software and flexibility towards changing requirements is preferred over following long-term, requirement-heavy, rigid plans. Frequent status updates and meetings and a high degree of communication and collaboration among the members of the project team are essential to ensure a quick response to changing requirements and specifications. This approach is typically made using a period of development followed by a short period of evaluation, refining, and fixing potential flaws or weaknesses before the next development period or sprint starts all over.

2.2.1 Kanban

The word Kanban is a contraction between the Japanese words *kan* meaning visual and the word *ban*, meaning card, making Kanban translate to visual card. Toyota created Kanban to manage the demand for a large variety of car models, but not in a large quantity. This did not work well with Ford's production model this has previously dominated the industry, which improved cost and speed of production but only on a small assortment of cars. Toyota had to be innovative and make sure every employee knew how far along in the process each car was. Later, when looking for improvement possibilities for IT systems, David J. Anderson came across the Toyotas Kanban model, which improved workflow by letting the different bottlenecks in the system determine the workflow[14].

Kanban Core Practices

Kanban strives to improve workflow. To achieve this improvement, some core practices have to be followed. These are

1. Making Work visible
2. Limit Work in Progress (WIP)
3. Manage flow
4. Implement feedback
5. Improve collaboratively

[14,pp 18]

Making Work Visible

These practices are essential, and most would argue, the most crucial part of the Kanban. By doing this it helps the participants know how far along they are in progress, what needs to be done, and the problems in the process[14,pp 18].

Work In Progress Kanban helps a lot in understanding what needs to be done. By limiting workflow to certain parts of the process, the team can focus 100% on the most critical tasks at the current stage. Klaus Kaltenecker demonstrates this with the example:

"From an economic perspective, it is thus cleverer to carry out one operation 100% of the way rather than 10% of each of 10 operations. Therefore, to reduce the lead times and establish a continuous workflow, it is sensible to limit the number of operations carried out simultaneously at any given stage." - [14, pp 19]

Managing The Flow The flow is essential to the Kanban's success. Therefore, one must put great detail into making sure the Kanban flows fluently. Because of this, bottlenecks and other forms of blockers must get extra attention. This means either allocating resources to fix the issues or backlogging them for a more convenient time. The most important part is to be aware of the issues and act accordingly [14, pp 20].

Implementing Feedback Feedback ensures that the quality of the work is up to par. It also ensures that the Kanban cards go through an iterative process, which helps increase the quality of the product. Furthermore, this helps the employees understand the issue with their output and how to avoid further issues in the future[14, p.20-21].

Improve Collaboratively The best work is never done alone. This step ensures that a group of people helps the issues different people run into. This collaboration helps the individual and the Kanban as a whole. This also ties into the feedback step, as people understand and learn collectively how to fix issues[14, pp 22]

2.3 Sentiment Analysis

2.3.1 How to Detect Sentiment in text

Determining whether a text string is positive or negative can be difficult for humans to comprehend. Therefore, it is not expected for the computer to yield tremendous or even good results as elements such as sarcasm and irony, primarily through text, can be challenging even for humans.

2.3.2 What is Sentiment Analysis?

Sentiment analysis (also known as opinion mining) is used to discern subjective data into quantifiable data using Natural Language Processing (NLP).

Sentiment analysis is a way to figure out the sentiment of a selected group toward a specific subject. This includes people's opinions, sentiments, appraisals, attitudes, and emotions. It will usually look at negative or positive statements to understand the sentiment. Sentiment analysis has become increasingly important in business, especially in marketing, and it has recently become helpful in finding and understanding bot-generated opinions or Fake News[13, pp 1-5].

The document level is utilized to classify whether the sentiment of the entire document is negative or positive. An important thing to remember at this level is that it is implicitly assumed that all documents have a negative or positive sentiment. This means that the sentiment analysis will not be applicable to evaluate multiple entities at this level[13, pp 9].

Sentence level is like document level, but instead of looking at the whole document, one must look at each sentence. Here it is possible to have a neutral sentiment, meaning no opinion is expressed in the text[13, pp 9].

2.3.3 Practical Use

As stated above, sentiment analysis is especially important for business and political figures. It allows them to understand their intended demographic's thoughts, feelings, and opinions on a matter, making it a lot easier to cater more heavily towards them. With the rise of social media, opinions have never been easier to share and therefore gather. This has led to a considerable increase in companies' in-house social media departments, spreading through most segments of companies [13, pp 9].

Before this boom in access to opinion data, companies would conduct different kinds of surveys or opinion polls to get a better understanding. These surveys were a much more tedious process and would be very time-consuming.

"In recent years, sentiment analysis applications have spread to every possible domain, from consumer products, health care, tourism, hospitality, and financial services to social events and political elections. There are now hundreds of companies in this space, start-up companies and established large corporations, that have built or are in the process of building their own" [1, pp 5-6]

2.3.4 Two different approaches

Sentiment analysis can be done in many different ways. Researching the topic, we found that most approaches use machine learning. This process is done by using a model trained on manually annotated training data in order to predict linguistic tags on new samples. This process is also possible to perform without the use of machine learning by creating a comprehensive set of rules and annotating each text individually.

The benefit of machine learning for the annotation process is that it allows for the quick processing of new data. However, this relies on the underlying model and thus may not be as exact as the manually annotated text. Below we will outline two options that assume a linguistically annotated data set that is preprocessed (cleaned) accordingly. Our approaches are a small part of the options available for conducting sentiment analysis and should not be seen as an exhaustive review of sentiment analysis.

2.3.5 The rules-based approach

With a linguistically annotated text, manual rules can be made to gather valid contextual words from a sentence. This approach is called the lexical approach: "The lexicon-based approach involves calculating orientation for a document from the semantic orientation of words or phrases in the document" [34]. The process involves collecting important linguistic words from a text and then using an already annotated sentiment score to calculate the total sentiment of the text.

The rule-based approach allows for great customization in selecting words and scoring them, but it also requires a lot of different checks. Researching this, we read how a rule-based approach could be implemented presented by Dmitry Kan [35]. His paper "Rule-based approach to sentiment analysis" shows how different checks have to be made to extract sentiment with rules. The most specific behavior should be accounted for first. If a sentence passes through the specific checks, it will be rated as an entire sentence[35]. Rating words in the rule-based approach required a comprehensive set of rules. Additionally it is limited because for each exception, a new set of rules is required.

An example of a rule that can be applied to a text could be the following: For a given text, collect all the adjectives. These words would then be considered necessary in the later determination of sentiment. Another rule could be a rule for negation or amplification: given an adjective, can we find a word indicating negation or amplification in the word before it[34, pp 269].

After gathering relevant words, we can pass them to the rating algorithm. This would return a score for the list of words. The overall sentiment of the text would then be equal to the average sentiment score for the gathered words. The following is an example of how a sentence would be evaluated in the described approach.

Example sentence: "The company Tesla is very focused on delivering green cars"

Gathering adjectives: "focused, green."

Checking negations/amplifications: "[+]focused, green"

Lookup in scored words: focused = 2*1.5, green=0

Sentiment for sentence: 1.5

We declare the sentence positive since it has a positive score.

2.3.6 The automatic approach to sentiment

Without creating the rules ourselves, a pre-trained machine-learning model can be used to try and predict sentiment on a given text. As seen in the rule-based approach, the rating is limited to the words gathered via the rules.

However most sentiment analyzers use supervised machine-learning algorithms. For this, a big data set of text with the corresponding sentiment is needed. Then a majority of this data is used to perform training. The goal of this is to teach the model to make a connection between the text and the sentiment that has been assigned to it. This connection can be achieved through many different algorithms, but the important thing is the overall concept [32].

This approach can be used on a broad set of data since it is not necessary to make rules for each possible case that can occur in a text. However, this approach is likely to sacrifice the accuracy of sentiment that ruled-based scoring can give.

Our iterative approach has led us to try both approaches with varying success. Each method can be used with a different result, and an ideal and nuanced product might involve both where it makes the most sense.

2.4 Pre-Processing

2.4.1 Regular Expressions

Regular expressions is a general term for a sequence of commonly occurring characters that define a pattern of text that needs to be found, in this case, to be omitted as the process of data-cleaning occurs. Stephen Cole Kleene initially coined the term in 1951 as he invented a method of matching patterns in raw text data using a mathematical notation. The purpose of using regular expressions as a method of text preprocessing is very common as is seen using a wide range of tools for this process[33].

2.4.2 Use of Emojis

The concept of natural language interpretation (i.e., human to machine) consists of a broad category of data. As it is known from daily life, it can sometimes be hard to understand a text that the author has spent little time preparing, for example, in text messages. Assuming this, in part, is caused by the generation of emoticons, perhaps more commonly referred to as smileys or emojis and ASCII-art [11].

2.4.3 Use of Sarcasm and irony

Social media and online communication can be prone to suffer from the usage of sarcasm and irony. Whereby an author, in fact, intends to propose an argument of the exact opposite meaning of what their writing would be interpreted as observed literally. Furthermore, sarcasm and irony are especially hard to recognize in text form. Interpreting sarcasm and irony is not something that has been reliably done yet, at least not given the amount of data taken into the interpretation[15].

2.4.4 Use of negations and amplifications

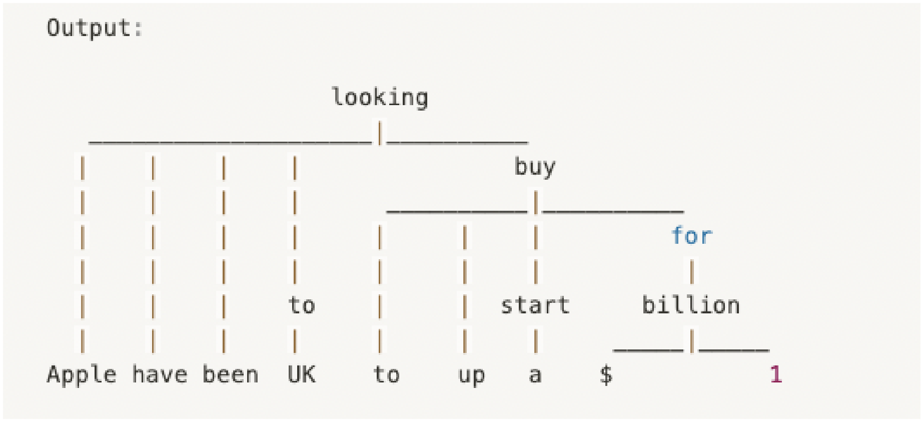
The use of negations can, for the majority, be bypassed using Regular expressions as most negated words follow the same structure of ”n't” suffixing the word ”not” into the verb. Such as ”isn't” or ”cannot.” However, for the sake of negations without the conjoining apostrophe, it does implicate some more nuanced solutions as the same method without the apostrophe would also remove a lot of other words such as ”extravagant,” ”discontentment,” etc. Just removing the verbs with this ending does not prove to be the solution either, as there are quite a lot of these words, for example, ”rant,” ”blunt,” etc.

2.4.5 Part-of-speech-tagging

Part of speech (POS) tagging gives tokens in a sentence, a tag related to their grammatical properties. These can be tags such as verbs, nouns, or numbers. The POS tags can be generated through dictionary entries of the word and their grammatical description. Ambiguity will occur, and rules must therefore be made to check the context of the word before assigning a POS-tag [18]. The result of a proper POS-tagging should be a well-annotated set of words, or tokens, that make up a sentence. These annotations give way to linguistic analysis that, with the proper domain knowledge, can make very sophisticated algorithms.

