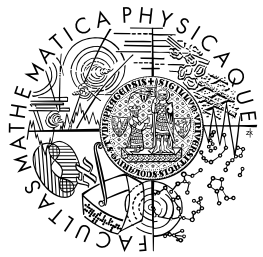


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

**BACHELOR THESIS**

Václav Stibor

# Financial News Sentiment Analysis

Department of Software Engineering

Supervisor of the bachelor thesis: doc. RNDr. Irena Holubová, Ph.D.
Study programme: Computer Science

Prague 2024

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date .....  .....

Author's signature

I would like to dedicate this thesis to my family and friends who have supported me throughout my studies. Special thanks to my supervisor doc. RNDr. Irena Holubová, Ph.D. for her expertise and guidance during the development of this work. To all who have assisted me, I sincerely appreciate your support.

Additionally, I am grateful for the consultations provided by Mgr. Petr Škoda, Ph.D., and Ing. Pavel Koupil, Ph.D. contributed to this thesis's improved outcomes.

Title: Financial News Sentiment Analysis

Author: Václav Stibor

Department: Department of Software Engineering

Supervisor: doc. RNDr. Irena Holubová, Ph.D., Department of Software Engineering

Abstract: In the current era of information overload, monitoring associations and comprehending media and online news content, especially for investment purposes, is becoming increasingly demanding. This thesis addresses the challenge by developing an application that evaluates sentiment analysis and visualises connections between companies and news articles. The cornerstone of the application is the extraction of data from news articles, sentiment analysis of the extracted entities, and the provision of this information as an indicator of potential future influences on a company's stock price. The application is designed to be easy to use and intuitive, allowing users to navigate to the articles and perform their analysis to verify the values provided by the application.

Keywords: entity-level sentiment analysis news stock market

# Contents

# Introduction

In today's era of information explosion and constant flow of information, it becomes more time-consuming to keep track of associations and deeply understand the published content through media and online news, primarily when investing in a specific area. For instance, the investment in a company like Apple Inc. requires acquiring and processing a wide range of available information with significant effort and dedication in studying articles and other sources. At the same time, publicly available information resources such as news articles and tools like sentiment analysis allow us to transfer real-world context into the digital environment and use it for our benefit.

Sentiment analysis, the ability to identify and evaluate the emotional charge of content, has evolved into a crucial instrument for comprehending opinions and the general atmosphere surrounding various topics. This work focuses on developing an application that allows users to visualise connections between companies and news articles using a graph network and the impact of news sentiment on a company's stock price.

Many experiments are currently being conducted based on historical data to examine the effect of sentiment, with promising results on datasets. The absence of such an application motivates this thesis. An application that extracts actual data from news for sentiment analysis and subsequently provides this information as an indicator of the future possible influence of a company's stock price.

One potential reason for the absence of such an application may be working with a constant flow of updating data, which can present a challenge to creating a functional application. Due to the valuable nature of news sources as information providers, they protect their data against similar usage. Despite the public availability of this data, legal complications may arise from potential violations of terms and conditions, as presented in the OpenAI, Microsoft and The New York Times dispute outlined in Chapter 3.

Another reason is that this is still an experimental technology and a speculative approach to the market. These technologies are still evolving, and there is no clear-cut approach to $100\%$ stock price prediction depending on news. Nevertheless, published studies involving experiments suggest correlations impact between stock prices and news data. Another notable aspect is shown in Figure 1, indicating the correlation between the number of news mentioning a company and its market price. Although it may be less than $100\%$ accurate in prediction, it is an exciting topic that can be very useful for examining the impact on prices and providing an overview of the situation.

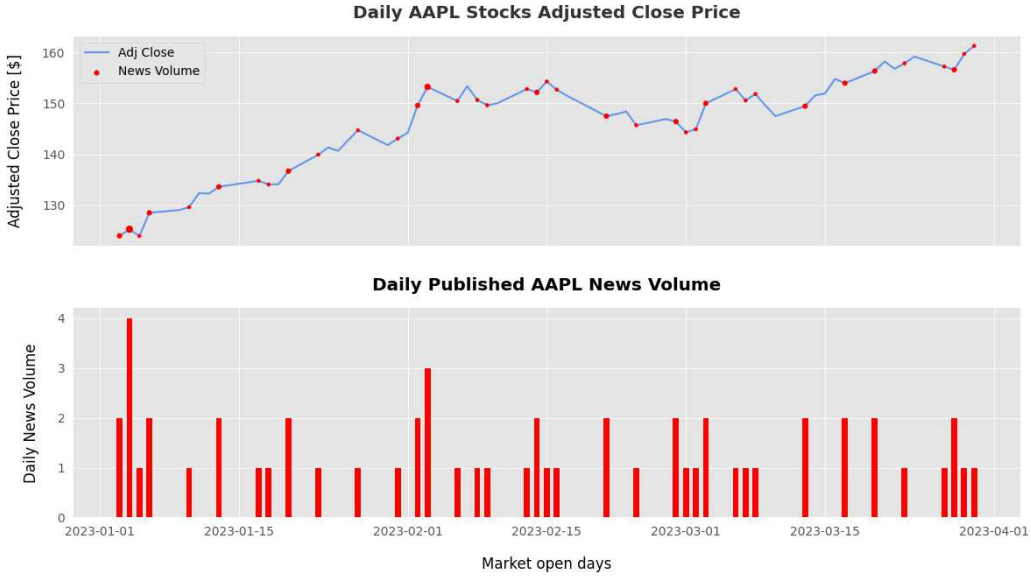



**Figure 1** Apple Inc. (AAPL) adjusted close price and the volume of news mentions in the Guardian's business and technology section in the first quarter of 2023.

Providing users with high-quality and reliable data is essential. Therefore, we will source data from trusted and verified newspapers and data providers. Additionally, we need to develop our algorithm for processing all data in the pipeline from extraction to recognition entity, analysis of sentiment, and loading to the database, considering there are few applications of this type, especially those focusing on entities, which is the ideal choice in our context.

The thesis is structured as follows: Chapter 1 discusses the foundational concepts of sentiment analysis and named entity recognition. In Chapter 2, we review existing applications and their methodologies in sentiment analysis, as well as recent studies related to sentiment analysis at the entity level and stock market behavior. Challenges associated with obtaining data from newspaper sources are addressed in Chapter 3, while Chapter 4 focuses on detecting entities in text and extracting company names with their ticker symbols. In Chapter 5, we develop an algorithm for entity-level sentiment analysis of the extracted entities and explore how this information can be used for market analysis. After describing the non-functional and functional application requirements in Chapter 6, Chapter 7 covers the application architecture based on the previous chapters. We present the developer documentation in Chapter 8. Chapter 9 is dedicated to the user documentation. Finally, the conclusion summarizes the results and discusses potential improvements and future application development.

# 1. Theoretical Background

Since the application's core is sentiment analysis, it is necessary to define the basic concepts. This chapter provides an overview of sentiment analysis, focusing on the levels of sentiment analysis and named entity recognition. We emphasised that sentiment analysis is most suitable for our application at the entity level, especially in the context of financial texts. Since sentiment analysis in different domains and languages faces many challenges, we decided to focus exclusively on the financial domain in English, allowing us to achieve higher accuracy and specificity in our analyses.

This chapter also introduced the techniques used in named entity recognition, which are crucial for entity-level sentiment analysis. These techniques include knowledge-based approaches, attribute engineering, and deep learning, proving particularly effective in capturing contextual information in text and achieving state-of-the-art results in entity recognition.

## 1.1 Sentiment Analysis Basics

Sentiment Analysis (SA) or opinion mining is a subfield of Natural Language Processing (NLP) that aims to identify and extract opinions and emotions from a text. The goal is to determine the author's attitude towards a particular topic or the overall contextual polarity of various document levels. We measure the text's polarity using a numerical scale ranging from $-1$ to $1$. The low-end score of the scale signifies a negative sentiment, zero represents neutrality, and the high-end score indicates a positive sentiment. This scale effectively estimates the degree of negativity or positivity in the text's tone.

The extraction of opinions and emotions has applications in various areas, from product reviews to political events. Hence, it is imperative to work in different domains (see Piryani et al., 2017). Because cross-domain and cross-language are two of the most general issues in sentiment analysis, this thesis will focus only on the financial domain in English. Nevertheless, domain-specific sentiment analysis achieves remarkable accuracy while staying highly domain-sensitive, as shown in (Saunders, 2020). To delve deeper into cross issues, Liu provides further details in his book (Liu, 2022).

### 1.1.1 Levels of Sentiment Analysis

Sentiment analysis has been studied at several levels of granularity: Document-level, Sentence-level, Phrase-level, and Entity-level[1], as illustrated in Figure 1.1.

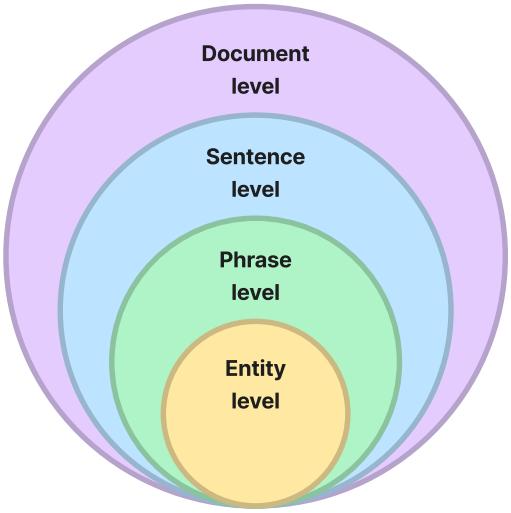



**Figure 1.1** Levels of sentiment analysis (inspired by Wankhade et al., 2022).

#### Document-level

Document-level sentiment analysis is the most straight level. The task is to determine the overall emotional context of the entire document, such as a chapter, article, or review, whether or not involving a study of entities or aspects. This level gives us a general assessment of whether the content is more likely to be positive, negative, or neutral.

#### Sentence-level

Sentiment analysis at the sentence level focuses on individual sentences within the text. We observe the polarity of each sentence autonomously, employing the same methodologies utilised at the document level but with an increased volume of training data and enhanced processing resources. This level is more challenging than the document level because it requires a more in-depth understanding of the text.

[1]Entities are sometimes referred to as targets, hence Target-level or Target-based sentiment analysis.

### Phrase-level

Phrase-level sentiment analysis examines sentiment within smaller linguistic units such as phrases or sentence members. Thus, it can better reveal the emotional charge in specific parts of sentences. Additionally, this level is more challenging than the sentence level because it requires a more detailed understanding of the text.

### Entity-level

The most elaborative level is entity-level sentiment analysis, where we study sentiment associated with specific entities mentioned in the text. This level provides a detailed look at the expressed polarity of certain products, individuals, or organisations. One of the main tasks in this scope is the named entity recognition, which will be discussed later.

Some researchers classify the last level as the aspect-level, as noted by Wankhade et al., 2022, or a more detailed entity-level version called the feature-level proposed by Mary et al., 2017. While both approaches aim to evaluate sentiment towards specific aspects, they differ in their task approach. Relationships between these levels are illustrated in Figure 1.2.

In the first case, aspects are considered without directly mentioning entities in the text. We are not interested in the entities since the input textual data are commonly associated with them[2], such as reviews. The study conducted by Wang et al., 2019 analysed sentiment at the aspect level within restaurant reviews. It primarily examines aspects such as food, price, service, and others. In the feature-based approach, aspects are commonly associated with an entity's features by connecting the entity and its aspects in text. To illustrate, consider the sentence:

*"The battery life of this phone is excellent, but the camera is not good."*

At the feature level, we identify *the battery life* and *camera* as specific features of entity *the phone*, allowing us to determine the polarity of each entity's feature.

The term entity-level sentiment analysis is frequently employed in literature, and some studies consider it synonymous with targeted sentiment analysis, as discussed Rønningstad et al., 2022 in the terminology review. For our purposes, entity-level sentiment analysis better captures the aggregate, document-wide approach, where a single entity can be associated with multiple targets in different sentences, discerning it from traditional target-level sentiment analysis.

[2]Entities are not handled in this case, but we provide them here for a better understanding.

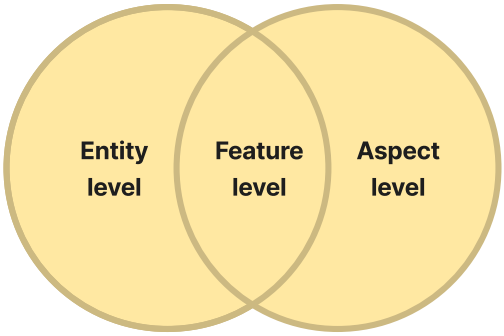



**Figure 1.2** Comprehensive overview of the last level.

However, this thesis primarily focuses on entity-level sentiment analysis, excluding consideration of the entity's features. This decision is motivated by treating the mentioned companies in news articles as entities rather than delving into their specific aspects. Additionally, entity and aspect extraction as separate tasks are complex and challenging, given that the methods employed for recognition differ due to their distinct characteristics (Liu, 2015; Zhang et al., 2014).

## 1.2 Named Entity Recognition

Named Entity Recognition (NER), or entity extraction, constitutes a fundamental component in NLP dedicated to identifying and classifying proper nouns into predefined semantic classes. These classes are unlimited since entities could be anything we can categorise using a tag, including the names of organisations, people, places, or other available information from unstructured textual data such as time, quantity, and money expressions. Someone well-versed in data analysis must have contemplated the possibility that some sources provide data in which the names of organisations are directly associated with their unique tags in text, but this is not our case (for more details, see Chapter 3). Understanding the importance of this task, we recognise its essential role in entity-level sentiment analysis, as it allows us to identify the emotion associated with specific entities mentioned in the text.

The example above illustrates that a single entity can contain more than one word. This challenge is addressed by token tagging formats outlined in the paper (Ramshaw et al., 1995). Individual words are referred to as tokens. The formats describe the position of each token in a named entity, such as the Beginning-Inside-Outside (BIO) format, also known as the Inside-Outside-Beginning (IOB) format

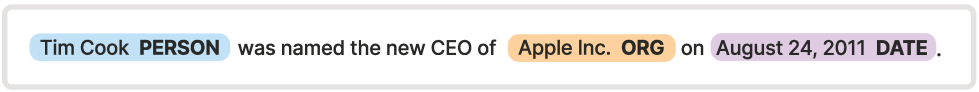



**Figure 1.3** Named entities in the text, along with their associated label classes.

as well as other derived names like Inside-Outside (IO) and Beginning-Middle-End-Whole-Outside (BMEWO). These formats discharge us from the constraints of entities, which are hard-coded or unreliably specified by regular expressions.