2.4.6 Word dependency

Word dependency or dependency grammar is a perspective within linguistics, particularly syntactic analysis. The core of the idea is that sentences consist primarily of binary asymmetrical relations, called dependency relations, that hold between words [37,chp 1]. Each word has a one-directional link to another word that modifies or compliments it. In this approach, phrase-structure rules do not have a role [1]. A classic example is a relationship between subject and object. These dependencies can be represented as a tree-structure.



```
Output:

                        looking
   ____________________|__________
  |     |     |     |             buy
  |     |     |     |     __________|__________
  |     |     |     |    |      |    |          for
  |     |     |     |    |      |    |           |
  |     |     |     to   |      |  start      billion
  |     |     |     |    |      |    |      _____|_____
Apple have  been   UK    to     up   a    $           1
```

In computer science, we could categorize the tree as a directional graph where vertices are words and edges the dependency relations. Such a graph structure depicting word dependency has been widely used in research regarding NLP. Most of this research has assumed a spanning tree, where every node is included (spanning condition) except the root (tree condition) [37, chapter 2.4]. We use such a dependency spanning tree for analysis regarding sentiment mining.

Word dependency, word embeddings, and POS-tagging are significant for sentiment analysis, but it is not feasible, or in the spirit of this paper, to do it manually.

2.5 Word Embeddings

"You shall know a word by the company it keeps"

- John Rupert Firth

Word-embedding is a technique for learning numerical representations for words to approximate their lexical meaning. These representations are constructed by parsing many words in their context of occurrence in very large data set.

The underlying idea behind word embedding is to understand a word by the context. The premise of this method is to compare each word in a vector space of a given number of dimensions. In the vector space, each dimension refers to a way of understanding each word within a threshold of similarity with other somewhat related words previously used in a similar context. In the context of sentiment analysis, it plays an essential part. It is used to map relations or similarities to other words. This is done by assigning numerical values to each word in each vector-space dimension. Closely related words will then lie close together in the vector space[30].

Furthermore, there are a plethora of different methods to calculate the word similarity.

For example, the word "green" might be close to the word "color," as well as "yellow" and "money," each in its own vector space, each representing one form of synonyms, origin, or etymology of the word in question.

2.5.1 Cosine similarity

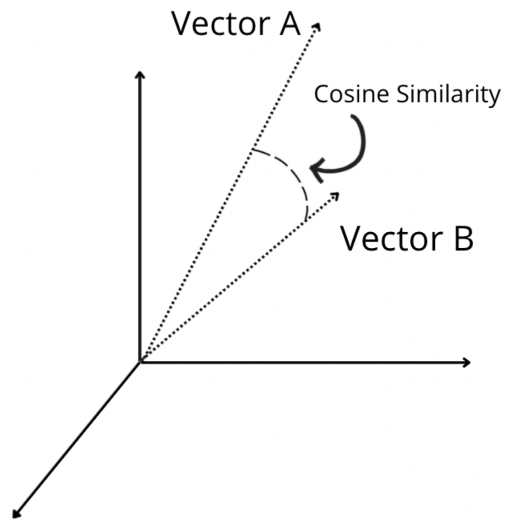
This chapter will include some examples created with the Python library SpaCy and Numpy as proof of concept. The important takeaway is to show how one part manually can do word-embedding. SpaCy and its use will be elaborated upon in the next chapter.

Cosine similarity is a commonly used metric for calculating the similarity between data objects or arrays. In cosine similarity, these arrays are treated as vectors. One of the advantages of using this method is that the length of the vector distance is irrelevant. Meaning that if some words are used more than others, it will ignore this and instead look for the way words are used, i.e., the degree between each vector. This can also be understood as the words are analyzed in relation to the surrounding words or the similarity between the compared words. This is not a major concern since we are using a pre-trained model.[29] It is calculated using the following formula.

$$Cos\theta = \frac{A.B}{(\|A\|.\|B\|)} \tag{1}$$

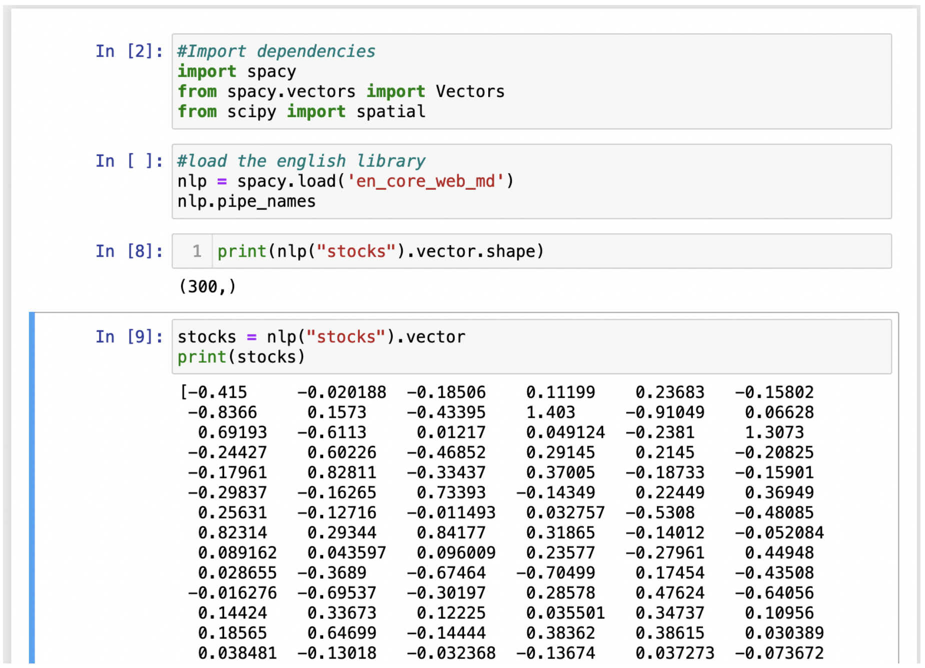
Where A and B are vectors, A.B is the dot product of A and B, And $\|A\|.\|B\|$ is the cross product of the two vectors A and B.

Illustration of Cosine Similarity



The length of Vector A does not change the Cosine similarity of between Vector A and Vector B. Using SpaCy's word model, each word is embedded into a 300 dimensional vector space and looks as seen below[28].

Words as vector shapes

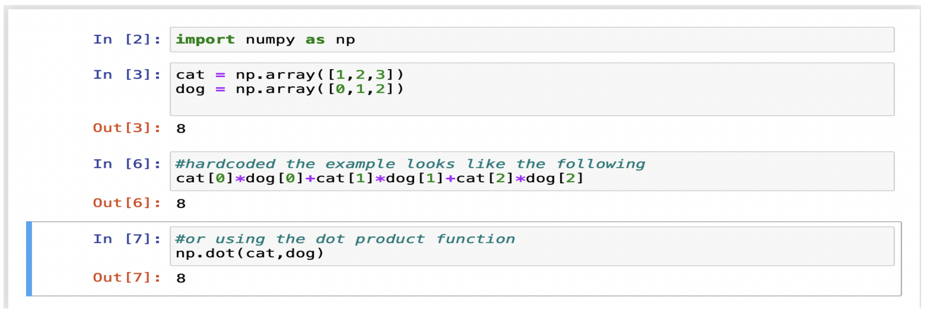


```
In [2]: #Import dependencies
        import spacy
        from spacy.vectors import Vectors
        from scipy import spatial

In [ ]: #load the english library
        nlp = spacy.load('en_core_web_md')
        nlp.pipe_names

In [8]: print(nlp("stocks").vector.shape)

        (300,)

In [9]: stocks = nlp("stocks").vector
        print(stocks)

        [-0.415     -0.020188  -0.18506    0.11199    0.23683   -0.15802
         -0.8366     0.1573    -0.43395    1.403     -0.91049    0.06628
          0.69193   -0.6113     0.01217    0.049124  -0.2381     1.3073
         -0.24427    0.60226   -0.46852    0.29145    0.2145    -0.20825
         -0.17961    0.82811   -0.33437    0.37005   -0.18733   -0.15901
         -0.29837   -0.16265    0.73393   -0.14349    0.22449    0.36949
          0.25631   -0.12716   -0.011493   0.032757  -0.5308    -0.48085
          0.82314    0.29344    0.84177    0.31865   -0.14012   -0.052084
          0.089162   0.043597   0.096009   0.23577   -0.27961    0.44948
          0.028655  -0.3689    -0.67464   -0.70499    0.17454   -0.43508
         -0.016276  -0.69537   -0.30197    0.28578    0.47624   -0.64056
          0.14424    0.33673    0.12225    0.035501   0.34737    0.10956
          0.18565    0.64699   -0.14444    0.38362    0.38615    0.030389
          0.038481  -0.13018   -0.032368  -0.13674    0.037273  -0.073672
```

In order to exercise an example of how the cosine similarity is calculated, a simplified example will be carried out. Assuming two words "cat" and "dog" with the following arrays:

$$cat = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \; dog = \begin{bmatrix} 0 & 1 & 2 \end{bmatrix} \tag{2}$$

Using NumPy, the dot product is calculated as follows:

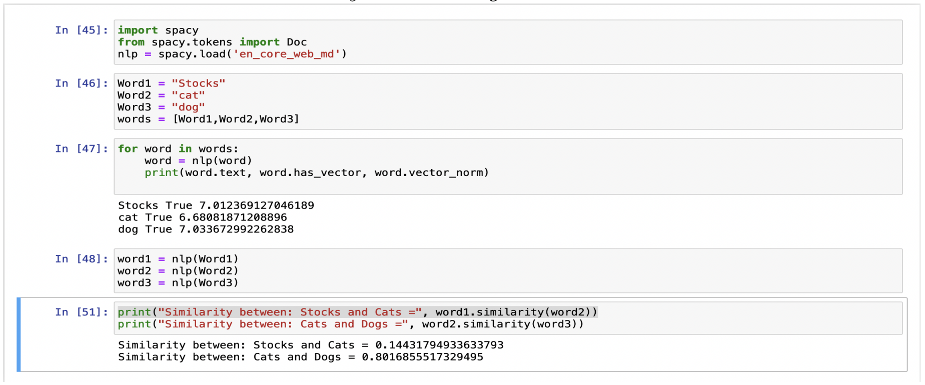


```
In [2]: import numpy as np

In [3]: cat = np.array([1,2,3])
        dog = np.array([0,1,2])

Out[3]: 8

In [6]: #hardcoded the example looks like the following
        cat[0]*dog[0]+cat[1]*dog[1]+cat[2]*dog[2]

Out[6]: 8

In [7]: #or using the dot product function
        np.dot(cat,dog)

Out[7]: 8
```

Calculating the dot product of the vectors as matrices using NumPy [29][26]. This, however, is just a simplified example to exercise the calculation of the dot product.

In reality, using larger arrays, this process can be quite a cumbersome task. However, since SpaCy is written in Cython, this process is parallelism and can be multi-threaded, meaning that more than one calculation can be done at once and in very few lines of code, using the **.similarity()** function in spaCy. Thus the cosine similarity between "cat" and "dog" using their respectable arrays as given by the spaCy en_core_web_md model will yield the following result.