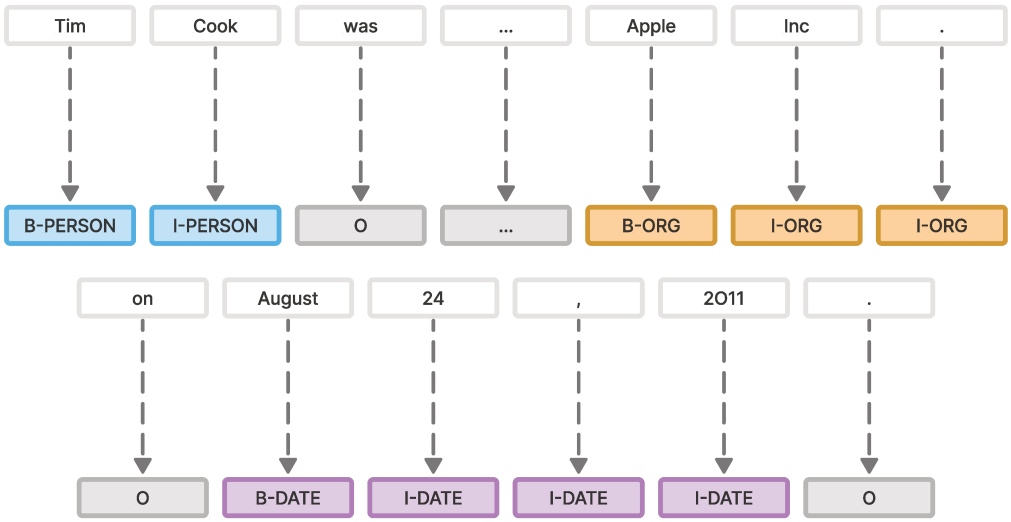



**Figure 1.4** Token tagging BIO2 format within the named entities.

The tags that assign tokens to each entity class contain prefixes. The prefix "*I-*" indicates that the token is contained in the named entity, whereas "*O-*" indicates that the token is not contained in any named entity. The prefix "*B-*" is slightly more specific and indicates that the token is contained at the beginning of the named entity, followed immediately by a token not containing the "*O-*" prefix. A particular case of the BIO approach is the BIO2 format, denoting all tokens beginning with the prefix "*B-*", regardless of whether a token with an "*O-*" prefix follows. A slightly more detailed approach is BMEWO, where the prefixes "*B-*" and "*O-*", as in the previous ones, indicate the beginning and absence of the named entity occurrence, respectively. "*M-*" prefix symbolises the middle token between "*B-*" and "*E-*", where the token with the prefix "*E-*" ends the named entity. Furthermore, the prefix "*W-*" indicates a single-token named entity. Table 1.1 below demonstrates how the mentioned sentence could be labeled with IO, BIO, BIO2, and BMEWO formats.

| | Format | | | |
|---|---|---|---|---|
| Token | IO | BIO | BIO2 | BMEWO |
| Tim | I-PERSON | I-PERSON | B-PERSON | B-PERSON |
| Cook | I-PERSON | I-PERSON | I-PERSON | E-PERSON |
| was | O | O | O | O |
| … | O | O | O | O |
| Apple | I-ORG | I-ORG | B-ORG | B-ORG |
| Inc | I-ORG | I-ORG | I-ORG | M-ORG |
| . | I-ORG | I-ORG | I-ORG | E-ORG |
| on | O | O | O | O |
| August | I-DATE | I-DATE | B-DATE | B-DATE |
| 24 | I-DATE | I-DATE | I-DATE | M-DATE |
| , | I-DATE | I-DATE | I-DATE | M-DATE |
| 2011 | I-DATE | I-DATE | I-DATE | E-DATE |
| . | O | O | O | O |

**Table 1.1** Token tagging formats IO, BIO, BIO2, BMEWO within the named entities.

### 1.2.1 Techniques

Approaches for entity extraction from unstructured data encompass a broad range of techniques, though they commonly converge into three principal categories. Specifically, the work (Keraghel et al., 2024) classifies them in the following way: Knowledge-based, Feature engineering, and Deep learning, as illustrated in Figure 1.5 below.

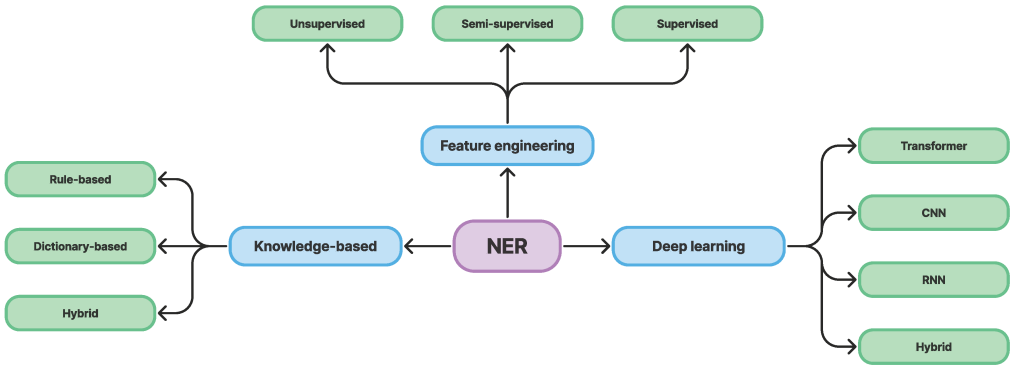



**Figure 1.5** Named entity recognition techniques overview.

### Knowledge-based

Knowledge-based approaches rely on predefined rules and dictionaries to identify entities. These rules and dictionaries, typically created by domain experts, recognise entities based on their characteristics. Rules can be given as regular expressions, denoting patterns to match character combinations in strings. Established on the chosen method, we separate this category into Rule-based and Dictionary-based or combine these approaches. The main advantage of knowledge-based approaches stems from their interpretability, as the rules and dictionaries can be easily understood and modified. Nevertheless, the principal disadvantage is frequent limitations to the entities in the manually created dictionaries and rules, as a result of which new entities cannot be recognised.

### Feature engineering

Instead of manually creating a set of rules and a dictionary, feature engineering-based approaches, popularly identified as machine learning, use linguistic and statistical features to identify entities. These features are generally derived from the text and subsequently are used to train a machine learning model to recognise entities. The primary advantage of this approach is the ability to learn more about the data and discover patterns that may not be apparent at first. Additionally, their ability to recognise new entities differs from dictionaries in some cases. However, the main disadvantage is that these approaches require a large amount of labeled training data to be accurate.

Before will continue in describing another methods, we should clarify the data under discussion and the definition of labels. In the early introduction regarding named entity recognition, we referred to tags. These tags correspond to labels that serve as identifiers describing particular data that are consequently categorised according to the assigned label. To promote better comprehension, referring to the example illustrated in Figure 1.3, we have textual data in which labels are assigned as outlined in the following Table 1.2.

| Data | Label |
|---|---|
| Tim Cook | **PERSON** |
| Apple Inc. | **ORG**ANISATION |
| June 8th, 2023 | **DATE** |

**Table 1.2** Textual data with according labels.

**Unsupervised learning** The unsupervised learning method discovers patterns of entity occurrence in raw and unlabeled data (see Figure 1.6). Consequently, the individual entities are split into groups based on their characteristic properties. The absence of pre-labeled data by human intervention in the training phase causes no supervisor to guide the model with information about the labels in training data, hence unsupervised learning. A conventional method employed in this approach is clustering, such as K-means clustering (Sinaga et al., 2020), which divides data[3] into groups based on similarity or dissimilarity.

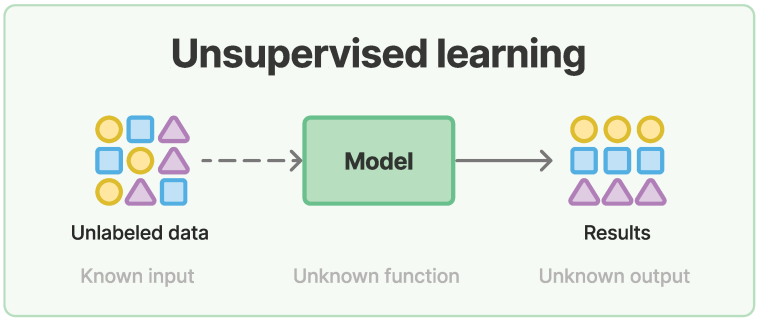



**Figure 1.6** Unsupervised learning approach.

**Semi-supervised learning** The semi-supervised learning method combines labeled and unlabeled data, with the former comprising a slight portion of the dataset (see Figure 1.7). As part of the classification training process, the unlabeled data learn the model's ability to generalise and represent the data in space using a statistical feature that better separates the classes. The algorithm aims to create the best decision boundary between classes based on a large amount of unlabeled data. Besides, the labeled data allows the model to determine the classification correctness for improvement, hence semi-supervised learning. Therefore, the model is partially supervised, hence semi-supervised learning. Semi-supervised learning is a preferred approach for model development because the labeled data is mainly expensive and time-consuming to acquire by requiring human intervention. In contrast, unlabeled data is more easily collectable.

[3]In our case, the data corresponds to tokens symbolising words.

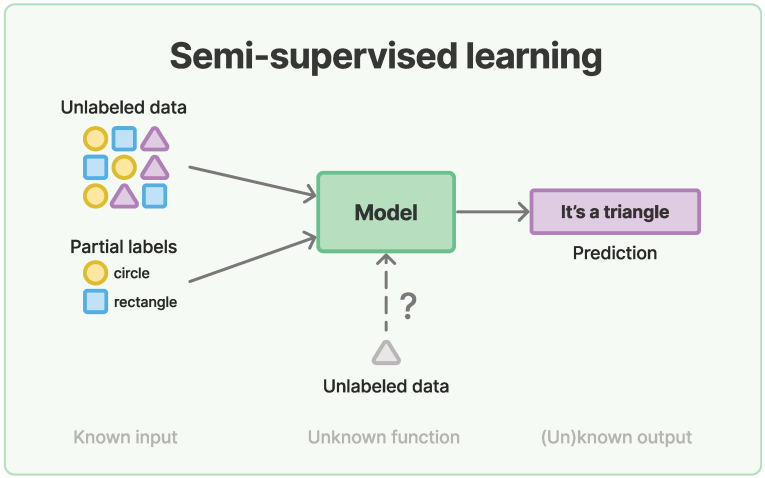



**Figure 1.7** Semi-supervised learning approach.

Self-training, Co-training, and Multi-view learning are frequently employed subcategories of this learning method. Self-training involves using a small amount of labeled data and a more significant amount of unlabeled data, repeatedly training the model on labeled data and using it to predict labels for unlabeled data. Co-training uses multiple independent models that communicate and share information to improve performance. Multi-view learning combines information from different sources or representations of data, such as combining textual data with metadata.

**Supervised learning** Unlike the semi-supervised learning method, the supervised learning approach exclusively utilises labeled data. Therefore, the training process is performed just on labeled data (see Figure 1.8). We could perceive this approach as a function $f : X \mapsto Y$, denoted as $f$, mapping input $X$ to output $Y$, where $X$ and $Y$ represent the input and output, respectively, known from the labeled data. Thus, as a supervisor guides the learning process, the model learns the most suitable way to map the input $X$ to the input $Y$. The principal contrast between fully and semi-supervised learning is that in the former case, learning is done only over labeled data and in the latter case on a combination with unlabeled data. The most common algorithms utilised in supervised learning for named entity recognition include statistical models like Support Vector Machine (SVM) (Wang, 2005), Conditional Random Field (CRF) (Sutton et al., 2012), Maximum Entropy (ME) (Berger et al., 1996), and Hidden Markov Model (HMM) (Eddy, 1996).

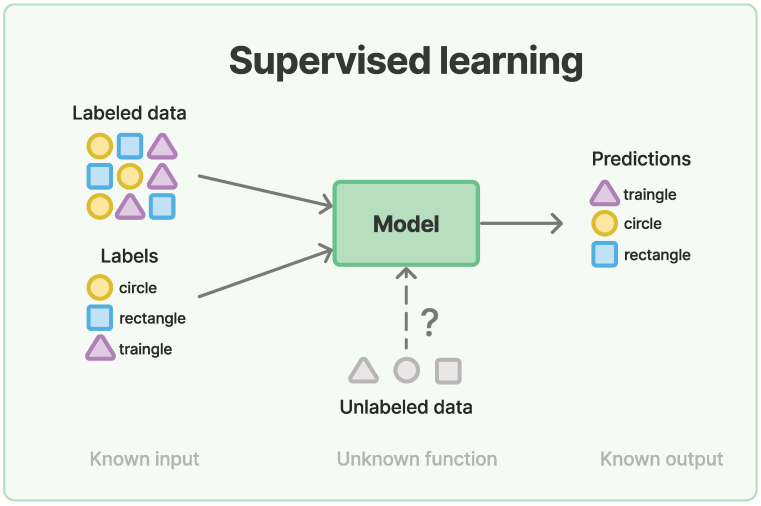



**Figure 1.8** Supervised learning approach.

**Deep Learning** A given text could be extensive and contain many sentences or paragraphs mentioning entities in various forms. Therefore, deep learning comes into play due to its prowess in achieving better comprehension and learning contextual semantics. Its approaches overcome the primarily classification-focused methods introduced thus far, leading to state-of-the-art results in entity recognition. To enhance elucidation of the semantic context, methods' consideration of words occurring before and after words possess a role. The overarching structure of this approach's most commonly employed methods, such as Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), and Transformer, is usually outlined into three fundamental stages: Data representation typically based on One-Hot Encoding, Word2Vec (Mikolov et al., 2013), Term Frequency-Inverse Document Frequency (TF-IDF) (Aizawa, 2003), FastText (Joulin et al., 2016), Global Vectors for Word Representation (GloVe) (Pennington et al., 2014), or Embeddings from Language Model (ELMo) (Peters et al., 2018)Context encoding and Entity decoding.

# 2. Related work

While examining online applications with similar characteristics, a common challenge arises from the need for more transparency regarding the techniques applied in sentiment analysis. Specifically, employed methods and the details of the models remain undisclosed. We acknowledge this limitation due to the proprietary nature of the software, mainly as it is an essential part of the business model. Hence, this chapter is organized into two primary sections.

The initial part provides an overview of existing applications. At the same time, the second section will concern the most recent and relevant research on the association of stock market behaviour and news sentiment analysis.

## 2.1 Existing Application Overview

Sentiment analysis applications accessible to the public are typically based on investigating social network posts, with StockTwist[1] as a notable illustration. According to (Singh, 2022), StockTwits in 2022 boasts more than six million registered users and one million monthly active users, underscoring its prominent user base. With the growing volume of social media contributions, it is difficult to determine which post will prompt action. In our case, these posts do not constitute highly relevant data since our interest is in news articles containing a more significant amount of information. Even if the selection of accounts is limited to informative sources, News organizations' social media accounts only link to their articles, usually accompanied by a headline or lead paragraph. These posts are not enough for our purposes, as we need the entire article to perform better context for entity-level sentiment analysis (discussed further in Chapter 3).