Similarity Score using word as vector



```
In [45]: import spacy
         from spacy.tokens import Doc
         nlp = spacy.load('en_core_web_md')

In [46]: Word1 = "Stocks"
         Word2 = "cat"
         Word3 = "dog"
         words = [Word1,Word2,Word3]

In [47]: for word in words:
             word = nlp(word)
             print(word.text, word.has_vector, word.vector_norm)

         Stocks True 7.012369127046189
         cat True 6.68081871208896
         dog True 7.033672992262838

In [48]: word1 = nlp(Word1)
         word2 = nlp(Word2)
         word3 = nlp(Word3)

In [51]: print("Similarity between: Stocks and Cats =", word1.similarity(word2))
         print("Similarity between: Cats and Dogs =", word2.similarity(word3))

         Similarity between: Stocks and Cats = 0.14431794933633793
         Similarity between: Cats and Dogs = 0.8016855517329495
```

Yielding a cosine similarity between the word cat and dog as 0.801.

Using the NumPy library's **.dot()** and **.norm()** function we should be able to calculate the same cosine similarity between the two words vectorized into two different arrays. This however will require a conversion from SpaCys vector formatting to NumPy's array formatting.

```
In [45]: stocks = nlp("stocks").vector
         cat = nlp("cat").vector
         dog = nlp("dog").vector

         300

In [46]: cat_array = np.array(cat)
         dog_array = np.array(dog)

         300

In [47]: dot_product = np.dot(cat_array,dog_array)
         print(dot_product)

         37.67176

In [48]: cat_abs = np.absolute(cat_array)
         dog_abs = np.absolute(dog_array)

In [54]: (dot_product/(norm(cat_array)*norm(dog_array)))

Out[54]: 0.8016855
```

3 Methodology

This chapter will explain how theories and tools described in the theory chapter have been applied in our case. This will help to support our learning experience throughout this process of development. First, we will cover used libraries used. Then these methods and their implications will be critically discussed in the discussion chapter.

To answer our thesis question, we have decided to gather and process our own data. Throughout the entire analysis, we have been practicing agile development. This chapter will therefore be separated into three chapters:

1. Kanban and agile workflow
2. Gathering of data
3. Processing and analyzing data.

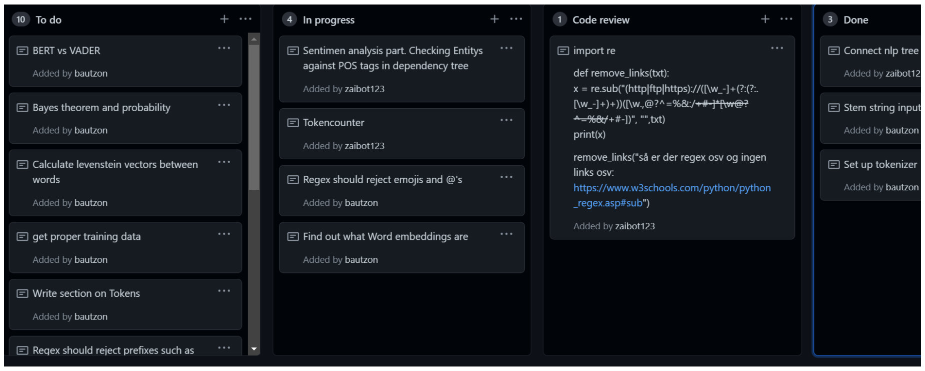
3.1 Kanban and agile workflow

Kanban

As mentioned earlier, Kanban provides excellent tools for agile project management. Since this project has two significant parts, the code and the paper, the project has been split in two, creating two separate Kanban's. One for the code and one for the paper. In this section, we will explain how we have used Kanban.

The Code Kanban

We created a Kanban board and divided it into four sections: To Do, in progress, Code review, and done. **To do** is what we intend to add to the project during the development of the code. The Kanban contains a mix of stuff we need and stuff that would be nice to have. Such tasks are ranked by priority. **In progress** is objectives that we are currently working on. **Code review** is review of the code. For this, we we tested and reviewed the code ourselves. **Done** are objectives that have gone through the testing phase and worked.


10 To do

BERT vs VADER
Added by bautzon

Bayes theorem and probability
Added by bautzon

Calculate levenstein vectors between words
Added by bautzon

get proper training data
Added by bautzon

Write section on Tokens
Added by bautzon

Regex should reject prefixes such as

4 In progress

Sentimen analysis part. Checking Entitys against POS tags in dependency tree
Added by zaibot123

Tokencounter
Added by zaibot123

Regex should reject emojis and @'s
Added by bautzon

Find out what Word embeddings are
Added by bautzon

1 Code review

```
import re

def remove_links(txt):
x = re.sub("(http|ftp|https)://([\w_-]+(?:(?:.
[\w_-]+)+))([\w.,@?^=%&:/~~+#-]*[\w@?~~
~~^=%&/~~+#-])", "",txt)
print(x)

remove_links("så er der regex osv og ingen
links osv:
https://www.w3schools.com/python/python
_regex.asp#sub")
```
Added by zaibot123

3 Done

Connect nlp tree
Added by zaibot12

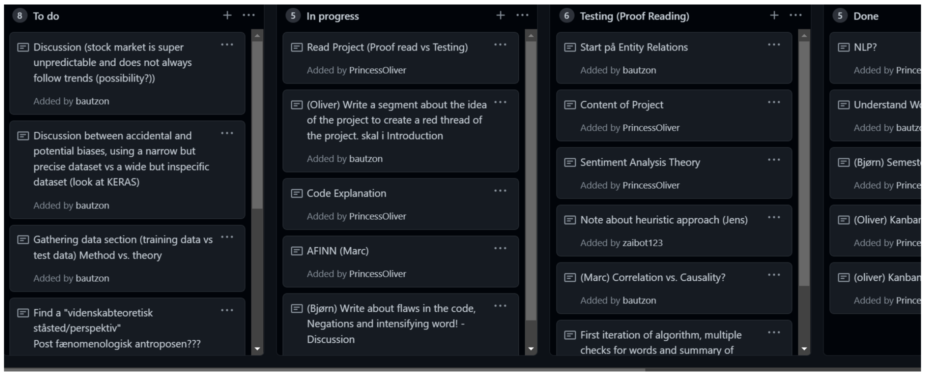
Stem string input
Added by bautzon

Set up tokenizer
Added by bautzon

Figure 1: Code Kanban

The Paper Kanban

Like the code Kanban, the paper Kanban has almost the same sections with the only exception being switching code review to proof reading.


8 To do

Discussion (stock market is super unpredictable and does not always follow trends (possibility?))
Added by bautzon

Discussion between accidental and potential biases, using a narrow but precise dataset vs a wide but inspecific dataset (look at KERAS)
Added by bautzon

Gathering data section (training data vs test data) Method vs. theory
Added by bautzon

Find a "videnskabteoretisk ståsted/perspektiv"
Post fænomenologisk antroposen???

5 In progress

Read Project (Proof read vs Testing)
Added by PrincessOliver

(Oliver) Write a segment about the idea of the project to create a red thread of the project. skal i Introduction
Added by bautzon

Code Explanation
Added by PrincessOliver

AFINN (Marc)
Added by PrincessOliver

(Bjørn) Write about flaws in the code, Negations and intensifying word! - Discussion

6 Testing (Proof Reading)

Start på Entity Relations
Added by bautzon

Content of Project
Added by PrincessOliver

Sentiment Analysis Theory
Added by PrincessOliver

Note about heuristic approach (Jens)
Added by zaibot123

(Marc) Correlation vs. Causality?
Added by bautzon

First iteration of algorithm, multiple checks for words and summary of

5 Done

NLP?
Added by Prince

Understand W
Added by bautz

(Bjørn) Semest
Added by Prince

(Oliver) Kanbar
Added by Prince

(oliver) Kanbar
Added by Prince

Figure 2: Paper Kanban

By separating the work into code and paper, we could more efficiently work on separate but equally important parts of the project at the same time. This approach also gave a mutual understanding of individual responsibilities, further ensuring that the five steps from the Kanban theory were fulfilled. We made sure to make the work visible by categorizing the work that needed to be done visually. We limited the work in progress by ensuring we would not get bottlenecked by the development of code by giving all project members an alternative objective in the paper Kanban. We managed flow by regularly updating the Kanban. Feedback was implemented in the form of either proofreading or code testing. Furthermore, we improved our collaborative work by using the Kanban to show each other what parts we were working on or experiencing trouble finishing.

3.2 Gathering of data

Although it is possible to use already gathered data, we seek to make a simple application to gather this data ourselves. This serves three purposes: **Firstly**, we wanted to implement code in our approach, and the gathering and processing of data seemed like a good fit. **Secondly**, having our own application means we can customize it as we see fit and therefore avoid any dependencies on outdated data. **Thirdly**, the data is essential when working with sentiment analysis. Thus, having a good grasp of the data scraped, we could also get a better idea of the possibilities made possible by the data. Although this process is done in the spirit of the scientific method, we are aware of the inevitable shortcomings and biases that we will encounter. Therefore, the methods and choices taken regarding our data gathering should be seen from the perspective of the heuristic approach. Traditionally, an attempt to predict stock prices is reserved for bankers and investors based on financial reports and quantitative data. However, we wanted to create a simple approach using sentiment analysis.

We had a hypothesis, based on anecdotal evidence that social media impacted the stock market. We, therefore, chose Twitter. In practice, we used the API provided by Twitter in the programming language Python. For the Twitter data, we used a Python module named Tweepy. Tweepy is a user-friendly

Python library for accessing the Twitter API. Using Tweepy gives us a way to access the Twitter API in a more abstract and user-friendly manner without sacrificing any of the extensive functionalities of the API itself.

3.3 Processing of data

After gathering the Tweets, it is good to perform some level of preprocessing. This preprocessing streamlines the data by removing unnecessary variables.

3.3.1 Regex

One of the text preprocessing steps involves cleaning the scraped tweets of unnecessary characters, words, and punctuation. We achieved this by using regular expressions via the RE python module. Regex is used in the function "remove_regex" which is a part of our cleaner functionality. The "remove_regex"-function takes the scraped tweets as input and first changes all the letters to lowercase. Then URLs are removed by checking if the word starts with either "http" or "https", followed by "://" and ending with ".XX". Similarly, emails are removed, using a somewhat similar approach as with the URLs. Next, Regex removes all occurrences of ampersands denoted by "amp;". Furthermore, punctuation such as "!", "?", "," has been removed, as the sentiment analyzer did not use it in the evaluation. The removal is done using Regex's built-in ".sub"-method by matching and then replacing correct matches with "", which corresponds to nothing, thereby removing the matched instance. However, small changes/modifications to the Regex-code were necessary for each major iteration.

3.4 Analyzing data using AFINN

AFINN is a .txt file with English words and a corresponding sentiment score. It was made by Associate Professor of the Department of Applied Mathematics and Computer Science at DTU, Finn Aarup Nielsen. He compiled and graded a comprehensive list of English adjectives and adverbs. The list was constructed by gathering words, slang, and abbreviations and their synonyms by examining tweets gathered, using pre-compiled word lists such as "The Compass DeRose Guide to Emotion Words" and "Original Balanced Affective Word List," and by using online dictionaries[20].

Nielsen evaluated the accuracy of the AFINN list by comparing the results, as well as the results of four other sentiment lexica to a data set consisting of 1.000 labeled, pre-rated tweets. Each tweet in the list was rated manually by ten different people to ensure consistent rating. The average score of each tweet was then compared to the score calculated by AFINN and the other four sentiment lexica (SentiStrength, Opinionfinder, ANEW, and General Inquirer). The test revealed that AFINN came in on second place, right behind SentiStrength[20]. The words in the AFINN list are graded on an integer scale ranging from -5 to 5, where -5 indicates the most negative sentiment, 0 indicates neutral sentiment, and 5 indicates overtly positive sentiment. The list has twice been expanded, going from 1468 words and phrases to 2477, and the most recent edition now contains 3382 words, phrases, and emoticons[39].
Since then, the list has been expanded several times in an attempt to create a list of rated words targeted specifically towards microblogging, as such a list did not exist at the time[20]. This suits the project's purpose well, and since AFINN was created for microblogs, it also includes internet speech and slang. This works in the project's favor since the way people write on Twitter is different from how they would talk or write in other places. For our sentiment analyzer, we converted the AFINN list into a Python dictionary where the word is the key and the score is the value. This provided a constant time operation when gathering scores for words.

3.5 SpaCy

To create a sentiment analyzer, we would need a way to annotate unseen data. Here we used the Natural Langue Processing (NLP) library for Python called SpaCy. SpaCy is an open-source NLP library for Python, and it comes with a default pipeline that can be further customized to the task at hand. To initialize the pipeline, a doc object has to be created. The doc object consists of tokens that each have different annotations such as dependencies, POS-tagging, and other NLP features ready to use. These tokens have been generated by a machine-learning model trained by SpaCy. This works by training the model using data that has already been annotated manually or part-manually by linguists. In the case of Spacy, the default English model is trained on a "large corpus comprising various genres of text" created by the Linguistic Data Consortium [28]. The goal is to annotate unseen data.

After gathering and cleaning the tweets, the next step is analyzing the sentiment. For this, there are basically two approaches, rules-based, and machine-learning. In our sentiment analysis, we are doing a rule-based approach (AFINN) and an "automatic" approach using machine learning (Word-embeddings). This correlates to the two approaches described in the previous chapter. Each of the following iterations will be a heuristic approach to developing both of these algorithms used to analyze data. Below we will describe how we implemented these two algorithms.

3.6 First Approach

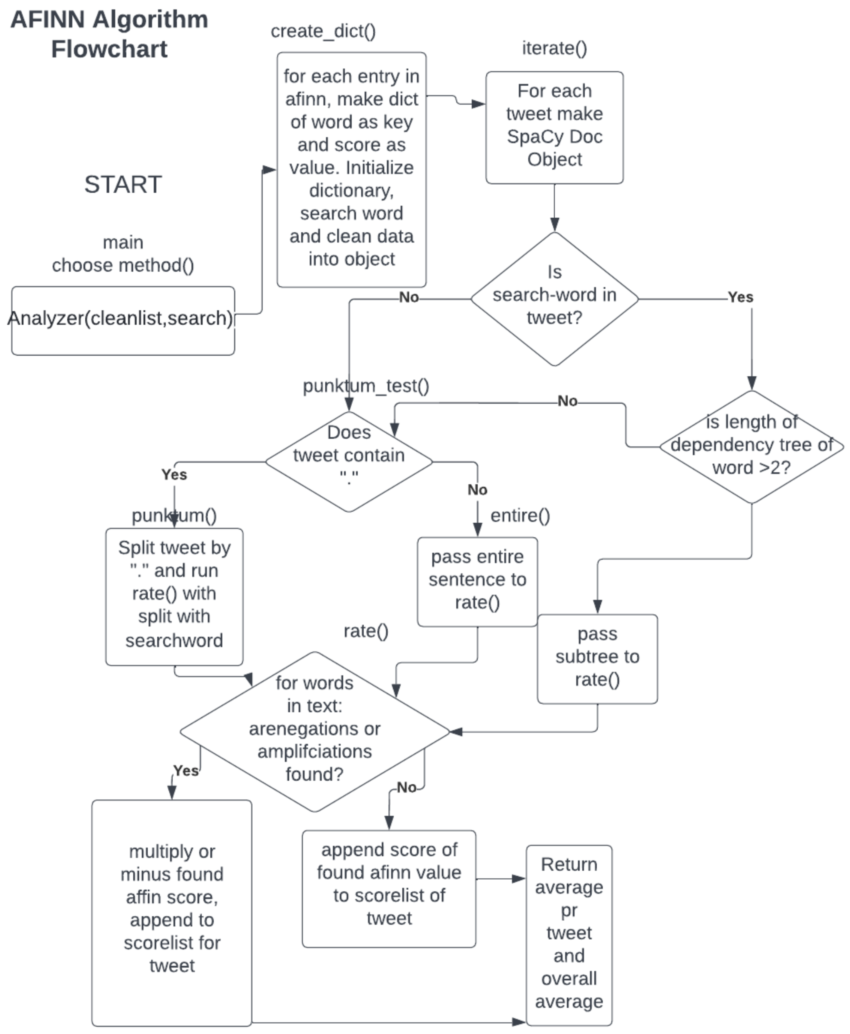
The first approach was to utilize the AFINN dataset to rate the sentiment by looking at the score of each word and then averaging them out for the entire tweet. We expected that the nature of AFINN would make the predicted words quite accurate but that it also would be limited due to its size. However, we first needed to gather the data to be matched in AFINN.

First, the data would need to be preprocessed by our Regex cleaning function. Each tweet would have to be split into a separate list. From this list of strings, the approach was to check each word in the tweet for the original search word given to the Twitter API. If tweets had the search word, we attempt to make a sub-tree of the word via dependency parsing in Spacy.

If the dependency-sub-tree has a length of over 2, we collect those words as our data for rating. If the word had no sub-tree of this length, we would test for a period. This test returned a boolean if there was a period in the tweet. If so, we would run a function to split the tweet by period and use the split where the search-word were inside. We assume that the words in this split would be of more importance since this search word was inside. If neither a tree nor a split-sentence could be made, we would use the entire data for rating. This would also happen if the tweet did not contain the search word. This would return in a list of words for each tweet.

For rating, we had made AFINN into a dictionary for constant-time-lookup. Then we would pass our previously made list of words, and then we would append the value of each word if it was present in the AFINN dictionary. This score would be multiplied by -0.5 if the current word followed a negation, such as "not," or 1.3 if it was following an amplification such as very.

An illustrative graphical representation of the code can be seen below.

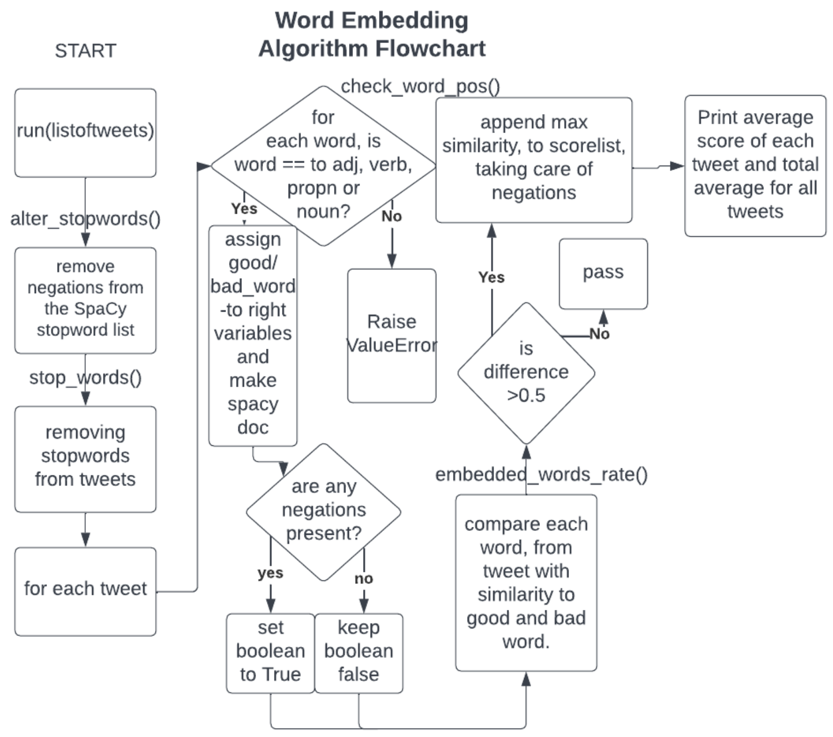

AFINN Algorithm Flowchart
START
main
choose method()
Analyzer(cleanlist,search)
create_dict()
for each entry in afinn, make dict of word as key and score as value. Initialize dictionary, search word and clean data into object
iterate()
For each tweet make SpaCy Doc Object
Is search-word in tweet?
No
Yes
punktum_test()
Does tweet contain "."
is length of dependency tree of word >2?
No
Yes
No
punktum()
Split tweet by "." and run rate() with split with searchword
entire()
pass entire sentence to rate()
pass subtree to rate()
rate()
for words in text: arenegations or amplifciations found?
Yes
No
multiply or minus found affin score, append to scorelist for tweet
append score of found afinn value to scorelist of tweet
Return average pr tweet and overall average

3.7 Second Approach

The limited annotated data in AFINN resulted in sentences that we could not rate. In an attempt to get a broader coverage of ratings, we made an iteration using word embedding. This would be able to rate nearly all words in a given sentence, although utilizing machine learning instead of human annotation. The idea was to gather more data for each word, even though it sacrificed some accuracy.

Like our first approach, we made use of cleaned data. In addition to the cleaning of data, we also removed stop-words. Spacy describes stop-words as: "List of most common words of a language that are often useful to filter out, for example 'and' or 'I' " [29]. This was unnecessary in the first approach as most of these words are not in AFINN. However, we realized that there were negations among the SpaCy stop-words such as 'not'. We made a list of words we wanted to search for when determining negations and made sure these words were not removed during our stop-words-removal.

The main loop in the word-embedding file uses spaCy's Part-Of-Speech (POS) tagging to determine, for each word, in each tweet, if the word is tagged as either a VERB or an ADJ. If this is the case, we calculate the cosine similarity from the word-embedding spaCy between the word and a word we have chosen. In our minds, it would be simple to make a pair of antonyms to test each word against. For example, a verb could be tested for similarity against "Good" and "Bad". Then the highest similarity would be returned and, if "bad", as a negative number. This would be the score for said word. Then another pair of words would be used for adjectives. In the same way as AFINN, we would compensate for negations. A graphic illustration of the algorithm can be found below.


Word Embedding Algorithm Flowchart
START
run(listoftweets)
alter_stopwords()
remove negations from the SpaCy stopword list
stop_words()
removing stopwords from tweets
for each tweet
check_word_pos()
for each word, is word == to adj, verb, propn or noun?
Yes
No
assign good/ bad_word -to right variables and make spacy doc
Raise ValueError
are any negations present?
yes
no
set boolean to True
keep boolean false
embedded_words_rate()
compare each word, from tweet with similarity to good and bad word.
is difference >0.5
Yes
No
pass
append max similarity, to scorelist, taking care of negations
Print average score of each tweet and total average for all tweets

4 Analysis

In this chapter, we will analyze our algorithms' performance individually and then do a comparative analysis between the two. At last, this chapter will investigate the correlation between stock-price fluctuations by applying the best-performing algorithm to relevant data. This data will be tweets from a period with a correlating drop in stock price. Then testing against a time period with a correlating rise in stock price.

4.1 Overall approach

After developing a way to rate tweets by sentiment, we needed a way of evaluating the results. The evaluation considers both our ability to gather relevant words from tweets and our rating. For testing purposes, we ran the program multiple times with different twitter queries. Then we printed the results to a file and tried to rate each tweet manually. We would then run the program on the same tweets and compare the results.