### 2.1.1 Bloomberg Terminal

The only software similar to the one created is a module in the so-called Bloomberg Terminal[2] from Bloomberg L.P.. This software system provides investors with analytical tools over financial data, including sentiment analysis of news articles and posts on social network X, formerly known as Twitter. The cost of the Bloomberg Terminal depends on the required specific features and services. A standard subscription typically amounts to approximately $\$24,000$ per year. It

[1] *https://www.stocktwits.com*

[2] *https://www.bloomberg.com/professional/solution/bloomberg-terminal/*

is a very complex and powerful software, but it is not accessible to the general public.

Bloomberg, in its work (Cui et al., 2024), describes two types of sentiment analysis: story-level and company-level sentiment, utilizing a suite of Supervised Machine Learning (SML) techniques. Classification engines are trained on labeled datasets containing news articles and social media posts. Reportedly, the labeling process is based on the question:[3]

> *"If an investor having a long position in the security mentioned were to read this news or tweet, is he/she bullish, bearish or neutral on his/her holdings?"*

Once model training is completed, the models are employed to analyze recently published posts and articles associated with organizations, seeking distinctive sentiment signals related to the business and finance domain. As mentioned above, sentiment is divided into two levels:

**Story-level** Sentiment score value of articles and posts is calculated after arrival in real time. The calculation includes score and confidence, where the score has one of three options: positive, negative, or neutral, each described by a numerical value from the set $\{-1, 0, 1\}$. The confidence is defined by a value ranging from $0$ to $1$, demonstrating the intensity[4] of the sentiment. Hence, the story-level sentiment ranges from $-1$ to $1$. For both, we get the following equation:

$$\textit{Story-level}_c^{\textit{Articles}} = S_c^a C_c^a, \quad a \in P(c) \tag{2.1}$$

$$\textit{Story-level}_c^{\textit{Posts}} = S_c^p C_c^p, \quad p \in P(c) \tag{2.2}$$

where $a$ represents an article and $p$ represents a post from $P(c)$, the set of published articles and posts referencing company $c$. $S_c^a$ and $S_c^p$ are the sentiment polarity scores of article $a$ and post $p$ that reference company $c$. $C_c^a$ and $C_c^p$ are the confidences of article $a$ and post $p$ that reference company $c$.

**Company-level** Sentiment score value is then calculated as the confidence-weighted average of the story-level sentiment scores, incorporating all relevant news articles and social media posts mentioning the company.

$$\textit{Company-level}_{c,t}^{Articles} = \frac{\sum_{a \in P(c,T)} S_c^a C_c^a}{N_{c,T}^a}, \quad T \in [t_b, t] \tag{2.3}$$

[3]This information is difficult to verify due to the unavailability of the dataset.

[4]Probability of being positive, negative, or neutral.

$$\textit{Company-level}_{c,t}^{Posts} = \frac{\sum_{p \in P(c,T)} S_c^p C_c^p}{N_{c,T}^p}, \quad T \in [t_b, t] \tag{2.4}$$

where $a$ represents an article and $p$ represents a post from $P(c,T)$, the set of published articles and posts referencing company $c$ during period $T$. Period $T$ is a time interval of length $t_b$ to $t$, where $t_b$ is the time constant of the beginning. $N_{c,T}^a$ and $N_{c,T}^p$ are the number of articles and posts referencing company $c$ during period $T$. In this way, the company-level sentiment is calculated as follows:

$$\textit{Company-level}_{c,t} = \textit{Company-level}_{c,t}^{Articles} + \textit{Company-level}_{c,t}^{Posts} \tag{2.5}$$

Intraday Company-level sentiment score for news articles is recalculated every two minutes, utilizing an eight-hour rolling window. The sentiment score for social network posts is recalculated every minute, employing a 30-minute rolling window. Due to the previous definitions, we can express these by a simple substitution of $t_b$ depending on the rolling window as follows:

$$\textit{Intraday Company-level}_{c,t}^{Articles} = \frac{\sum_{a \in P(c,T)} S_c^a C_c^a}{N_{c,T}^a}, \quad T \in [t-8, t] \tag{2.6}$$

$$\textit{Intraday Company-level}_{c,t}^{Posts} = \frac{\sum_{p \in P(c,T)} S_c^p C_c^p}{N_{c,T}^p}, \quad T \in [t-0.5, t] \tag{2.7}$$

$$\textit{Intraday Company-level}_{c,t} = \textit{Company-level}_{c,t}^{Articles} + \textit{Company-level}_{c,t}^{Posts} \tag{2.8}$$

Daily company-level sentiment scores are published every morning about 10 minutes before the market opens. The calculation is determined as a confidence-weighted average of sentiment scores derived from the story-level sentiments of news and social media posts over the past 24 hours as follows:

$$\textit{Daily Company-level}_{c,t} = \frac{\sum_{d \in P(c,T)} S_c^d C_c^d}{N_{c,T}}, \quad T \in [t-24, t] \tag{2.9}$$

where document $d$ represents a news article or social media post, the sentiment polarity score of document $d$ referencing company $c$ is denoted as $S_c^d$, and the confidence associated with this reference is represented by $C_c^d$. The set $P(c,T)$ encompasses non-neutral documents referencing company $c$ published within the last 24 hours. $N_{c,T}$ expresses the count of non-neutral documents referencing company $c$ during period $T$. This approach is further explored in terms of the informational role of social media by Gu et al. (2020).

## 2.2 Entity-level Sentiment Analysis

This section will discuss the most recent and relevant research conducted in entity-level sentiment analysis and its application in news articles, including named entity recognition over news articles.

The initial prominent model is FinBERT (Araci, 2019), which is based on the Bidirectional Encoder Representations from Transformers (BERT) model. FinBERT is specifically designed to analyse sentiment in the financial domain, outperforming state-of-the-art classification tasks by $15\%$ in accuracy while already outperforming them with significantly less (one-sixth) training data. This is admirable, given that deep learning techniques typically require large amounts of data for training. The authors also point out that FinBERT is outstanding at capturing explicit expressions of feelings. However, modelling implicit information is still an open problem, especially when these implications are not clear even to the author of the text. Testing entity-level sentiment analysis in financial news, the work (Tang et al., 2023) presents results that fine-tuned FinBERT, as a pre-trained language model, namely FinBERT-CRF, can achieve better results than other models such as the large language model ChatGPT-$3.5$ turbo.

Another model within the BERT family is explored in the work (Zhao et al., 2021), which employed RoBERTa, a Robustly Optimised BERT Pretraining Approach (Liu et al., 2019b), to propose a sentiment analysis and entity detection strategy in financial text on social media. In the first step, sentiment analysis, mainly focusing on negative polarity, is performed. Then, entity detection is considered in different granularities with Machine Reading Comprehension (MRC) (Liu et al., 2019a), or sentence-matching tasks. As a result, this study serves entity detection differently than traditional named entity recognition. The authors claim that the proposed method outperforms standard sentiment analysis and entity detection methods. The authors also emphasise the importance of the financial domain, where the sentiment of an entity can significantly affect the stock price.

Moreover, it is essential to consider the FinABSA (Son et al., 2023) based on the Text-to-Text Transfer Transformer (T5) Large (Raffel et al., 2020) model. Even though the model is aspect-based, it can be regarded as a comprehensive model analysing sentiment at the entity level because aspects can be considered entities. We show this in more detail in Chapter 5, where we use this model to construct our algorithm. The evaluation results achieve $87\%$ accuracy using the test split arbitrarily from SEntFiN 1.0 (Sinha et al., 2022). The author of FinABSA pointed out[5] that due to the structural limitations of the task, the conventional FinBERT model, which does not consider entities, can not correctly classify sentences with conflicting opinions regarding two distinct entities, which is very important for

[5]https://www.github.com/guijinSON/FinABSA

our analysis. Therefore, this thesis will focus primarily on this model.

## 2.3 Stock Market Behaviour

Several approaches for predicting stock market behaviour and price trends have been proposed, utilizing sentiment analysis of financial news and historical stock prices. Several studies prove a strong correlation between financial news sentiment and stock prices (Li et al., 2014) (Wan et al., 2021). Due to the nature of unstructured textual data, predicting stock market behaviour is a challenging task.

Khedr et al. (2017) focuses on creating an effective model for forecasting future trends in the stock market, using sentiment analysis of financial news and historical stock prices. The model achieves more accurate results than previous works by considering different market and company news types combined with historical stock prices. The experiments utilize datasets from three companies: Yahoo Inc., Facebook Inc.[6], and Microsoft Corporation. The authors use well-known and informative news sources such as Reuters, The Wall Street Journal, and Nasdaq. The first step of sentiment analysis to get the text polarity using a Naive Bayes classifier was shown to achieve accuracy from $72.73\%$ to $86.21\%$, while the second step, which combines news sentiment with historical prices, improved prediction accuracy up to $89.80\%$. Moreover, that is why we find this study motivating and inspiring.

[6]Known as Meta since 2021.

# 3. Textual Data

Integrating data, particularly newspaper article content is a fundamental component of our application. To ensure a sufficient implementation, we must consider several aspects. The following chapter will discuss these aspects from different perspectives, including the developer's viewpoint and legislative considerations.

Using full text maximises the potential and value of the data, but this complicates the development of an application based on historical and current data. The chapter further states that static datasets are unsuitable for our application and emphasises the need to select a source that provides both historical and current data. The chapter also focuses on terms of service because of recent legal disputes between the New York Times, Microsoft and OpenAI, highlighting the importance of adhering to these rules when using data resources.

Various data extraction techniques, such as RSS feeds, web scraping, and API, are discussed from the perspective of using them for the development and reliability of our application. Each method has its advantages and disadvantages, which are further elaborated. The chapter mainly focuses on APIs, preferred when provided directly by data sources. They provide good quality and reliable data, although third parties may offer alternative solutions with some limitations, which are discussed. The experiments performed in this chapter are available in the attached Python notebook[1].

At the end of the chapter, we decided that the Guardian was the most appropriate source and, because obtaining other data was unattainable, our only source. We have also decided that the timeframe we have set is three months, and due to the data's lifecycle, we have to remove all data from the database every 24 hours at most.

## 3.1 Introduction

In the context of entity-level sentiment analysis, it is essential to retrieve the entire textual content of each article. The internet holds vast amounts of information, making it more reasonable to obtain the content of an article rather than just its headline. The full text contains all the entities concerned, whereas the headline may not include all mentioned entities. Consequently, this allows us to maximise the potential and value of data such as news articles. This requirement complicates the development process from the beginning. Building an application on a dataset

[1]Located in the directory /ipynbs/textual-data/.

from the past would be inefficient as it would not include current news coverage, making it useless to the user. Therefore, we address this data retrieval process in the following sections. We aim to find a source that provides both historical and current data. Therefore, we will not consider static datasets as a data source for the deployed application. When selecting a data source for news article content, it is essential to consider the following aspects:

**Reliability** Expresses the degree to which data sources can be trusted based on their history and reputation.

**Availability** Refers to the availability of the news articles for our application and factors such as the cost of the data.

**Accessibility** Refers to the ease with which data source can be accessed. Consider data retrieval methods and any restrictions on accessing news articles.

**Consistency** Stands for the data source that maintains a consistent format and structure, facilitating easier integration into our application.

**Terms of Service** Considers the terms of service of the data source, which define the rules and regulations for using the data source.

Before we get to these and delve into specific extracting techniques and data sources, where we will elaborate on these aspects, we are devoting a separate section to the terms of service aspect due to recent events.

## 3.2 Terms of Services

When integrating data such as news articles into an application, it is essential to consider the data source's terms of service. The terms of service define the rules and regulations we must follow when using the data. These terms include restrictions on use and lifecycle. Complying with the terms of service is crucial because violating these rules may lead to blocked access to data or lawful problems. The impetus for including this section is a recent dispute initiated by the New York Times, which filed a lawsuit against Microsoft and OpenAI (Stempel, 2023; Bergen, 2023).

Although our application does not use generative artificial intelligence, which is the cause of this dispute, the data use terms for most providers cover machine learning models in their entirety. These terms are often mentioned in the context of data used for training models. Data providers are concerned that their data is being used to train models that may lead off readers who could be subscribers to their paid content or miss out on monetary gains from advertising. The New York

Times was concerned that chatbots like ChatGPT and Copilot from the companies mentioned above used direct excerpts from their articles in the responses from their generative models.

The dispute initiated by the New York Times was directed towards regulating the use of their data to train models and delineating the damages that were incurred due to the breach of the agreement between the companies and the New York Times. This litigation certainly has implications for the future use of data from other newspaper resources, not just Microsoft and OpenAI. Therefore, it is essential to pay attention to this matter. The New York Times was the first major US media organisation to raise the issue. In April of this year, eight newspapers owned by Alden Global Capital made further allegations against Microsoft and OpenAI (Robertson, 2024; Guardian, 2024a).

With this section, we want to emphasise that we can not arbitrarily use data such as articles. They are subject to copyright, and most companies make legitimate attempts to protect this data from unauthorised use, which we cannot ignore.

## 3.3 Data Extraction Techniques

Different approaches to data extraction exist. In our case, we focus on the possibility of extracting textual data as whole content from newspaper articles. This section looks at several basic methods for extracting text data from news articles, including web scraping, Really Simple Syndication (RSS) feeds (Marc, 2024), and Application Programming Interface (API). We intentionally omit static datasets in the following subsections because they do not provide recent information.

### 3.3.1 RSS Feeds

One of the oldest approaches for obtaining data is Really Simple Syndication, also known as RSS. It is a standard used to provide updated content on web pages. These feeds are typically used to provide breaking news, blogs, and other information on the internet. It uses XML format and typically contains headlines, descriptions, and link information to the article. RSS feeds are, in most cases, provided only by section and include 20 to 100 new articles over a while, typically a day to a week. This is insufficient as we need control over the actual specifications for this period.

The feeds with direct article content are impossible to find in official and well-known sources such as the New York Times, Bloomberg, Reuters, and more. This choice of data would be ideal if we were primarily looking for updates to a data source without the possibility of obtaining historical data within the

specified time range. In that case, consider the successor to RSS, namely the Atom Syndication Format (Sajid, 2023) standard for feeds. However, it still needs to address the necessity for historical data, and we have also been searching for an official channel that provides the entire content but unsuccessful. Therefore, we omit RSS feeds when evaluating individual sources.

### 3.3.2 Web Scraping

Another option is web scraping, which involves extracting data from web pages. This approach is subject to limitations defined by the Robots Exclusion Protocol (REP) in a text file called *robots.txt*. This file is typically located in a website's root directory and contains rules for programs, such as robots. It determines whether the page can be crawled, a standard process associated with web scraping.

Given the number of requests needed to obtain the full content of several articles, web scraping is unsuitable for our application. We would have to crawl all the pages of all articles and extract all the text content from them. One disadvantage of this approach is the potential blocking of the IP address, which could result in being denied access to the data, as it is not an official method for accessing the data that websites require. This is one of the reasons why websites try to regulate crawling using *robots.txt* for ethical crawling and scraping. However, it is essential to realise that REP does not grant us access to the data. It simply specifies the rules we should follow when crawling the pages.

Another disadvantage is the data quality and reliability we can obtain this way. The data's consistency and structure are also problematic because each website has a different structure. We would have to create a specific web scraper for each source, which is challenging and inefficient. Compared to other obtaining data techniques, such as API, creating a scraper is much less efficient and demanding in terms of development, especially in terms of maintenance. However, we would use this approach, for example, if we wanted to source from a blog or news source that does not provide an official API. In addition, we use this approach rather than RSS feeds because it allows us to access both historical and current data.

### 3.3.3 API

Application Programming Interface, also known as API, is a collection of defined rules and protocols allowing software applications to communicate. It serves an interface that facilitates communication between the data source and our application by querying existing endpoints. APIs are complemented by documentation that details how to use the API, which endpoints are available, what parameters can be used, and what limitations apply. API providers typically offer plans that determine how many queries we can complete in a given period or impose other

limitations. We distinguish between APIs provided directly by the data source and those provided by third parties.

In the first case, we might use APIs from providers such as The New York Times[2], Bloomberg[3], Reuters[4], and others. APIs provided directly by the data source are usually the best choice because they provide direct access to the data and naturally have the highest quality. These APIs typically offer endpoints that allow us to retrieve entire articles, though it remains a question whether they are available in a paid plan or for free.

Third-party APIs can be helpful if we do not have access to an API provided directly by the data source or want to combine data from various sources with less effort. However, third-party APIs may have limitations that can affect data quality, such as incomplete or inconsistent data, since they provide data from many sources[5], which can lead to vaguely defined specifications and data inconsistency, for example as information about what the articles sections contain.

Due to the variability of APIs from specific providers, we will divide our discussion into two separate sections. The first section will focus on third-party data providers, and the second will delve into first-party data providers.

## 3.4 Third-party Data Providers

Third-party APIs provide article data aggregated from various sources on the internet. Since they are available in different pricing plans and we have not noted any exceptions for research purposes, we will focus only on free plans. Thus, we are subject to the most common limitation, the maximum number of queries within a specified period. Most providers offer free plans, primarily allowing developers to test various endpoints and verify if they meet the application's requirements. The number of allowed queries with this type of provider is meagre, typically ranging from 10 to 20, with not many articles in response. The data also contains limitations with a delay of 12 - 24 hours, as seen with NewsData[6].

In our application, we would likely be able to limit the number of queries to still use the free plan without losing access to the required endpoints. Of course, we could use APIs providing sentiment analysis, such as Alpha Vantage[7]. However, none of these providers provide information about the models and calculations they use, which we do not consider suitable given the nature of this

[2]https://www.developer.nytimes.com/apis
[3]https://www.bloomberg.com/professional/products/data
[4]https://www.liaison.reuters.com
[5]They may include sources offering their APIs as primary providers.
[6]https://www.newsdata.io/
[7]https://www.alphavantage.co

thesis. We have attempted to contact several providers but have not received a response.

During our examination of various APIs, we consistently encountered a common issue: incomplete or misunderstood service specifications. To illustrate this concern, we present an example within the context of news article APIs. Although Finnhub[8] does not provide full-text content article data, we also tried an endpoint that includes purely articles within the given company during testing. While Finnhub does not impose strict limits on query volume, its website indicates a maximum of 60 API calls per minute under the free plan.

Finnhub's documentation (Finnhub, n.d.) claims to offer "*1 year and real-time updates*" within its fundamental data services. Its company news endpoint allows developers to specify parameters such as *from* and *to* for retrieving data within specific periods. Developers like us may request data from the past month, which could be, from a particular perspective, qualified as the latest data. However, how many articles or days of data are included within this timeframe remains unclear, highlighting a need for more transparent specifications and validation. We conducted tests using the Finnhub API for the last two quarters of 2023 and the first quarter of 2024, focusing on companies such as Apple Inc. (AAPL), Microsoft Corp. (MSFT), Alphabet Inc. (GOOGL), and Amazon.com Inc. (AMZN). The volume of articles for individual quarters, distributed across individual days, is shown in Figure 3.1 and those attached in Appendix A.1.

We have not even identified a consistent pattern to guide the data, so it is unclear what exactly the latest fundamental data means, even though we can set the range parameters *from* and *to* for the past year. In addition, Figure 3.1 illustrates an article at Amazon.com on January 11th, early in the first quarter of 2024, highlighting the data's inconsistency. It is not our goal to cast a pall over Finnhub but only to point out a shortcoming that is also present in other third-party providers within the specification.

Analysing each third-party resource is a time-consuming process that complicates the development of our proposed application due to misleading API specifications. Moreover, we are uncertain if all third-party providers have proper permissions to acquire data by the terms of service of the original data providers. With data from numerous newspapers, determining liability in potential legal issues remains ambiguous. Additionally, the extent to which we should adhere strictly to the terms established by third parties or direct data sources remains unclear.

[8] https://www.finnhub.io

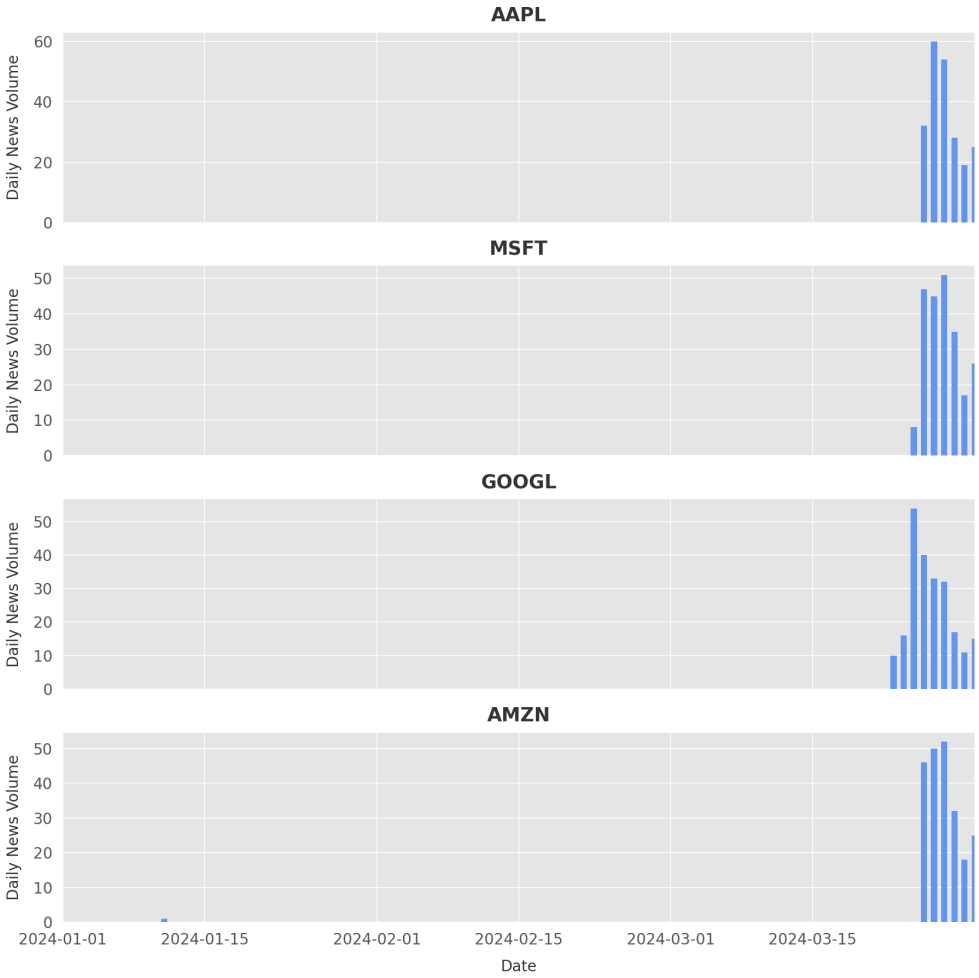



**Figure 3.1** Finnhub daily news articles volume of Apple Inc. (AAPL), Microsoft Corp. (MSFT), Alphabet Inc. (GOOGL), and Amazon.com Inc. (AMZN) for the first quarter of 2024

## 3.5 First-party Data Providers

First-party data providers are the ideal choice. Typically, the API specifications of these providers differ in minor ways, but they can provide quality and reliable results. We have reviewed many sources that might be suitable for our application. Reuters and Bloomberg do not provide a free plan allowing us to extract entire articles. The New York Times provides a free API, but it does not have the ability to extract entire articles. We must request a modified data solution to extract this data, which is almost impossible for free. We further attempted to obtain data from the Financial Times. However, they were unable at the time of our investigation to handle the volume of requests they receive from the students and faculty for accessing their API. Ultimately, the only one who provided us access to its data is the Guardian through its platform, the Guardian Open Platform[9].

The Guardian is a British daily newspaper that covers news across the world. Through its official Open Platform, the Guardian provides a free API that allows us to extract articles, including entire textual content. After describing our research and thesis objectives to the Guardian, we accessed its complete API, allowing us to retrieve full-text articles with a daily limit of $500$ API calls. This is sufficient for us as it provides an endpoint for searching by section, where the default result of the number of articles per response page is $10$. However, this parameter can be changed to $200$, whereby we can retrieve $200$ articles per query. An additional benefit is the capability to retrieve data for the current day.

Our study will focus on a three-month timeframe, explicitly exploring the business and technology sections. However, following their condition, which is that the data lifecycle must be less than $24$ hours, is essential. This means we must remove the data from the database every $24$ hours. For these reasons and other capacity constraints, we chose a time frame of three months to process the data, and we see this as sufficient time for our needs.

To compare the daily volume of news articles with third-party data providers, we conducted a comparative experiment using data from the Guardian as a first-party data provider. Unlike Finnhub, which uses direct ticker references, we utilised literal mentions querying of companies. The results from The Guardian are illustrated in Figure 3.2 and those attached in Appendix A.2.

First-party data providers proved preferable, providing comprehensive data quality and reliability as direct sources. This approach enables us to obtain data that accurately reflects reality while ensuring compliance with the terms of service to avoid legal issues. After explaining the thesis's purpose and intentions for data usage, the API key was provided to us. We extend our gratitude to Guardian for providing us with the data for research and implementation in this

[9]https://www.open-platform.theguardian.com

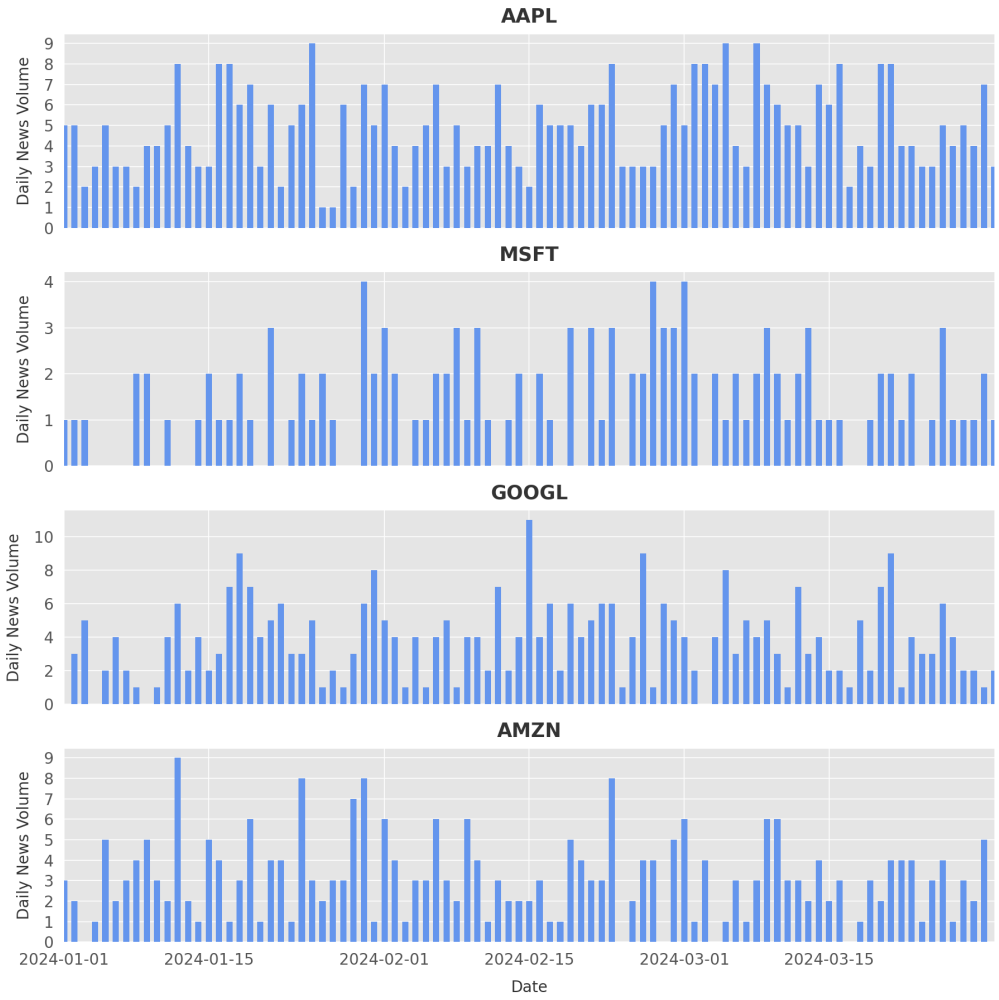



**Figure 3.2** The Guardian daily news articles volume of Apple Inc. (AAPL), Microsoft Corp. (MSFT), Alphabet Inc. (GOOGL), Amazon.com Inc. (AMZN) for the first quarter of 2024

thesis, as procuring such a data source was very challenging. Reflecting on the aspects discussed earlier in this chapter, the Guardian is an ideal choice for our application as a first-party data provider. Its data is of outstanding quality, reliable, and efficiently accessible, and we fully comply with the terms of service.

# 4. Company to Symbol Linking

This chapter delves into the challenge of linking companies mentioned in the text to their corresponding ticker symbols and exchanges. It highlights the limitations of the naive approach that relies on static data and score thresholds. However, it also introduces a more robust approach to entity linking using Wikidata.

The cornerstone of the entity linking approach is the comprehensive Wikidata knowledge base. By leveraging structured data and relationships between entities, the entity linking method can effectively link companies to their ticker symbols, even if their names appear in different forms. Moreover, it eliminates the need for arbitrary score thresholds and provides access to wealthy company information.

The entity linking approach offers several advantages over the naive method, including increased accuracy, better handling of name variations, and access to comprehensive company data. However, it is crucial to recognize potential limitations. Nevertheless, the result shows that the entity linking approach using Wikidata is a powerful tool for linking companies to their ticker symbols, providing a flexible and accurate solution over the naive approach.