4.2 Evaluation of rule-based Approach

In the evaluation of the rule-based approach, we discovered a vital behavior: that some words worked a lot better than others. Among these were the names of people, products, and companies related to a given stock. This behavior was not in our favor as they would often yield worse results, as the algorithm could not parse some of these tweets. Two words commonly used to test this behavior were 'war' and 'apple' (tweets are all lowercase, so the search is too). 'war' would often make the program work as intended, returning a rating for almost all found tweets. 'apple' however, would return only a handful of requested ratings.

When printing the results of both, we found that the scraping was not the problem. After isolating the problem, we found that our check for search words in each tweet was responsible for this behavior. It turns out, tweets related to apple rarely use the word "apple" inside the actual sentence we are trying to analyze. However, in the case of #war, the actual search word is more generally included in a sentence and not just hashtagged after the actual tweet. The search word was initially included to ensure the scraped tweets were relevant. Nevertheless, this might have been the wrong approach. Maybe it is safe to assume that tweets will be relevant, as they were scraped from the hashtagged word.

Some days, some tweets yielded better results than others, which made our test inconsistent. After some testing, we found that we did not, in fact, use data that was lowercase but that our search word would be made lowercase. So in examples of "tesla", we would fail to identify "Tesla" as the search. Fixing this as well as allowing tweets without a search word to be parsed as an entire sentence instead of getting stopped in the first check for the search word yielded better results. However, there were other issues when testing different hashtags.

The word "war" generally carries negative connotations and thus might not be a good detector of sentiment. However, this was solely used to test the functionality of the scraping mechanism and not the sentiment result. There is also a level of ambiguity in the case of the words "apple" and "tesla" since these words are homonyms and have different meanings depending on the context. However, we argue that in the case of Twitter use, it is reasonable to assume that most tweets on the matter are related to the companies.

Tweets regarding organizations such as Apple would be more prone to advertisements or short tweets not carrying sentiment. The content of the tweets was, therefore, still tricky to rate, and we had to accept a worse quality of rating in certain hashtags due to the nature of how people Tweet.

For the words that worked the best such as 'war,' we agreed that, in most cases, our rating was correct. By rating, we mostly checked whether a tweet's score was negative or positive as interpreted by the AFINN scores' numerical value (either positive or negative numbers). A takeaway was also that we would need some threshold. Whereby a tweet's score is considered neutral. The tweets close to 0 were most likely wrong, and some could be interpreted both ways. The overall conclusion for the test of the first approach is: that given the right words, we can, in most cases, conclude whether a tweet has a negative or positive sentiment with a high level of certainty; however, we should have implemented a threshold for those cases where the sentiment was not very strong. We also found that many tweets would not get parsed as they were comprised of words not included in the AFINN list. The goal of the subsequent iterations should be to find a more general solution.

4.3 Evaluation of Second Approach

In a desire to catch tweets not processed through the first iteration and to test the different methods of performing sentiment analysis against each other, we implemented a second approach using embedded words. This approach attempted to create a more general solution to catch the tweets, for which the first method failed to analyze the sentiment. This approach was still somewhat naive, as it assumed that sentiment could be observed by calculating the similarity between two words using the **.similarity()** function from SpaCy. The initial version calculated the average distance between each verb or adjective of the tweet and the keywords "positive" and "negative". This approach did not take the difference between these words into perspective. As these two words are often used in the same context and thus, their co-sine similarity is quite closely related, yielding a very similar result as seen below:

```
positive is  0.8138099093626968 similar to negative
    and 1.0 similar to positive
difference : 0.18619009063730318
```

To develop a more domain-specific method, we altered the keywords used for analysis. These keywords were altered to words providing more meaning in an investment context. For example, "rising" or "falling". However, for a higher level of precision, it would probably have been better to include a list of keywords from which to analyze and then average each score out.

The next challenge arose as the structure of the sentence was altered before turning it into a SpaCy doc-object. This challenge occurred due to removing stop-words before converting the sentence into a doc-object, which meant that each word was tagged out of context. For example, words such as "love" and "hate" were interpreted as proper nouns instead of verbs because the prior pronouns had been omitted in the removal of stop-words. This could result in fallible results, as some tweets containing potentially sentiment-carrying words were not analyzed correctly.

This approach had a broader coverage of scores, but the scores were not as accurate as of the previous approach. The nature of word-embedding made it so we could score every word. However, the context-independent nature of our Word-embedding model made the scoring less accurate. This score is not a sentiment score but a marker of how closely related each word was with the keywords. In a more fleshed-out implementation of the possibilities with word embeddings, these problems would be less severe and allow the scoring to be more accurate.

This method was developed further by also accounting for nouns and proper nouns. Thus proper nouns were compared to proper nouns, and nouns were compared to nouns. This method was implemented to fix the above-mentioned problems relating to the words "love" and "hate."

4.4 Comparison of approaches

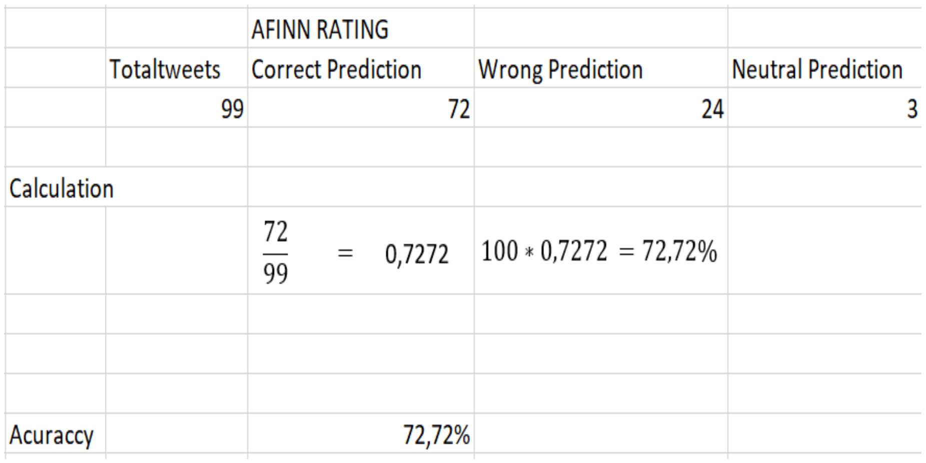
After evaluating each method individually, it was time to compare the two using a more extensive and more coherent data set. Due to a series of recent scandals, we assessed that Tesla might be a good candidate, as this stock is widely mentioned on social media and historically has had a volatile stock price.

To perform this comparative analysis, we collected a data set by scraping tweets related to tesla or tsla for the duration of the stock market opening hours over seven days. This scraping method only returned the most recent tweets from that given time. We should have performed this task more frequently as this could have significantly impacted the amount of data and given a more precise picture of the development of sentiments correlation to stock-price movements. Our method of data gathering was not ideal for several reasons. First and foremost because most Twitter users utilize Twitter's cash-tag function for tweets related to a given stock [21]. This is done on Twitter by using a '$' sign in front of the Ticker code for each stock. Twitter offers its own enterprise API for scraping this type of data; however, this option is used as a financial instrument, and they do charge a premium for this feature.

4.4.1 Accuracy of iterations

To analyze the accuracy of the two sentiment algorithms, we used the tweets that AFINN was able to rate from the dates 20th of May and the 23rd of May. Each tweet was manually rated on a scale of negative, neutral, and positive. Once all the tweets had been manually rated, a comparison between AFINN Word Embeddings scores could be made.

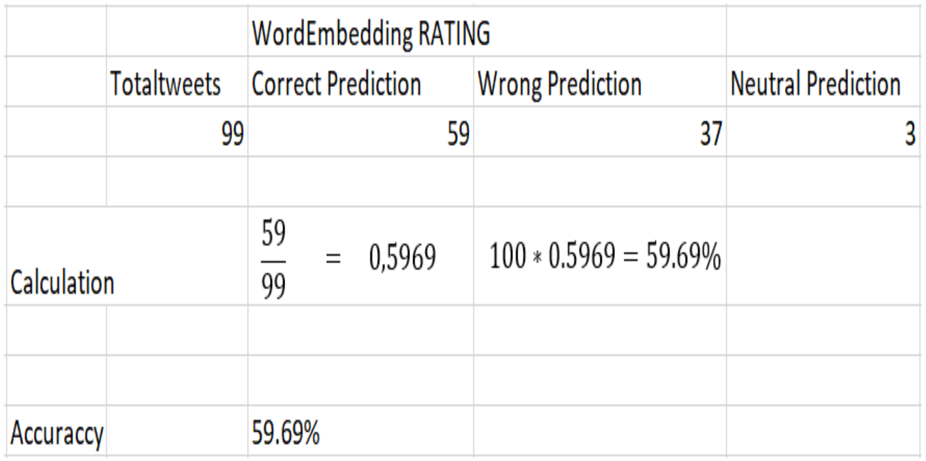
AFINN Accuracy: The result of this comparison saw that AFINN had a total of 27 correct predictions and 15 wrong predictions on the 20th and 45 correct predictions, 9 wrong predictions, and 3 neutral predictions on the 23rd. Three of the tweets were neutral, but AFINN predicted them as either positive or negative, so this counts towards a wrong prediction:



		AFINN RATING		
	Totaltweets	Correct Prediction	Wrong Prediction	Neutral Prediction
	99	72	24	3
Calculation				
		$\frac{72}{99}$ = 0,7272	$100 * 0{,}7272 = 72{,}72\%$	
Acuraccy		72,72%		

Overall, this gives AFINN 72.72% accuracy.

Word Embedding Accuracy: The same method was used for the Word embedding algorithm. It resulted in 24 correct predictions and 18 wrong predictions on the 20th and 35 correct predictions, 19 wrong predictions, and 3 neutral predictions on the 23rd. Just as with AFINN, word embeddings predicted the neutral tweets as either positive or negative - this also counted towards a wrong prediction.



		WordEmbedding RATING		
	Totaltweets	Correct Prediction	Wrong Prediction	Neutral Prediction
	99	59	37	3
Calculation		$\frac{59}{99} = 0{,}5969$	$100 * 0.5969 = 59.69\%$	
Accuraccy		59.69%		

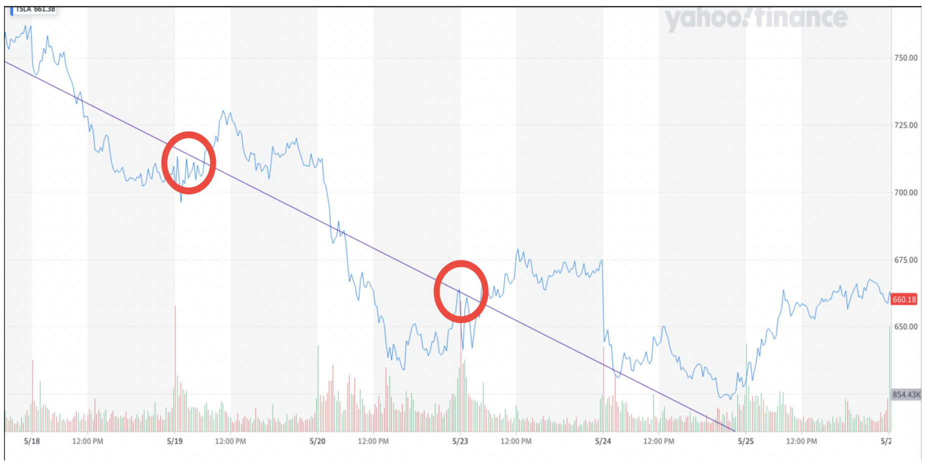
Overall, this gives Word embeddings a 59.69% accuracy.