The chapter also distinguishes between a company and an organisation. A company certainly refers to an entity that trades on an exchange (has a ticker). In contrast, an organisation refers only to a potential candidate by named entity recognition that might be tradable on an exchange or associated with that company. In addition, the Python notebook[1] is provided for both sections of this chapter to guide the reader through the process by which the results were obtained.

## 4.1 Introduction

Before proceeding further into this chapter, let us briefly revisit the concept of named entity recognition. It is a classification task to recognise entities within a given text, including names of organisations, people, places, dates, and more. While named entity recognition assigns entities to classes based on the syntax and semantics of the text, it does not provide specific details about individual entities. The necessity to identify companies in articles arises from their potential appearance in various forms and references. To give an example, the company Microsoft might be mentioned as "*Microsoft*", "*Microsoft Corp.*", "*Microsoft Corporation*", or "*MSFT*". In cases where multiple syntactic variations referring to the

[1] In the directory /ipynbs/company-to-symbol-linking/.

same company occur in an article, named entity recognition categorizes them all as organizations. However, it is crucial to have additional information to confirm that they represent the same entity. Bringing together these diverse forms[2] into a single entity identifiable by a unique symbol is essential.

Each company has a unique identifier called a ticker, used for stock market identification. This ticker must be assigned to every mention of the company within the article. Additionally, we need to ensure that every mention of a company has a corresponding ticker. The assignment facilitates the extraction of companies operating across various exchanges and helps eliminate unnecessary entities identified by named entity recognition. Not every organisation identified by named entity recognition, such as Greenpeace, the World Health Organization, or the United Nations, necessarily represents a company listed on the exchange. Moreover, this process will play a pivotal role in the subsequent stages of application development.

While some newspaper articles, such as those from Bloomberg, Reuters, and CNBC, may include a company's ticker symbol or another specific identifier immediately following its name in the text, as demonstrated by "*Microsoft (MSFT)*", this practice is generally the exception rather than the norm across all news sources. Our primary textual data source, the Guardian, does not contain information about company tickers in this way (for more details, see Chapter 3). Hence, the task involves associating a company's name with its symbol. This chapter will focus on strategies for addressing this challenge, aiming to identify the optimal approach for handling this issue.

## 4.2 Problem definition

Our objective is to address the challenge of assigning a unique identifier to each company mentioned in the text. Companies identified in the named entity recognition classification typically belong to the organisation class, narrowing our focus to entities categorised as organisations. In general, this challenge can be divided into three essential parts:

**Input:** An article's text.

**Output:** A set of companies with their unique identifiers.

1. Recognising companies mentioned in the article.

[2]Different forms often encountered in newspaper articles due to authors' inconsistencies and text length.

2. Assigning a unique identifier to each identified company, provided one exists.

3. Extracting a set of companies contained in the article.

The initial phase of the concern involves utilising named entity recognition to classify entities, as mentioned above, specifically focusing on those categorised as organisations. The subsequent step entails implementing a method to assign a unique identifier to each company. Hence, the second phase relies on matching company names against a database of companies and their detailed information. The third component is straightforward and involves the unification of all occurrences into a set of companies already assigned identifiers.

To ensure we take advantage of every mention of a company. In the subsequent sections, we will explore various methods that could address the given problem and select the most suitable one for our case. Let us consider the following excerpt from the article (Guardian, 2024b) about deepfake technology published by the Guardian on February 25, 2024:

> *"Executives from Adobe, Amazon, Google, IBM, Meta, Microsoft, OpenAI and TikTok gathered at the Munich Security Conference to announce a new framework for how they will respond to AI-generated deepfakes that deliberately trick voters."*

The named entity recognition identifies Adobe, Amazon, Google, IBM, Microsoft, OpenAI, TikTok and Meta as organisations. It is good to note that TikTok is owned by ByteDance, not a publicly traded company. OpenAI finds itself in a similar position to that of a private company. Thus, buying directly from companies that own TikTok or OpenAI is impossible. However, a different scenario arises when discussing Facebook, Instagram, or Messenger, organisations owned by Meta, a publicly listed company on the exchange. The task is to assign the correct ticker symbol to each company mentioned in the article and to determine the stock exchange on which the company is listed. The following Table 4.1 lists the companies mentioned in the article with their official name, corresponding ticker symbol, and stock exchange.

According to Table 4.1, organisation names are not always identical to their official names. This prevalent discrepancy underscores the necessity to address this issue, which will be thoroughly examined in subsequent sections focusing on methods for linking company symbols. Consequently, our approach will leverage a database containing companies listed on exchanges and their corresponding tickers. This database will enable us to match the official names of companies with the recognised organisation entities extracted from the article. By obtaining the ticker symbol, we can confidently access additional information about the

| Organisation | Official Name | Ticker | Stock Exchange |
|---|---|---|---|
| Adobe | Adobe Inc. | ADBE | NASDAQ |
| Amazon | Amazon.com Inc. | AMZN | NASDAQ |
| Google | Alphabet Inc. | GOOGL | NASDAQ |
| IBM | International Business Machines Corp. | IBM | NYSE |
| Meta | Meta Platforms Inc. | META | NASDAQ |
| Microsoft | Microsoft Corp. | MSFT | NASDAQ |
| OpenAI | N/A | N/A | N/A |
| TikTok | N/A | N/A | N/A |

**Table 4.1** Organisations identified in the Guardian's article about deepfake technology with their official names, ticker symbols, and stock exchanges.

company from other databases, including its stock exchange listing and other pertinent details.

## 4.3 Naive approach

Methods presented in this section are based on matching company names against an exchange static dataset of companies. The dataset contains a list of companies with their official name and ticker symbol. The naive approach involves comparing organisation names extracted from the article with the official names in the dataset. The corresponding ticker symbol and exchange are assigned to the organisation if a match is found. This straightforward approach is a good starting point for linking company names to their ticker symbols. However, it has its limitations, as it may not be able to handle variations in company names.

### 4.3.1 Database

The first step in implementing the naive approach is to obtain the data. We need a dataset containing company official names, ticker symbols, and information about relevant stock exchanges. The dataset should be in a format conducive to efficient processing, encompassing file formats such as CSV, JSON, and XML, or alternatively, be structured within a relational database table. We discovered suitable datasets of companies listed on various exchanges on the EODData website[3], with a particular focus on the NASDAQ, NYSE, and AMEX datasets. The datasets contain information about companies listed on exchanges, including

[3]https://www.eoddata.com

their official names and ticker symbols. Each exchange has its dataset, so we have information on which exchange the company is listed. The datasets are in CSV format and can be easily preprocessed, allowing us to match organisation names extracted from the article.

### 4.3.2 Data preprocessing

It would be beneficial to do some preprocessing before starting the matching process. This involves making a few simple adjustments to make it easier to compare individual strings. We execute these steps on the dataset of each exchange and the organisation names, which we are trying to match accurately. The preprocessing steps encompass:

**Lowercase conversion** Convert all characters to lowercase to ensure case-insensitive matching.

**ASCII conversion** Convert all characters to ASCII to remove any non-ASCII characters in the string.

**Punctuation removal** Remove punctuation to eliminate any special characters that could interfere with the matching process.

**Common suffix removal** Remove common suffixes such as "*Inc.*", "*Corp.*", "*Ltd.*", and similar terms, as their usage may be inconsistent across the article's company name and the dataset's official name.

**Common word removal** Elimination of common words such as "*holding*", "*company*", "*group*", and others, which may not be essential for matching the company name.

During the implementation, we discovered an effective Python library called Name Matching, available on GitHub[4]. This library enables data preprocessing and the use of the distance metrics discussed in the following subsections. Additionally, after data preprocessing, the Cosine similarity method is used to reduce the number of potential matches by converting strings to n-grams and applying a TF-IDF transformation. With the preprocessed and reduced data in hand, we are ready to advance to the matching process based on distance metrics.

[4]https://www.github.com/DeNederlandscheBank/name_matching/

### 4.3.3 Fuzzy matching

Fuzzy matching, or approximate string matching, is a technique used to determine the similarity between two strings using distance metrics. The lower the distance, the more similar the strings are. In contrast to exact matching, which requires a perfect match, fuzzy matching allows working with data that may contain incomplete matches. Therefore, this method is beneficial when dealing with variations in company names. In our case, the fundamental aim of this matching approach is to identify the most similar company name from the dataset for each organisation name extracted from the article.

#### Discounted Levenshtein distance

The leading and most advantageous distance metric, particularly suited to our use case, is the Levenshtein distance (Levenshtein et al., 1966) used to calculate the number of single-character operations such as insertion, substitution, and deletion, needed to transform one string into another. When calculating the distance, we can utilise the ability to weight individual operations differently. Specifically, the discounted variant reduces the cost of adjustments made closer to the end of the string. This characteristic holds significant importance in scenarios involving company names.

To illustrate an example, let us examine the process of matching "*Amazon*" from the Guardian's article and "*Amazon.com Inc*" from the dataset. Following all preprocessing steps, the extracted entities become "*Amazon*" and "*Amazon.com*". In such cases, employing the discounted Levenshtein distance ensures that "*amazoncom*" is not considered significantly outlying from "*amazon*" within the metric system. This approach is crucial because it helps discern that any other company name with differing first three letters can not accurately refer to "*Amazon.com Inc*" aligning with our intended goal. In summary, variations in suffixes are more common for each company name than variations in prefixes.

Using all preprocessing steps and the sample of the Guardian's article in which we have extracted organisation entities, we get the most similar company name from the dataset with a score based on discounted Levenshtein distance for each. An exact match scores $100$, while $0$ signifies no similarity. The results are presented in Table 4.2.

#### Wighted Jaccard similarity

Another possible approach is Weighted Jaccard similarity, in which the distance metric is expressed by a measure used to compare the similarity between two token sets. One set obtains an organisation name in an article, while the other represents an official company name in the dataset. The Name Matcher library

| Organisation | Official Name | Score |
|---|---|---|
| Adobe | Adobe Systems Inc | 100 |
| Amazon | Amazon.com Inc | 74.450 |
| Google | Neos Yield Premium Strategy Google [Googl] ETF | 35.917 |
| IBM | Ibio Inc | 55.647 |
| Meta | Kennametal Inc | 37.796 |
| Microsoft | Microsoft Corp | 100 |
| OpenAI | Open Bank | 73.286 |
| TikTok | Cytokinetics | 27.436 |

**Table 4.2** Extracted organisation names from the Guardian's article and their matches with official names in the dataset using the match quality scores based on the discounted Levensthein distance.

defines the Weighted Jaccard similarity for the article organisation name set $X$, the official company name set $Y$, and a weight $w$ as follows:

$$sim_{Jaccard_w}(X, Y) = \frac{w \cdot |X \cap Y|}{w \cdot |X \cap Y| + |X \setminus Y| + |Y \setminus X|} \tag{4.1}$$

In terms of a two-by-two confusion table, this similarity is expressed as:

$$sim_{Jaccard_w} = \frac{w \cdot a}{w \cdot a + b + c} \tag{4.2}$$

Where the following definitions apply:

- $a = |X \cap Y|$: The number of words common to both sets (true positives).
- $b = |X \setminus Y|$: The number of words in set $X$ but not in set $Y$ (false positives).
- $c = |Y \setminus X|$: The number of words in set $Y$ but not in set $X$ (false negatives).

Using the weight $w$, which is set by default to $3$, we apply the same preprocessing steps to our sample article as we did with the discounted Levenshtein distance. The results we obtained are shown in Table 4.3.

The results differ when we compare the search results using discounted Levenshtein distance (in Table 4.2) and Weighted Jaccard similarity (in Table 4.3). With discounted Levenshtein distance, the result for "*Amazon*" is $74.450$, while the Weighted Jaccard similarity result is $78.261$. In this case, there is an improvement in identifying the correct company name from the dataset. However, the "*Google*" result is $35.917$ for discounted Levenshtein distance and $50$ for Weighted Jaccard similarity. Even though the "*Google*" score is higher with Weighted Jaccard similarity, the correct company name is still not displayed. Instead, it shows the "*Neos*

| Organisation | Official Name | Score |
|---|---|---|
| Adobe | Adobe Systems Inc | 100 |
| Amazon | Amazon.com Inc | 78.261 |
| Google | Neos Yield Premium Strategy Google [Googl] ETF | 50 |
| IBM | Ibio Inc | 54.545 |
| Meta | Kennametal Inc | 47.368 |
| Microsoft | Microsoft Corp | 100 |
| OpenAI | Open Bank | 75 |
| TikTok | Cytokinetics | 39.130 |

**Table 4.3** Extracted organisation names from the Guardian's article and their matches with official names in the dataset using the match quality scores based on the Weighted Jaccard Similarity.

*Yield Premium Strategy Google [Google] ETF*", which needs to be corrected. Our objective is to directly label the parent Google's company, Alphabet Inc., with the ticker symbol GOOGL on the NASDAQ exchange. Therefore, higher scores do not necessarily indicate finer accuracy in this context. Similar results can be observed for the other matches. The only minimal improvement is in the case of "*IBM*", where there is a $1.102$ reduction in the score for the incorrectly labelled company name.

### Token Set Ratio

The Token Set Ratio represents another possible approach to solving our problem. It prioritizes the strings' meaning over the original word order and duplicate word removal, making the method flexible for comparing text accuracy.

Let us consider two token sets, $X$ and $Y$, denoting the name of an organization extracted from the text and an arbitrary company from the dataset, respectively. We also operate with the intersection of $X \cap Y$, representing common words between the two strings. The resulting similarity score is then determined by the highest value among the following three similarity combinations:

- The similarity between the article organisation name and common words.

$$sim(X, X \cap Y) \tag{4.3}$$

- The similarity between common words and the dataset company name.

$$sim(X \cap Y, Y) \tag{4.4}$$

- The similarity between the article organisation name and the dataset company name.

$$sim(X, Y) \tag{4.5}$$

Where $sim$ denotes the similarity score calculated by SequenceMatcher ratio[5] from the difflib library in Python. Also, in this case, keeping all preprocessing steps as in the previous two discussed metrics makes sense. Again, we will use our sample of Guardain's article to demonstrate this approach. The results are shown in Table 4.4.

| Organisation | Official Name | Score |
|---|---|---|
| Adobe | Adobe Systems Inc | 100 |
| Amazon | Amazon.com Inc | 82.353 |
| Google | Neos Yield Premium Strategy Google [Googl] ETF | 100 |
| IBM | Ibio Inc | 66.667 |
| Meta | Kennametal Inc | 62.500 |
| Microsoft | Microsoft Corp | 100 |
| OpenAI | Open Bank | 83.333 |
| TikTok | Cytokinetics | 40 |

**Table 4.4** Extracted organisation names from the Guardian's article and their matches with official company names in the dataset using the match quality scores based on the Token Set Ratio.

The Token Set Ratio achieves a higher score in most matches than the previous two similarity approaches. However, as we mentioned in our previous results discussion, this does not always point in the right direction. Ignoring the identical results for "*Adobe*" and "*Microsoft*", which we achieved for all naive approaches, there is an improvement in the case of "*Amazon*", which leads to the best result for the correct company name match. However, in the case of "*Google*", it is more of a deterioration as the score increases to $100$ for the mislabeled company from the dataset. The same problem occurs for all other cases as we increase the score for incorrect labels.

### Summary

Upon examining all the data (see Table 4.5) obtained using the presented metrics, we are still dealing with the problem of the highest scores that do not guarantee the correct identification of a company from the dataset. We can be more willing to try different combinations of parameters or different preprocessing steps. However,

[5]https://docs.python.org/3/library/difflib.html#difflib.SequenceMatcher.ratio

it needs more than the approximate string matching to retrieve "*Alphabet Inc*" for "*Google*". Another problem is the score bound, which we must determine to mark a company match as correct. In our case, we could choose a score ranging from 70 to 80, but this would mean the possibility of skipping some companies that could be correctly labelled while facing the problem of labelling the wrong company. The results are still insufficient. Therefore, we need to take a different approach to get the correct company labelling from the dataset with higher accuracy.

| Organisation | Official Name | Score | | |
|---|---|---|---|---|
| | | DL | WJ | TSR |
| Adobe | Adobe Systems Inc | 100 | 100 | 100 |
| Amazon | Amazon.com Inc | 74.450 | 78.261 | 82.353 |
| Google | Neos Yield Premium Strategy Google [Googl] ETF | 35.917 | 50 | 100 |
| IBM | Ibio Inc | 55.647 | 54.545 | 66.667 |
| Meta | Kennametal Inc | 37.796 | 47.368 | 62.500 |
| Microsoft | Microsoft Corp | 100 | 100 | 100 |
| OpenAI | Open Bank | 73.286 | 75 | 83.333 |
| TikTok | Cytokinetics | 27.436 | 39.130 | 40 |

**Table 4.5** The match quality scores for extracted organisation names from the Guardian's article using different similarity measures.

## 4.4 Entity linking approach

This section focuses on a more sophisticated technique, addressing the shortcomings encountered with the naive approach discussed in the previous section 4.3. It presents a method that leverages the possibility of adding a trained knowledge base[6] to a named entity recognition module to enhance information extraction. Utilising a knowledge base connected to a source offers significant advantages, as it allows us to fully exploit the context and meaning of the words identified by named entity recognition. The naive methods primarily relied on fundamental word similarity, which we have shown to be insufficient in some cases to solve our problem.

It is essential to point out that we are concerned with more than just basic information, such as the fact that the extracted entity is an organisation, person, or event. We already have such data through named entity recognition processing. Therefore, the optimal solution is to link the already extracted organization

[6]https://www.spacy.io/api/kb

entities with specific companies traded on the stock exchange or have relevant relationships with them (Meta Platforms owns Facebook). Furthermore, it would be beneficial to eliminate the need to set arbitrary score thresholds to determine the correct company.

We aim to be independent of a static dataset and have access to dynamic online information with a more comprehensive structure that can evolve. Thus, we present methods that outperform our naive approach and can process text with greater accuracy and success despite possible relationships. In addition, we will not use the official name in the results since the actual approach does not matter to it compared to the previous matching approach, and in the future, we will be able to obtain it based on the ticker. The main task from this chapter's problem definition is to obtain the ticker and the exchange it is associated with.

### 4.4.1 Wikidata

The cornerstone of the linking approach is Wikidata[7], which meets all the criteria needed to achieve the desired results. Wikidata is a storage repository of structured data freely available online, easily readable, and editable by humans and machines. It is a part of the Wikimedia family, which includes Wikipedia, Wikibooks, and others.

Wikidata consists of items with unique identifiers, denoted as Q<number> as QID. In Figure 4.1, the identifier for the item labelled as Microsoft is Q2283. This item has the description "*American multinational technology corporation*" and several aliases (also known as). Items in Wikidata provide statements containing individual properties tagged P<number> and their corresponding values. These values can have various types, including multiple values, item values, quantitative values, and unknown or no values. Microsoft has a property *instance of*, which refers to multiple item values describing Microsoft as a software company, enterprise, technology company, and public company. In this case, the individual values refer to other items. Additional information can be found in Wikidata glossary (Wikidata, 2024).

### 4.4.2 Spacy Entity Linker

The most suitable solution for our problem, which meets the requirements and provides adequate integration with our spaCy implementation of the named entity recognition module, is the Spacy Entity Linker library in Python, available on GitHub[8]. This library creates a pipeline with an external knowledge base built on

[7] https://www.wikidata.org/

[8] https://www.github.com/egerber/spaCy-entity-linker

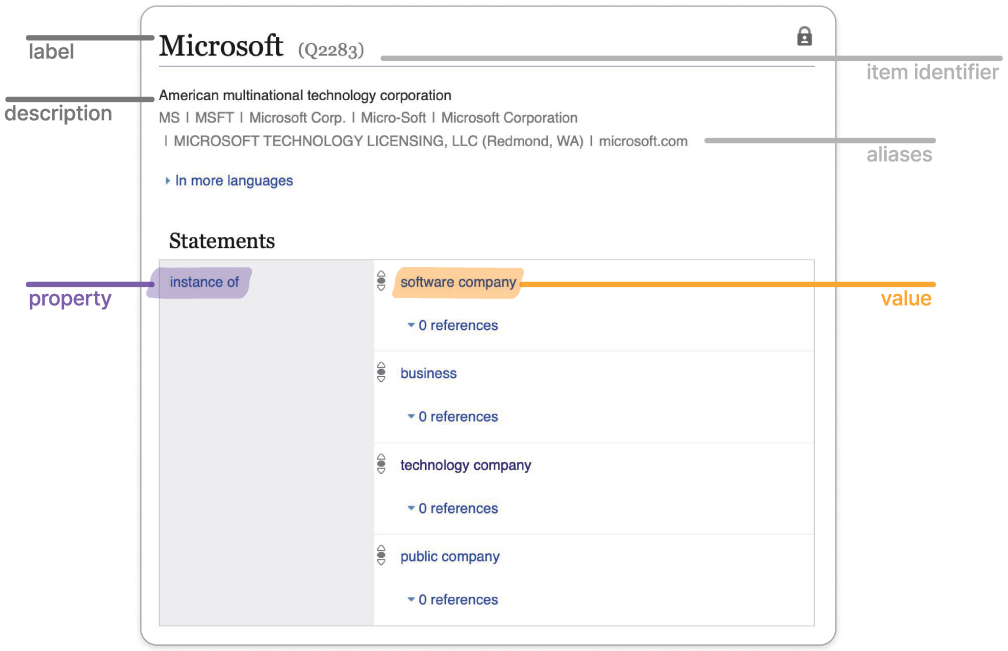



**Figure 4.1** Example of the Microsoft item in Wikidata.

Wikidata that matches possible entities from the text with potential entities (the Entity Linker refers to each item in Wikidata as an entity) on Wikidata. The main advantage of this library is that each entity found in the text provides a QID on Wikidata, allowing us to obtain additional information about the entity, including its label, aliases, description, and more.

The main priority now is to filter out entities in the text that are not identified as organizations. The Entity Linker does not provide entity type information to determine if an entity is an organization. Instead, it only uses references to other entities via the properties *instance of* (P31), *part of* (P361) and eventually *subclass of* (P279) to determine the type. For example, when we extract the entity Microsoft, we obtain information about the categories (understood as types) it belongs to, such as software company, business, technology company, and public company, which can further branch into various subcategories. This allows an entity to be part of a wide range of possible classes. However, even if an entity belongs to the organization class according to our criteria, it may be labelled differently and miss one of the classes, causing us to lose the extracted entity. Therefore, we stick to the original partitioning associated with the organization type that handles the initial named entity recognition and compare it with the entities extracted from the entity linker using span. Additionally, staying with

the original implementation of spaCy entity objects in the code will make our work more manageable in the next phase of sentiment analysis and encourage code consistency.

The Entity Linker only uses the properties we mentioned when categorizing into classes. Thus, we can not directly get information about the ticker using the Entity Linker since the *stock exchange* property provides the ticker information. Regardless of aliases, it would be necessary to compare them with a database of tickers to see if one of the many aliases has an exact match. However, the Entity Linker provides us with the QID of the entity, which we can use to get any information about the entity. This opens up several possibilities for us to get this data. First, we tried the pywikibot[9] library, which works similarly to scraping. This approach is not ideal, as each entity page must first be created as an object and then loaded from the page repository, where the data is extracted. However, it is much more efficient and more accessible to query Wikidata using the Wikidata SPARQL endpoint, giving us more modularity and flexibility we want to maintain in the code.

### 4.4.3 SPARQL Wrapper

Access to Wikidata's SPARQL endpoint is enabled by the SPARQL Wrapper library in Python, which is available on GitHub[10]. Now, we have a tool that we can use to query data about entities with a given QID using SPARQL. We will show three queries that cover the most common situations where we want to get a ticker along with the name of the exchange on which the company is traded. The queries are run sequentially, with each subsequent query processing entity that did not produce a result in the previous query. We also mention the list of essential properties that we will use in the queries:

- **P414**: The ***stock exchange*** on which the entity is traded.
- **P249**: The ***ticker symbol*** of the entity.
- **P582**: The ***end time*** of the property.
- **P127**: The entity ***owned by*** by another entity.
- **P1889**: The entity is ***different from*** another entity.

To better illustrate, we add an Instagram entity to the actual entities extracted from our sample Guardian article to demonstrate the second query associated

[9]https://www.github.com/wikimedia/pywikibot
[10]https://www.github.com/egerber/spaCy-entity-linker/tree/master

with the *owned by* property. The reader can try the following queries on the Wikidata SPARQL endpoint[11].

### Query 1: Direct ticker retrieval

The first query aims to retrieve information about a *ticker symbol* and a *stock exchange* associated with an entity's *stock exchange* property. The SPARQL query can be found in Appendix B.1. This query selects entities that directly possess the *stock exchange* property, providing details about the ticker symbols and the exchanges on which the entities are traded. It also filters the results to include only those records where the exchanges do not have an *end time* specified. The results of this query are presented in Table 4.6.

| Organisation | | Ticker | Stock Exchange |
|---|---|---|---|
| QID | Label | | |
| Q11463 | Adobe | ADBE | NASDAQ |
| Q3884 | Amazon | AMZN | NASDAQ |
| Q37156 | IBM | IBM | NYSE |
| Q2283 | Microsoft | MSFT | NASDAQ |

**Table 4.6** The results of the SPARQL Query 1: Direct ticker retrieval.

The data retrieved for individual organizations is accurate. Nevertheless, the results do not include the expected information about the Google entity. Although Google has a ticker through the *stock exchange* property on Wikidata, it also has an *end time* value set despite still being tradable on NASDAQ under the tickers GOOG and GOOGL. At the time of writing, this *end time* value can not be edited for unknown reasons, which presents a limitation we must accept. Consequently, additional queries are needed to determine details for the entities Meta, TikTok, OpenAI, and Instagram.

### Query 2: Owner-based ticker retrieval

The second query focuses on retrieving information about the exchange and ticker of the entities associated with the querying entity through an *owned by* relationship. The SPARQL query is displayed in Appendix B.2. This query selects entities with the *owned by* property and retrieves information about the tickers and exchanges associated with those entities. It explicitly targets entities related to the queried entity through the *owned by* property. The results of this query are shown in Table 4.7.

[11] https://www.query.wikidata.org/

| Organisation | | Ticker | Stock Exchange |
|---|---|---|---|
| QID | Label | | |
| Q209330 | Instagram | META | NASDAQ |

**Table 4.7** The results of the SPARQL Query 2: Owner-based ticker retrieval.

The retrieved data are again accurate. Meta Platforms indeed own Instagram, which is accessed through the Instagram *owned by* Meta Platforms association. In this case, we successfully filtered out the FB ticker, now META[12], due to the *end time* filtering.

### Query 3: Differentiated ticker retrieval

The third query retrieves information about the exchange and ticker of entities associated with the querying entity through a *different from* relationship. The SPARQL query is presented in Appendix B.3. Similar to the owner-based ticker retrieval, this query selects entities with the *different from* property and retrieves information about the tickers and exchanges associated with those entities. It focuses on entities related to the queried entity through the *different from* property. The results of this query are shown in Table 4.8.

| Organisation | | Ticker | Stock Exchange |
|---|---|---|---|
| QID | Label | | |
| Q18811574 | Meta | META | NASDAQ |

**Table 4.8** The results of the SPARQL Query 3: Differentiated ticker retrieval.

Again, the obtained data aligns with what is generally accepted as accurate. However, it is essential to note that Meta represents multiple distinct entities (as many others) on Wikidata. Therefore, querying the *different from* property is necessary in this context to retrieve Meta Platforms.

### Summary

Using the three queries outlined above, we have successfully covered a noteworthy amount of organizations that are directly traded or have relationships with companies traded on the exchange. As shown in Table 4.9, we achieved an almost one hundred per cent success rate. However, Google is not included in the results due to the previously mentioned issue with the *end time* setting. We must accept this limitation for now and will not alter our overall approach because of one

[12]https://www.bloomberg.com/quote/FB:US

entity. Regarding changing the data in Wikidata, a future opportunity to modify can arise.

Additionally, TikTok and OpenAI are absent in the results, which is expected since neither has a ticker or a relationship with a company traded on an exchange. Our querying approach allows flexibility, enabling us to create additional filtering queries as needed. This flexibility is a valuable advantage in refining our results.

| Organisation | | Ticker | Stock Exchange |
|---|---|---|---|
| QID | Label | | |
| Q11463 | Adobe | ADBE | NASDAQ |
| Q3884 | Amazon | AMZN | NASDAQ |
| Q37156 | IBM | IBM | NYSE |
| Q2283 | Microsoft | MSFT | NASDAQ |
| Q209330 | Instagram | META | NASDAQ |
| Q18811574 | Meta | META | NASDAQ |

**Table 4.9** The results of the SPARQL queries for all extracted organisation names from the Guardian's article.

# 5. Entity-level Sentiment Analysis

This chapter focuses on entity-level sentiment analysis and includes concise experiments. Using our named entity recognition algorithm described in Chapter 4, we identify organisations along with their corresponding ticker symbols. Instead of presenting naive approaches that involve holistically parsing sentences where entity mentions occur and subsequently analysing only sentiment-determining keywords associated with those entities, we describe and partially modify an algorithm based on the FinABSA model, which we have chosen as the foundation for our sentiment analysis.

We also discuss the challenges associated with finding a suitable dataset to validate our algorithm. Finally, we present the results of our experiment using the FinEntity dataset, which contains paragraphs from newspaper articles, entity annotations, and corresponding sentiments. For the sentiment analysis, we achieved a success rate of $92\%$. However, it is essential to note that this high accuracy was obtained after significantly reducing the original dataset to ensure precise measurement.