Using this data, we could conclude that AFINN was 13.03% better at predicting sentiment than word embeddings, making AFINN the best of the two algorithms at predicting sentiment.

In conclusion, we have determined the rule-based approach using AFINN to be the best at evaluating sentiment. Because the accuracy of AFINN was better at analyzing sentiment than using embedded words, we will continue forward using the AFINN algorithm for the rest of the analysis.

4.5 Testing Algorithm vs the stock market

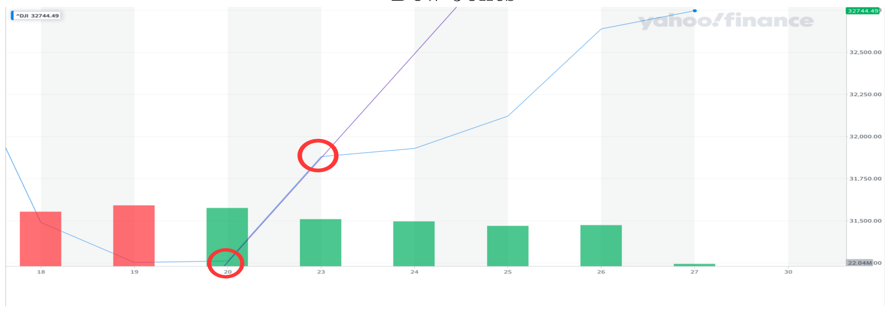
In order to determine whether or not the AFINN algorithm is valid, we would first need to find a period for which a given stock has had fluctuations. The Tesla stock was chosen as an example for reasons mentioned earlier. Prior to the period of the scraped data (20th and 23rd of May), Tesla faced a series of scandals which would, if our hypothesis holds, yield a falling stock price and a negative change in public sentiment. This expected effect seems to be present, with Tesla falling about 6% over the chosen period, as indicated in the graph below.



[1]

The above mentioned data was cherry-picked as we saw a drop in stock pricing during that period. This data made it possible for us to test if the fall in stock price was related to public sentiment on Twitter. The market was generally rising in the same period, as depicted below. The relationship between causation and correlation regarding this will be discussed later.

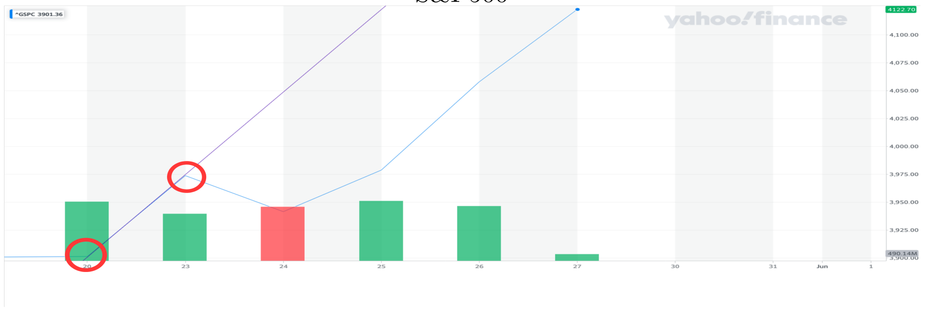
Dow Jones



[2]

[1]https://finance.yahoo.com/chart/TSLA/#
[2]https://finance.yahoo.com/chart/%5EDJI#

S&P500



[3]

For the sake of the AFINN method we observed a fall in sentiment.

Overall sentiment score: AFINN results.

$Date$	$OverallScore$
$20-05-2022$	$0,3844$
$23-05-2022$	$-0,0162$
$Total\ average$	$=0,3682$

While we did expect a similar result, the effect was far lower than anticipated, as the AFINN score does rate on a scale from -5 to 5. Furthermore, because neutral tweets were filtered out, only the tweets carrying a strong sentiment were analyzed.

4.6 Tests of a broader dataset

We also decided to analyze the correlation between the sentiment score derived from AFINN and the stock prices from 24-27th May. This comparison was made to understand how the sentiment correlates with the stock in question and whether that stock movement could be influenced by movements in the market.

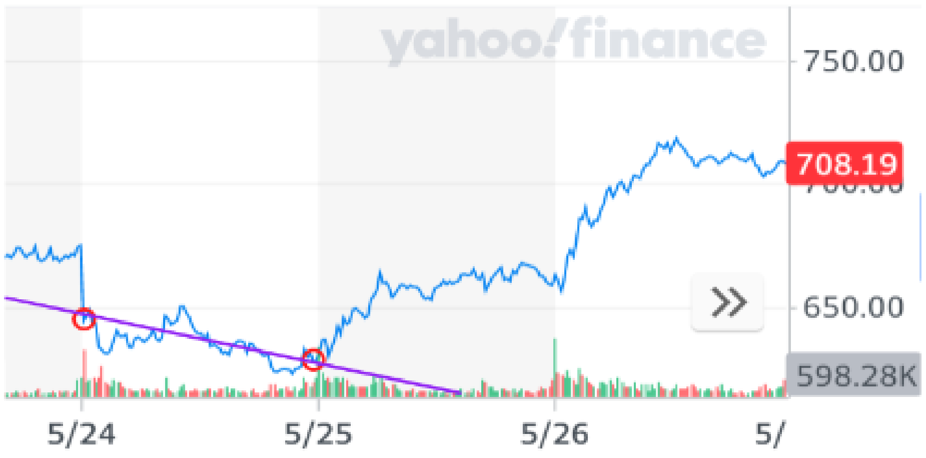
The same method was used for these additional four days. We scraped all tweets related to 'tesla' or 'TSLA' from 9:30 am to 4:00 pm, where the market is open, and ran it through the AFINN algorithm. These are the results:

[3] https://finance.yahoo.com/chart/%5EGSPC#

Overall sentiment score: AFINN results.

$Date$	$OverallScore$
$24-05-2022$	1.3548
$25-05-2022$	0.25
$26-05-2022$	2.1935
$27-05-2022$	0.5862
$Total\ average$	$1,0961$

This table shows that the sentiment was positive all four days. This means that if the hypothesis holds, we would be able to see a rise in Tesla's stock price every day. Depending on the sentiment, we would also like to see if it is possible to say that if the sentiment is higher than the previous day, the percentage rise in stock price should also see an increase from that previous day and vice versa.



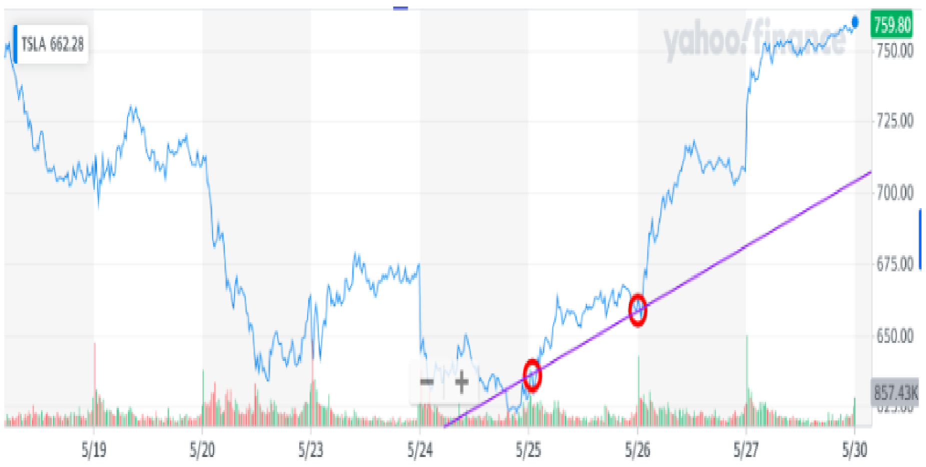
[4]

4https://finance.yahoo.com/chart/TSLA/#

Stock pricing on the 24th of May.

Time	*Price*
Open :	653, 53$
Close	628, 16$
Percentage	−3.90%

On the 24th of May, Tesla's stock price opened at 653,53$ and closed at 628.16$. This change was a loss of 3,90% in stock price. The sentiment analyzer scored the tweets a very positive sentiment of 1.35 total score from this day. This score should mean that the price should rise during the day. The price fell, so here the prediction failed. Because of this, it also failed to predict an increase in the percentage increase from the previous day.



5

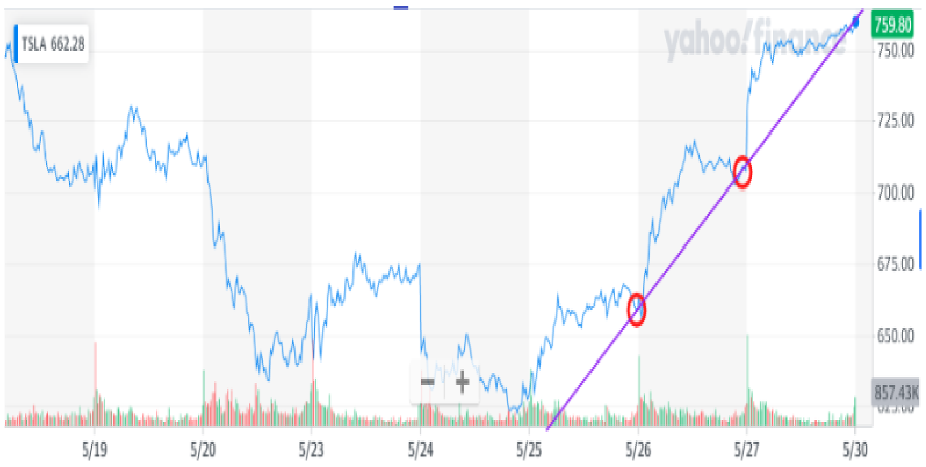
Stock pricing on the 25th of may.

Time	*Price*
Open :	623, 35$
Close	658, 80$
Percentage	+5, 60%

On the 25th of May, Tesla stock opened at 623,35$ and closed at 658,80$.

[5] https://finance.yahoo.com/chart/TSLA/#

This was a stock price increase of 5,60%. Compared to the sentiment analysis results, the sentiment was positive, with a total sentiment score of 0.25. This means that a positive sentiment could correlate with the change in stock price. Furthermore, this was an increase in percentage gain from the previous day.



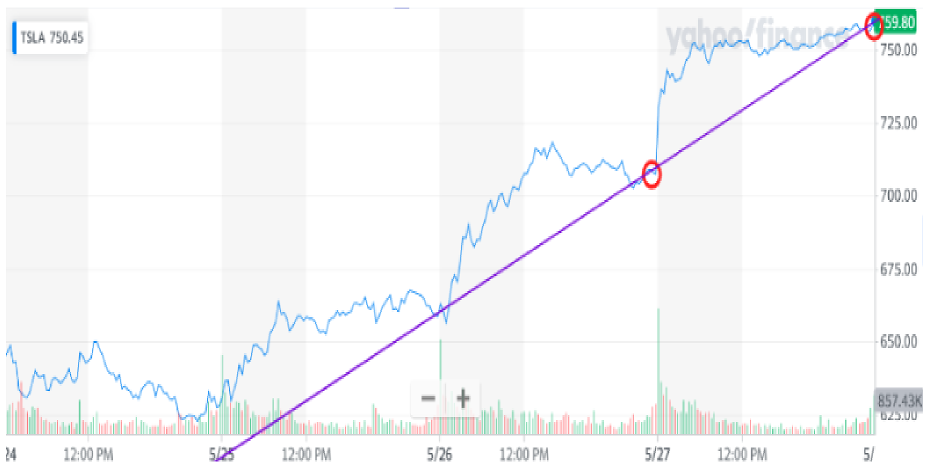
[6]

Stock pricing on the 26th of may.