Additionally, we explore the correlation between entity-level sentiment and adjusted close prices of stocks, demonstrating promising relationships that underline the potential of sentiment analysis in financial markets. A detailed analysis of these experiments is available in a dedicated Python notebooks[1] that provides comprehensive insights into our methodology and findings.

## 5.1 Introduction

At the current stage, leveraging our named entity recognition algorithm discussed in Chapter 4, we have identified entities, specifically organizations and their corresponding tickers, which we refer to as companies. The algorithm selectively filters these companies to include only those listed on specific exchanges, namely NASDAQ, NYSE, and AMEX. This section focuses on sentiment analysis at the entity level within these exchanges.

We will omit naive approaches involving sentiment analysis, where sentences mentioning an entity are analyzed holistically through the whole sentence. Such approaches may yield partial insights assuming a single entity per sentence, attributing uniform sentiment to all mentioned entities. However, these methods

[1] Located in the directory /ipynbs/entity-level-sentiment-analysis/.

overlook whether keywords related to sentiment within sentences are associated with the entity.

Another comparable approach is entity-level sentiment analysis, where sentences are considered based on keywords determining sentiment in coexistence with the mentioned entity. This method offers efficiency and addresses challenges posed by multiple entities within a single sentence, establishing a direct link between keywords and entities. Nevertheless, complications may arise in interpreting keywords with ambiguous sentiments linked to other words so that their meaning can vary and deepen, relying on the other words, such as in the simple example of "*like*" and "*not like*".

## 5.2 FinABSA model

Instead of considering naive approaches, we will focus on slight modifications to meet the needs of our sentiment analysis algorithm implementation based on the FinABSA model[2]. This model was trained specifically for the financial domain using the SEntFiN 1.0 dataset (Sinha et al., 2022). Even though the model is considered aspect-based, where aspects refer to targets as "*[TGT]*", its practical application extends to entity-level sentiment analysis by defining these targets through any tagging mechanism. The model excels at handling multiple entities across sentences and enables numerical sentiment analysis. To derive an overall entity's sentiment score and distribution, it categorises sentiments into positive, neutral, and negative classes based on the highest value ranging from $0$ to $1$.

The model is available in two versions, namely FinABSA[3], which shows lower performance with longer sentences, and FinABSA-Longer[4], which should be able to handle more extended sentences. Due to the length of articles we are dealing with, we have chosen the extended version. However, both versions are constrained by a token limit of $512$. Therefore, in our algorithm adaptation, we process input articles in chunks, each containing a maximum of $512$ tokens. It is crucial to avoid splitting sentences based on token count into two parts to prevent loss of context and ensure accurate analysis. If a last sentence exceeds the token limit within a chunk, it is taken into the subsequent chunk. These chunks are processed in parallel, and their outcomes are consolidated as a collected result. The final sentiment determination for each entity is derived from the highest averaged class scores.

Regarding entity tagging, we leverage our already mentioned entity recognition algorithm. This algorithm guarantees entities representing companies with

[2]https://www.github.com/guijinSON/FinABSA

[3]https://www.huggingface.co/amphora/FinABSA

[4]https://www.huggingface.co/amphora/FinABSA-Longer

the ticker. Each identified entity is then mapped to a target "*[TGT]*", enabling the model to recognise companies and perform sentiment analysis accordingly.

## 5.3 Experiment: FinEntity Dataset

Obtaining a suitable dataset to validate the correctness of our algorithm presents significant challenges, such as the necessity of at least text excerpts from news articles, including companies as entities and their associated sentiments. Publicly available datasets possessing entire articles are almost impossible to obtain. There are enough datasets containing entity information concerning tweets or article headlines focusing on single-entity contexts, proving non-optional for our experimental needs because both tweets and headlines have a different structure than articles.

Creating our dataset would be excessively time-consuming or monetarily expensive if relying on human annotators. Nevertheless, the work (Tang et al., 2023) describes the development of a FinEntity dataset, available on Hugging Face[5] built on the financial domain with the contribution of financial experts. This dataset contains $979$ paragraphs sourced from newspaper articles, each annotated with entities and their corresponding sentiments. Notably, each paragraph may encompass multiple entities with their associated sentiments. Despite the dataset containing only excerpts rather than entire articles, it remains the most suitable option for our experiment.

The entities within the FinEntity dataset consist of companies without explicit information on their details about exchange listings. Consequently, our named entity recognition algorithm is restricted to NASDAQ, NYSE, and AMEX entities. Thus, we must filter out from dataset entities not traded on these exchanges. In addition, some annotations do not have information about the associated ticker. Thus, we must remove these entities to determine whether they are traded on the exchanges we consider in our named entity recognition algorithm. Table 5.1 gives the specific entity counts, which do not consider unique entities - taking into account entity occurrences and duplicate entities as well. After filtering, $540$ entities remain available for our experiment, distributed across $282$ paragraphs out of $979$ (see Table 5.2).

Our named entity recognition approach recognised $460$ entities, each of which does not necessarily equal any of the $540$ already annotated entities in the dataset. Our entities include, for example, those that did not contain a ticker and were removed from the annotations as a result.

Upon closer examination, it becomes evident that inconsistencies in annota-

[5]https://www.huggingface.co/datasets/yixuantt/FinEntity

| Entities | |
|---|---|
| Description | Count |
| Total | 2,131 |
| Without ticker | 1,359 |
| With ticker | 772 |
| **On specified exchanges** | **540** |

**Table 5.1** Entities summary in FinEntity dataset. (Note: Including duplicates.)

| Paragraphs | |
|---|---|
| Description | Count |
| Before cleaning | 979 |
| **After cleaning** | **282** |

**Table 5.2** Paragraphs before and after cleaning summary in FinEntity dataset.

tion, such as Tesla sometimes being tagged with a ticker and sometimes without, contribute to the variation in the number of entities retrieved. This inconsistency also obscures unrecognised entities due to the limited context provided by the shorter paragraphs. Some entities were not identified by our named entity recognition algorithm because they did not progress to the stage of querying Wikidata. For instance, Johnson & Johnson[6], despite having stock exchange NYSE information listed on Wikidata, was not recognised as an organisation by the spaCy tagger. The Entity Linker may also need more information about the entity, resulting in an undetermined QID. Our methodology currently employs only three query variants over the SPARQL data. If the Entity Linker identifies an entity as Johnson & Johnson (United Kingdom)[7], we could enhance accuracy by incorporating a query that considers the "*parent organisation*" property. This would ensure that the primary entity, Johnson & Johnson, which contains the relevant stock exchange information, is correctly recognised.

However, this section primarily focuses on entity-level sentiment analysis to evaluate sentiment. After extracting entities, our sentiment analysis algorithm, rooted in the FinABSA-Longer model, processes these entities to determine their sentiment. Before presenting the results of the experiment, we describe three perspectives that we focus on to evaluate the results for each paragraph:

**Matched** : The number of entities with the same ticker and entity's text that have the same sentiment in the dataset and our annotations.

[6]https://www.wikidata.org/wiki/Q333718
[7]https://www.wikidata.org/wiki/Q30338424

**Unmatched** : The number of entities with the same ticker and entity's text that have different sentiments in the dataset and our annotations.

**Unfound** : The number of entities that are in the dataset but not in our annotations.

Table 5.3 and Figure 5.1 show that the number of matched entities is 287, representing 53.1%. There are also 25 unmatched entities whose sentiment did not match the dataset, accounting for 4.6%, and 228 entities not found in our annotations, making up 42.2%. The success percentages may seem low if we only consider these unmatched entities. Thus, focusing only on sentiment evaluation, the results are entirely sufficient, with a success rate of 287 out of 312, or 92.0% (see Figure 5.2).

| **Entities** | |
|---|---|
| Description | Count |
| Matched | 287 |
| Unmatched | 25 |
| Unfound | 228 |

**Table 5.3** Results of entity-level sentiment analysis. The proportion of Matched, Unmatched, and Unfound entities.

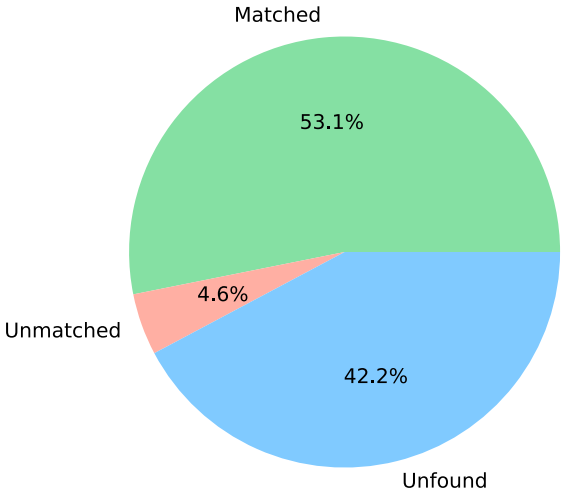



**Figure 5.1** The pie chart shows the result proportion of Matched, Unmatched, and Unfound entities.

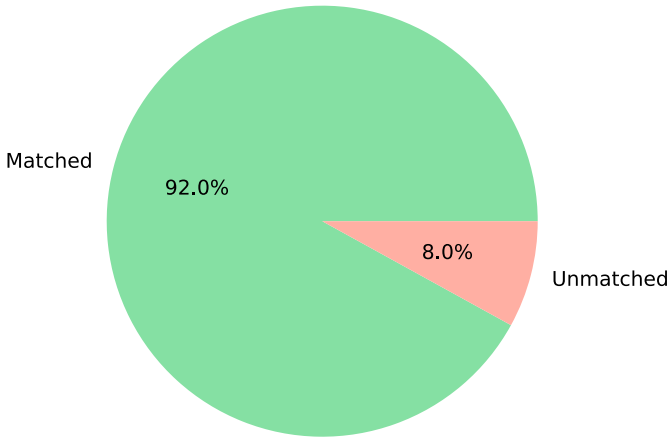



**Figure 5.2** The pie chart shows the result proportion of Matched and Unmatched entities.

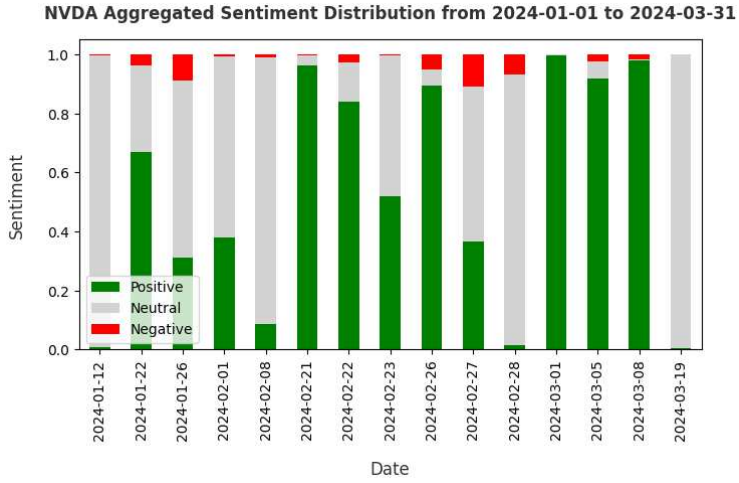
## 5.4 Experiment: Sentiment and Adjusted Close Price

Once we have built an algorithm to extract companies and their associated sentiment from the text, we can analyse how sentiment value and stock price are related. For this purpose, we utilise the Yahoo Finance Python library[8] to obtain historical stock adjusted close price (Quantstart, 2024) of companies based on their ticker symbols. The news articles are sourced from the Guardian, providing us with enough data including whole content for analysis. We focus on articles from the business and technology sections published in the first quarter of 2024, from January 1 to March 31.

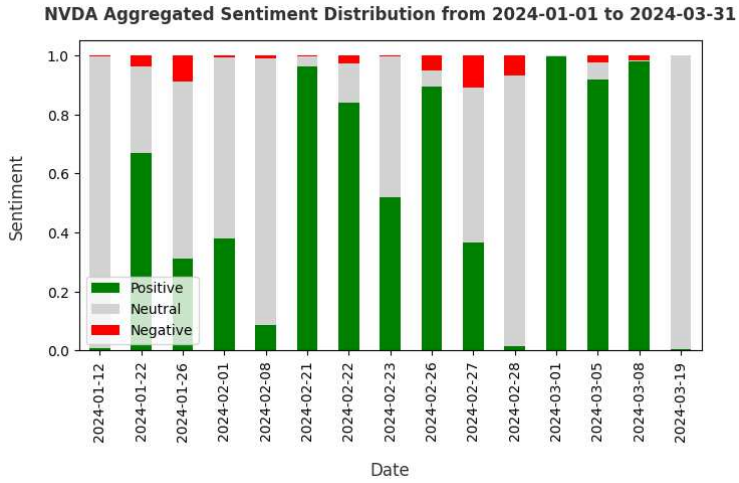
The data, such as articles from the business and technology sections, are published inconsistently. Multiple articles may be published on any given day. In such cases, we will consider the average sentiment of all articles. We assign the sentiment value to the next market day if an article is published on a non-market day, such as a weekend or holiday. The data from the original distribution of articles before aggregation is shown in Figure 5.3a, and after aggregation in Figure 5.3b. The figures illustrate the results of the sentiment analysis of the company Nvidia Corp. (NVDA) evaluated by our named entity recognition algorithm and using the modified FinABSA-Longer model. The company's sentiment is divided into positive, neutral, and negative components, totalling a value of 1.

It is worth reminding the basis behind our choice of entity-level sentiment

[8]https://www.pypi.org/project/yfinance/




**(a)** Distribution of sentiment before aggregation.




**(b)** Distribution of sentiment after aggregation.

**Figure 5.3** The sentiment analysis of Nvidia Corp. (NVDA) in the first quarter of 2024.

analysis. Rather than assessing the overall sentiment of an entire article, we focus specifically on the sentiment related to Nvidia. This distinction is crucial because the overall sentiment of an article may not accurately reflect the sentiment toward individual companies mentioned within it. Providing sentiment analysis on the entity level gives investors more precise and actionable insights.

For instance, if we were to evaluate the sentiment of an entire article without isolating the sentiment tied to specific companies, the resulting sentiment score could be misleading. An article might have a generally negative tone but still contain positive sentiments about Nvidia, or vice versa. Our approach ensures that the sentiment value reflects only the sentiments pertinent to a single company, making it a reliable indicator for investors.

### 5.4.1 Hold Strategy

A positive sentiment towards a company typically suggests a potential rise in its stock price. Consequently, if the sentiment analysis indicates a predominantly positive sentiment, it would be classified as positive, a potential buy opportunity marked as a buy signal. Conversely, if the analysis reveals a predominantly negative sentiment, it would be classified as negative, suggesting a sell signal. If the sentiment is predominantly neutral, it would be classified as neutral, indicating without action.