$Time$	$Price$
$Open:$	$661,42\$$
$Close$	$707,73\$$
$Percentage$	$+7\%$

On the 26th of May, the stock opened at 661.42$ and closed at 707,73$. This was an increase of 7% in stock price. The sentiment analyzer gave the sentiment 2.1, which was a very positive sentiment. Here it was expected that with such a positive sentiment, the stock price would increase during the day, and by a higher percentage, than the previous day, and it has. 1.40% higher stock increase from the 25th to the 26th.

[6]https://finance.yahoo.com/chart/TSLA/#



[7]

Stock pricing on the 27th of may.

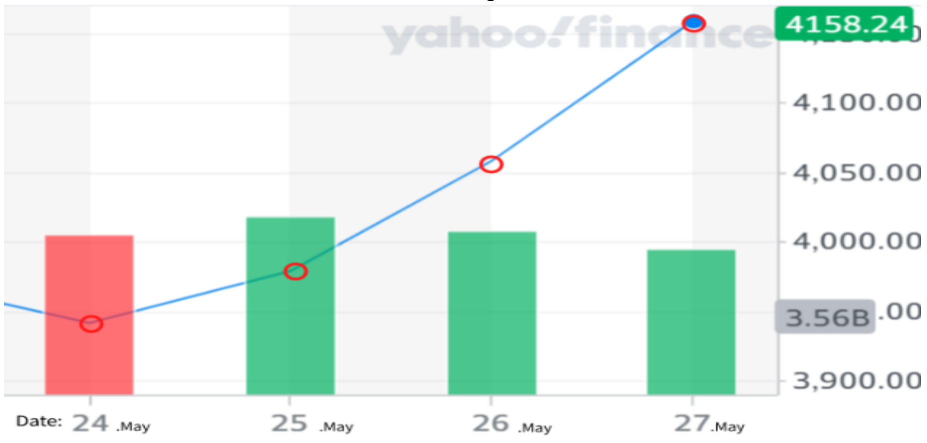
Time	*Price*
Open :	723, 25$
Close	759, 63$
Percentage	+5, 03%

On the 27th of May, tesla stock opened at 723,25$ and closed at 759,63$. This was an increase of 5,03%. Sentiment on this day was again positive, with a total score of 0.58. This was a lower sentiment than the day before, but still a positive sentiment. According to our hypothesis, the stock should increase in price during the day but by less than the 26th. This prediction was indeed accurate.

Overall, our hypothesis was proven correct on the 25th,26th, and 27th and incorrect on the 24th. These results mean that the theory was correct 75% of the time, correlating almost precisely with the AFINN algorithm's accuracy. Even though this seems like a clear correlation, a mistake in this data could be how the actual stock market performed in the same days.

[7]https://finance.yahoo.com/chart/TSLA/#

S&P500 market performance



[8]

Looking at the SP500 Index, we can see that on the 24th, the market fell, and on the 25th, 26th, and 27th, the market rose. This was the same trend Tesla's stock pricing had and could indicate that there is a good chance that Tesla's stock has a much larger correlation to the overall market than from the sentiment.

[8]https://finance.yahoo.com/chart/%5EGSPC#

5 Discussion

Following the results of our analysis, we can confirm that the best of our approaches is the rule-based approach. This approach detected a drop in the overall sentiment of the Tesla stock in a period where the Tesla stocks fell and could indicate when the price rose. It will also contain the criticisms of the paper in detail.

5.1 Correlation versus Causality

This project aims to uncover if it is possible to detect a correlation between stock price solely based on public sentiment from social media. While we, to some degree, are getting expected results from our analysis, we cannot conclude that these results would work outside our very limited test case. Doing so would be fallacious, as correlation does not necessarily equal causation.

In addition, causality can act in both ways and act self-reinforcing. Whereby a change in the stock price could result from a change in sentiment, this effect could also go the other way, with comments about the recent change in stock price further enforcing a given trend of change on social media. It is, however, also easy to think of scenarios in which the opposite happens, again creating a negative feedback loop. In contrast, the stock price due to unfortunate news drives the stock down, again reinforced by the associating comment can further worsen the general public sentiment.

Thus we are unable to conclude what way the causality points in general, as it might point in both directions. Some assumptions can be made on a tweet basis. Consider the sentence ”Fire in the Tesla Model 3 locks the door trapping the owner.” In such a sentence, the product of Tesla is the complaint, and it can be argued that in such a sentence, the sentiment would affect the stock price and not the other way around. However, in some cases, the opposite is true. A tweet hinting at how Tesla is ”doing worse than ever” is likely a conclusion based on the market value of Tesla. While these correlations still do not guarantee causation, they are worth considering when discussing the usefulness of the predictions. A further study into the mechanics of this would undoubtedly yield interesting results, but for the sake of this paper, it is out of scope.

An unclear correlation between correlation and causality and direction of causality means that the analysis results, though somewhat correct, could be caused by factors other than sentiment.

5.2 Hollistic vs naive

To examine pitfalls in our work, we will contrast them with a hypothetical holistic approach. Our description of this approach will likely not be exhaustive of possibilities due to our limitations regarding specific fields and domains. The holistic approach would, first of all, have a closer relation to the fields of economics. Not incorporating economic key numbers and basing a financial advising instrument purely on public sentiment in a minimal selection of economic fora is far from advisable. Given a company, Tesla, it would be helpful to monitor important economic events such as, e.g., change in yearly earnings (negative as well as positive), large buy/sell orders being filled, and analysts' price target updates. All of which could lead to sudden price movements. These factors have deliberately not been taken into account, but they would be accounted for in a more holistic approach. Another thing to keep in mind is the scarcity in the amount of data in our analysis which could also be a pretty significant pitfall.

Our analysis is based on a few "snapshots" in time and could be attributed to coincidence. Another randomly chosen drop in price would likely yield different results. To gain more confidence in our results, we should avoid cherry-picking data. A better approach would involve taking multiple long term time-periods into account to make a more general conclusion. A better approach would be to choose a more extensive selection of time periods showing price increases and price decreases. We could then average out the sentiment gain/loss and try to identify a correlation between our sentiment score and the price. Assuming that this correlation can be observed on a more general long-term data set, it would be interesting to investigate said correlation and how a change in sentiment relates to stock price development. A hypothetical example of such relations could be that for every 0.1 difference in sentiment, we would expect a 1% increase in Tesla stock. The model could also be tested against a linear regression analysis or any other economic stock price evaluating model performed on the actual stock price, which we could then compare to our model. Suppose we assume that a linear regression would be more accurate. In that case, we should aim to achieve a similar correlation or enhance the model further using a combination of the two. There is also the scenario that most companies with less eccentric chair members would not be possible to rate, as there would be very little to no data to scrape. Even after accounting for the different domains and methods, there is still an inevitable problem of human biases and subjectivity in this analysis.

5.3 High Level of Subjectivity

Concerning the overall heuristic approach, some degree of subjectivity has been involved with choices taken throughout our approach. When we tested the program's behavior, we scraped more or less arbitrary subjects to test for sentiment detection. These choices were unfounded and chosen at the moment. The best example of expressed subjectivity is in our two primary algorithms. In AFINN, we are parsing scores based on dependency trees. This decision was made to extract the most critical parts of tweets. Although we mainly got good results, the decision to use this method was based on initial test results from different twitter queries. In the word-embedding, we use words to compare against, that again are chosen by us. These methods have been tried, tested, and then altered if necessary. Nevertheless, these combinations were not something we stumbled upon from research. The results were the result of us using different methods to try and achieve an acceptable score.

5.3.1 Selection Bias

Since the project only focuses on the sentiment expressed and gathered through social media, the project will undoubtedly contain a certain amount of selection bias. The bias occurred when the data gathered came from a specific group of people and not a broader selection, causing the data not to represent the general population[24].

In the case of this project, we know that the sentiment was collected from Twitter users tweeting in English and including "TESLA" or "TSLA." The majority of twitter-uses are between the age of $18 - 34$ years old [8]. The selection bias could impact the sentiment, as most of the tweets analyzed will be from this specific age range. Furthermore, Twitter has most users from the United States, meaning that most tweets will come from Americans. The amount of unique users that fall into this category is unknown. So is the degree they represent of the total number of interest agents that might enjoy some form of utility from a given change in stock price. Due to only looking at the company Tesla in the analysis, selection bias comes into play. Using only one company for the analysis can have the side effect of showing data that is only applicable to that specific company, meaning that this approach to sentiment analysis might not work for every company. However, this bias is somewhat intended as we hypothesize that some stocks are more widely discussed online fora than others and that Tesla falls neatly into this category. Furthermore, all scraped tweets were chosen randomly from a specific posting time, meaning that no tweets or users were intentionally selected or left out. This process validates the work, but the project group does not ignore that a certain amount of bias is prevalent in this paper.

5.4 Grammatical Errors

Unlike some other models, AFINN does not account for grammatical errors. This means that while the sentiment of a given tweet might come across quite clearly to humans, it does not apply to the algorithm. It would have made much sense to look at the lexical text similarity between words that do not appear in the list and words that do. For words not caught in AFINN, we could use the score for the word with the highest similarity. An example of this could be that the word "Horrendous" is not in AFINN. However, the word "Horrible" has a similarity of 0.92; we would score it as the score belonging to "Horrible." Though this approach might have improved accuracy, it also comes with the downside of being computationally expensive.

5.5 Assets with no intrinsic value

Assets such as stocks carry an intrinsic value, although prices can be over-and undervalued. Depending on the size of shares, a shareholder has some influence on the company. For this reason, we suspect our approach might have yielded better results were we to look at assets with a price more dependent on public sentiments, such as the case for art and cryptocurrencies. As these assets carry no intrinsic value, we assume that their price is more heavily influenced by public sentiment. For cryptocurrencies, we believe it is reasonably safe to assume that this sentiment is online and, to a large extent, made publicly available.

5.6 High level of bot-generated sentiment

We did not consider a method for omitting auto-generated tweets by non-human entities, such as ads, spam, or giveaways. However, we did observe a significant amount of bot-generated tweets in our data. We suspect that this could have been accounted for by filtering data from users with no followers or practicing a particular type of behavior. However, as this would call for further research into bot behavior, we chose not to continue further down that route. Another argument why it should be included is that auto-generated tweets can still carry sentiment affecting the public opinion of a given matter. This is an exciting aspect, and we expect this to happen increasingly in the following years as this trend develops, just as seen in the 2016 U.S. presidential election, where several tweets from social bots accounted for about 20% of the online discussion on twitter[17].

5.7 Using pre-trained models

It is essential to consider what data the artificial intelligence model was trained on using a pre-trained model. The words are assigned a similarity based on the context they appeared in the data set from which the algorithm was trained. In our case, using a pre-trained word model from spaCy, it was primarily trained on Wikipedia [4]. However *"spaCy is not research software"* and "spaCy is opinionated"[18] The training data should ideally always be representative of the data one is trying to process. Therefore, it might give a worse result to use the SpaCy model on a specific domain like Twitter. For these other models, domain-specific, custom-made, or both might have proven more effective.

5.8 Following the news

Not admitting some form of bias in the fact that news of an attempted acquisition of Twitter by Tesla founder Elon Musk would be a mistake. Musk has been a long-time shareholder in Twitter. Further enforcing relation between Twitter Tesla. This information, however, was not known to begin with and has therefore not had an impact on our writing. However, in light of this information, it would have made more sense to look at other stocks with a less present public correlation. The acquisition seems to be temporarily put on hold as of writing this paper. According to Musk, this is due to pending details supporting the conclusion that the number of fake/spam accounts amounted to less than 5% [40].

5.9 Inter rater subjectivity and reliability

As mentioned earlier, the tweets that the AFINN model was able to rate were subsequently rated using the Word Embedding model. These tweets were then added to spreadsheets and manually rated with either a positive, neutral, or negative score, indicating their sentiment as perceived by the individual. Due to many tweets, we found it necessary to divide the work between us. Two group members split the tweets, rating roughly half of them each. Most tweets had exact sentiment, meaning the different group members in charge of rating would perceive and rate them identically. However, roughly a third of the tweets had some degree of ambiguous sentiment, making the rating less obvious as some tweets leaned towards a neutral sentiment. This ambiguity could be seen in tweets containing words that otherwise indicated positive and negative. An example of one of these edge-cases is the following:

RT @heydave7: Should Tesla sell their $BTC stake and use the proceeds to buy back $TSLA shares?

[9]

AFINN rated this specific tweet as expressing a positive sentiment, while word embedding rated the tweet as unfavorable. The group member in charge of rating perceived the tweet as primarily expressing neutral sentiment. As mentioned in the analysis, the tweets were manually rated as neutral, but either positive or negative by the algorithm was calculated as a wrong prediction.

[9] Appendix2: Excel Data

6 Conclusion

Based on related works and our own examination, we can conclude that public sentiment in our case could, in part, be used to determine a correlation between sentiment and stock pricing. This correlation was shown in the analysis section, where the prediction of stock pricing correlated nicely to our scoring accuracy, with an 2,80% variation. By improving the algorithms, accuracy could, in turn, improve the predictive capabilities.

Navigating the different tools within NLP, we managed to develop and design our own version of a sentiment analyzer. To validate these results, we had to account for subjectivity, both in the algorithmic design but also when we had to act as the arbiter for the sentiment of a tweet. Even though our prediction mapped well to the stock market in our tested cases, different pitfalls make it problematic to pinpoint the exact direction and dependency between causation and correlation. We would argue that, given a specific domain or company, our sentiment analyzer can deliver a correct sentiment score in most cases. However, it would be naive to conclude that this sentiment analyzer could be used as the only tool to attempt stock price predictions. Other factors in economics and statistics would be necessary to combine with the analyzer for a more nuanced and general solution. For a more complete solution, we would have to do more tests on more varied data regarding the number of tweets and the collection period.

Given this conclusion, many things are to be considered. When assessing the stock pricing, how do we take Twitter and other nontraditional factors into account when assessing the economy. If, and at what point, Do the 200 million Twitter users dictate the market. The correlation between public sentiment and stock pricing does not just concern human activity on the internet but also bots that aim to sabotage or abuse. In summary, we can conclude that it is possible to see a correlation between stock pricing and social media sentiment. Through the analysis, we can conclude that the best of the two algorithms created was the rule-based AFINN approach and that the AFINN scores correlated to stock pricing.

7 Bibliography

[1] D. Jurafsky and M.H. James, ”Speech and Language Processing”, 2021. Accessed: May 2022. [Online]. Available: https://web.stanford.edu/ jurafsky/slp3/14.pdf

[2] Beck, et al., The Agile Manifesto. Agile Business Consortium, 2001. Accessed: May 2022. [Online]. Available: https://cdn.ymaws.com/www.agilebusiness.org/resource/resmgr/docum

[3] Ferrara, “Social Bots Distort the 2016 US Presidential Election Online Discussion,” First Monday, vol. 21, no. vol 21, Art. no. 11, 2017, doi: https://doi.org/10.1111/j.1540-6261.1997.tb03807.x.

[4] R. Excell, “Investment Exchange Forum - Natural language Processing,” First Monday, 2021. https://www.cfachicago.org/podcast/investment-exchange-forum-natural-language-processing/

[5] B. De, L. H. Summers, and R. J. Waldmann, “Noise Trader Risk in Financial Markets,” Journal of Political Economy, vol. vol. 98, no. no. 4, Art. no. 4, 1990, [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0167923612000875

[6] T. D. team, “Date: Dictionary: Enterprise,” Developer Twitter, 2022. https://developer.twitter.com/en/docs/twitter-api/enterprise/data-dictionary/native-enriched-objects/entities

[7] Statista, “Leading countries based on number of Twitter users as of January 2022,” Statista, 2022. https://www.statista.com/statistics/242606/number-of-active-twitter-users-in-selected-countries/ (accessed May 2022).

[8] Statista, “Distribution of Twitter users worldwide as of April 2021, by age group,” Statista, 2022. https://www.statista.com/statistics/283119/age-distribution-of-global-twitter-users/ (accessed May 2022).

[9] A. Tumasjan, T. Sprenger, P. Sandner, and I. Welpe, “Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment,” Proceedings of the International AAAI Conference on Web and Social Media, vol. 4, no.

no. 1, Art. no. 1, 2010, [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/14009

[10] A. Shleifer and R. W. Vishny, "The Limits of Arbitrage," The Journal of Finance, vol. 1, no. 52, Art. no. 1, Mar. 1997, doi: 10.1111/j.1540-6261.1997.tb03807.x.

[11] Department of Software Engineering, Bahcesehir University, Besiktas, Istanbul, Turkey, S. Ayvaz, and M. O. Shiha, "The Effects of Emoji in Sentiment Analysis," International Journal of Computer and Electrical Engineering, vol. 9, no. 1, Art. no. 1, 2017, doi: 10.17706/IJCEE.2017.9.1.360-369.

[12] Q. Xu, V. Chang, and C.-H. Hsu, "Event Study and Principal Component Analysis Based on Sentiment Analysis – A Combined Methodology to Study the Stock Market with an Empirical Study," Information Systems Frontiers, vol. 22, no. 5, Art. no. 5, Oct. 2020, doi: 10.1007/s10796-020-10024-5.

[13] J. Zhao, K. Liu, and L. Xu, "Sentiment Analysis: Mining Opinions, Sentiments, and Emotions," Computational Linguistics, vol. 42, no. 3, Art. no. 3, Sep. 2016, doi: 10.1162/COLI_r_00259.

[14] K. Leopold and S. Kaltenecker, Kanban Change Leadership: Creating a Culture of Continuous Improvement. John Wiley Sons, Inc, 2015. doi: 10.1002/9781119019732.

[15] D. I. H. Farias and P. Rosso, "Irony, Sarcasm, and Sentiment Analysis," Elsevier, 2017, pp. 113–128. doi: 10.1016/B978-0-12-804412-4.00007-3.

[16] R. P. Schumaker, Y. Zhang, C.-N. Huang, and H. Chen, "Evaluating sentiment in financial news articles," Decision Support Systems, vol. 53, Art. no. 3, Jun. 2012, doi: 10.1016/j.dss.2012.03.001.

[17] A. Bessi and E. Ferrara, "Social bots distort the 2016 U.S. Presidential election online discussion," First Monday, Nov. 2016, doi: 10.5210/fm.v21i11.7090.

[18] "spaCy 101: Everything you need to know · spaCy Usage Documentation," spaCy 101: Everything you need to know. https://spacy.io/usage/spacy-101 (accessed May 2022).

[19] Twitter, "Twitter Announces First Quarter 2022 Results." https://s22.q4cdn.com/826641620/files/doc_fi
Q1%e2%80%9922-earnings-release.pdf (accessed May 2022).

[20] F. Å. Nielsen, "A new ANEW: Evaluation of a word list for sentiment analysis in microblogs." http://www2.imm.dtu.dk/pubdb/edoc/imm6006.pdf (accessed May 2022).

[21] "Native Enriched entities object." https://developer.twitter.com/en/docs/twitter-api/enterprise/data-dictionary/native-enriched-objects/entities (accessed May 2022).

[22] O. F. Kirkeby, "heuristik — lex.dk," Den Store Danske. https://denstoredanske.lex.dk/heuristik (accessed May 2022).

[23] J. Chen, "Heuristics," Investopedia. https://www.investopedia.com/terms/h/heuristics.asp (accessed May 2022).

[24] "Selektionsbias." https://metodeguiden.au.dk/selektionsbias (accessed May 2022).

[25] "Python - Sentiment Analysis using Affin," GeeksforGeeks, Nov. 2020. https://www.geeksforgeeks.org/python-sentiment-analysis-using-affin/ (accessed May 2022).

[26] S. Yıldırım, "Linear Algebra Basics: Dot Product and Matrix Multiplication," Medium, Jun. 2020. https://towardsdatascience.com/linear-algebra-basics-dot-product-and-matrix-multiplication-2a7624942810 (accessed May 2022).

[27] "Python — Word Similarity using spaCy," GeeksforGeeks, Jul. 2019. https://www.geeksforgeeks.org/python-word-similarity-using-spacy/ (accessed May 2022).

[28] "spaCy 101: Everything you need to know · spaCy Usage Documentation." https://spacy.io/usage/spacy-101 (accessed May 2022).

[29] "Cosine Similarity," GeeksforGeeks, Oct. 2020. https://www.geeksforgeeks.org/cosine-similarity/ (accessed May 2022).

[30] Codecademy. Available: https://www.codecademy.com/paths/natural-language-processing/tracks/nlp-language-quantification/modules/nlp-word-embeddings/lessons/word-embeddings/exercises/introduction (accessed May 2022).

[31] “Linguistic Features · spaCy Usage Documentation,” Linguistic Features. https://spacy.io/usage/linguistic-features (accessed May 2022).

[32] “What is Supervised Learning? — IBM.” https://www.ibm.com/cloud/learn/supervised-learning (accessed May 2022).

[33] L. KARTTUNEN, J-P. CHANOD, G. GREFENSTETTE, and A. SCHILLE, “Regular expressions for language engineering,” Natural Language Engineering, vol. 2, Art. no. 4, 1996, doi: 10.1017/S1351324997001563.

[34] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, “Lexicon-Based Methods for Sentiment Analysis,” Computational Linguistics, vol. 37, Art. no. 2, Jun. 2011, doi: 10.1162/COLI_a_00049.

[35] D. Kan, “Rule-based approach to sentiment analysis at ROMIP 2011,” 2012.

[36] N. Periwal, “Twitter Sentiment Analysis for Beginners.” https://kaggle.com/stoicstatic/twitter-sentiment-analysis-for-beginners (accessed May 2022).

[37] de Marneffe, Marie-Catherine and J. Nivre, “Dependency Grammar,” Annual Review of Linguistics, vol. 5, Art. no. 1, Jan. 2019, doi: 10.1146/annurev-linguistics-011718-011842.

[38] Python — Named Entity Recognition (NER) using spaCy, Geeks for Geeks, June 2019, https://www.geeksforgeeks.org/python-named-entity-recognition-ner-using-spacy/?ref=gcse, (Accessed May 2022)

[39] F.Å. Nielsen (2015). AFINN-en-165.txt [online]. https://github.com/fnielsen/afinn/blob/master/afinn/d en-165.txt (Accessed May 2022).

[40] Twitter,Elon Musk (@elonmusk) May 13, 2022