Let us define a simple, straightforward holding strategy without using any other indicators. Buy and sell signals from previous day trigger corresponding position entries: long positions for buy signals and short positions for sell signals. Both positions are held for a predetermined duration of seven days[9]. The initial capital is $\$10,000$, and each position is facilitated by shares corresponding to $\$100$. According to Figure 5.4, Nvidia registered only buy signals during the first quarter of $2024$, meaning positive sentiment classifications only. The results of this strategy can be seen in the portfolio value in Figure 5.4. In this case, the average sentiment is positive, which aligns with the expectation that positive sentiment implies a stock price increase.

Our aim is not to create the perfect investment strategy but to provide a foundational understanding of using sentiment data. Focusing on the data related to Nvidia and its sentiment distribution after aggregation (Figure 5.3), we notice frequent occurrences of neutral sentiment values, which we exclude from our strategy. To maximise the utility of the data when encountering neutral classifications, we refine the approach. If the positive sentiment value exceeds the negative sentiment value, we classify it as neutral lean positive. Otherwise, it is classified

[9]For an initial exploration of sentiment analysis, we will focus solely on the core concept and disregard temporal considerations and position entry detailed specifics.

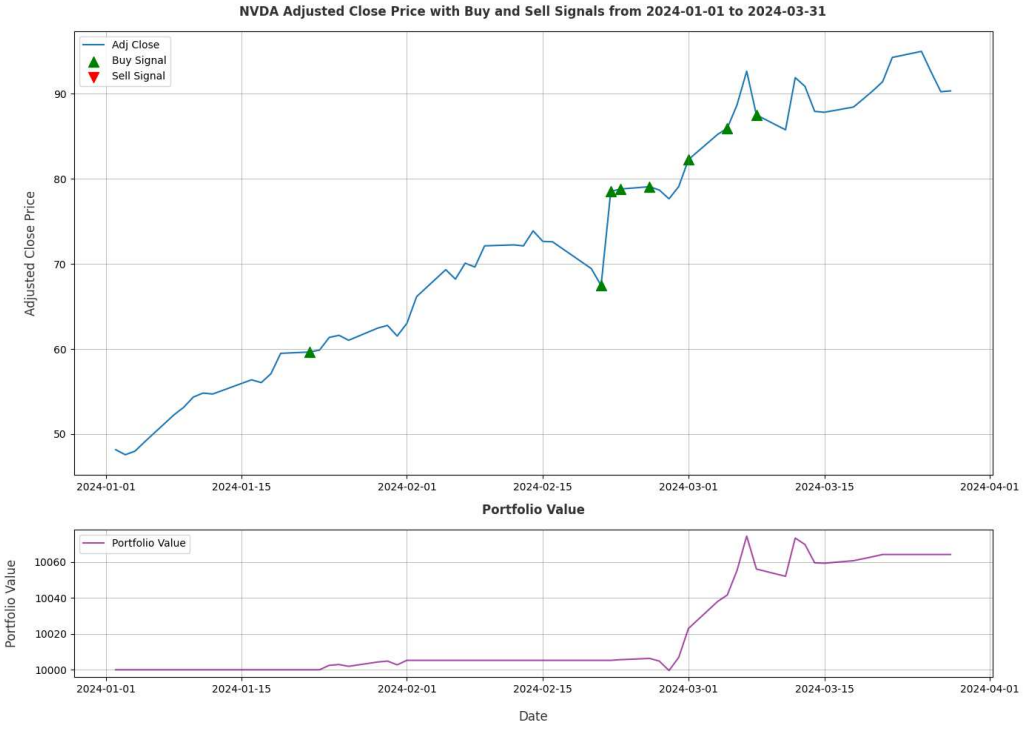



**Figure 5.4** Nvidia Corp. (NVDA) adjusted close price with buy and sell signals in the first quarter of 2024. Hold strategy based on sentiment signals.


**Figure 5.5** Nvidia Corp. (NVDA) adjusted close price with the buy, sell and neutral signals in the first quarter of 2024.

as neutral lean negative. The Nvidia with neutral categories results are shown in Figure 5.5.

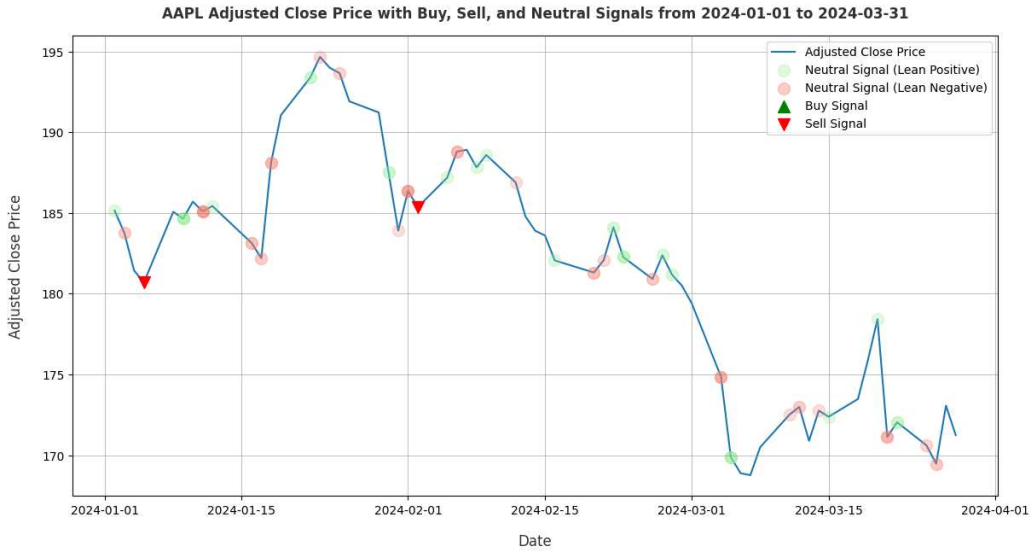



**Figure 5.6** Apple Inc. (AAPL) adjusted close price with the buy, sell and neutral signals in the first quarter of 2024.

For a more precise illustration of neutral sentiment projection, we apply it to Apple Inc. (AAPL), as depicted in Figure 5.6. This technique ensures that even neutral sentiments are interpreted to maximise the data's potential. Additionally, the sentiment value adjusts the colour's alpha channel, increasing it by approximately $0.25$ for better visibility. This procedure allows us to intentionally overlook sentiments with high neutral values and low positive and negative values. We ensure that the most relevant sentiments are highlighted, enhancing the clarity and effectiveness of our analysis. Testing the strategy and displaying neutral sentiment for other companies, such as Apple, Amazon.com Inc. (AMZN), Alphabet Inc. (GOOGL), Meta Platforms Inc. (META), Microsoft Corp. (MSFT), and Tesla Inc. (TSLA), we obtain the results shown in Appendix C.1. The results demonstrate the effectiveness of our approach using buy and sell signals. There is potential for further development in employing neutral sentiment.

### 5.4.2 Correlation

Considering the correlation between sentiment and adjusted close price is appropriate to avoid concluding and provide an independent view based on the chosen market strategy. First, we will focus on the correlation between sentiment and the stock price of Nvidia. Recalling the sentiment distribution after

aggregation in Figure 5.3b, which we extend to include neutral categories lean positive and lean negative, all sentiment data are interpolated using the previous day method, see Figure 5.7. The strategy initially focused solely on positive and negative sentiments, but we are also interested in neutral sentiment for correlation purposes.

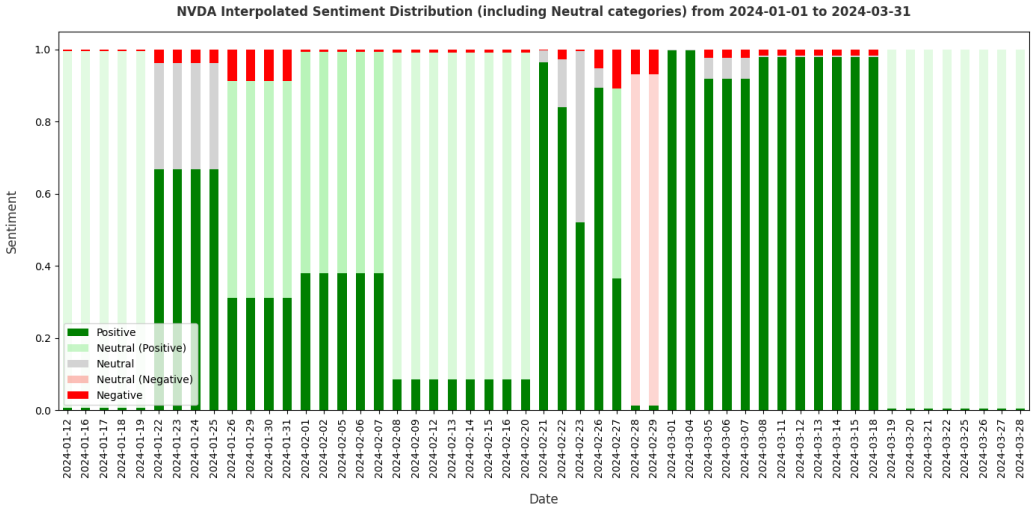



**Figure 5.7** Nvidia Corp. (NVDA) adjusted close price with interpolated sentiment in the first quarter of 2024.

The choice of interpolation method is based on the policy that if sentiment data are missing for a given day, we use the values from the previous day. This approach is chosen because it aligns with sentiment behaviour. For instance, linear interpolation could be chosen if sentiment changes were expected to be linear, which is unrealistic. Using the previous day method allows us to simulate that the company sentiment remains the same if no new sentiment data are available.

A negative correlation between negative sentiment and adjusted close price suggests that the adjusted close price tends to decrease as negative sentiment increases. Conversely, the adjusted closing price tends to increase as negative sentiment decreases. This negative correlation indicates an inverse relationship between these variables. Therefore, negative and neutral lean negative sentiments in the results are inverted. The correlation results between sentiments and Nvidia's adjusted close price are shown in Figure 5.8. The results show a positive correlation with a value of about $0.25$ in three sentiment cases, which means a successful correlation between sentiment and stock prices. In the case of lean negative, the correlation is negative, which means that the adjusted close price

increases as negative sentiment grows. The correlations of other companies are shown in Appendix C.2.

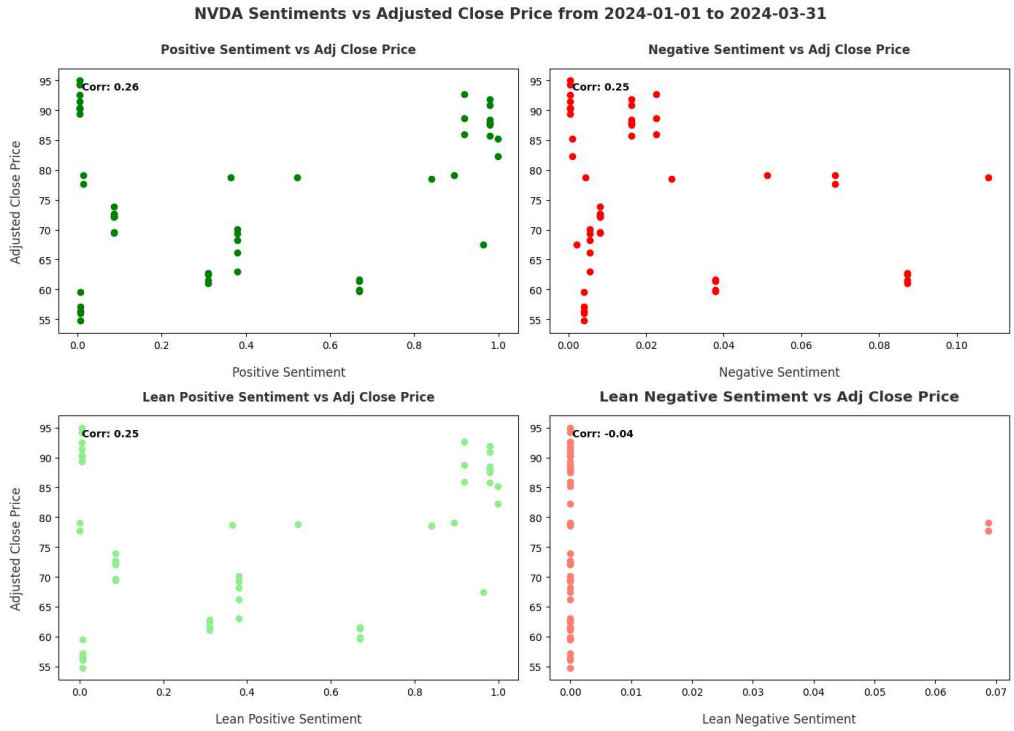



**Figure 5.8** Nvidia Corp. (NVDA) sentiment correlation with adjusted close price in the first quarter of 2024.

If we look at both the strategy and the correlation analysis results, they are slightly satisfactory. The strategy employs a straightforward approach and does not account for other factors that could influence stock prices. High negative or positive sentiment is thus partly a reaction to stock prices and partly an influence of them. For better accuracy, setting a threshold for classifying a given sentiment could help raise the level of sentiment required to be considered purely positive or negative. Neutral sentiment also plays a role and can be helpful in certain situations, as confirmed by the correlation with the adjusted close price.

The sum of the profits from portfolios based on this strategy is positive. We achieved a total return of about $0.74\%$, which amounted to approximately $\$74$ profit. Although there were more loss-making positions than profitable ones, the losses were minor compared to the more significant profits. If we had more articles available in given time period, there would undoubtedly be more opportunities, but the question remains whether they would be reliable and not too numerous. We should set a boundary for when we are willing to enter positions.

Regarding the correlation analysis, the results confirm that utilising the full potential of negative and positive sentiment data in classifying neutral sentiment is sensible. Additionally, some correlation, albeit small, does exist and is mostly positive. This indicates that sentiment can be a good indicator of stock prices. It is important to remind readers not to be misled when comparing the strategy portfolio and correlation results, as the portfolio value is based on positive and negative sentiment, and positions are held for seven days. The results are thus rather satisfactory. However, further experiments with more data and different strategies, additional indicators, and conceivably a different interpolation method in the case of correlation would be appropriate.

# 6. Application Requirements

The application requirements are clear and concise, so this Chapter will be less extensive than the others. It discusses the requirements for an application of this specific nature within the realm of possibility. In addition, we consulted some of the steps with long-standing stock investors from the context of another confidential project and had discussions with similar groups to refine them.

We primarily want the user to be able to verify and test historical data to see how our application has evaluated previous sentiment data and how the market reacted before the user considers the values our application will evaluate for them. Looking at the price reactions immediately after the sentiment labels illustrated during experiments in Chapter 5 and also in Appendix C should be sufficient to display a three-month timeframe.

## 6.1 Functional Requirements

The functional requirements are as follows:

- **Company Search**: Allow users to search for a company based on its name or ticker symbol.
- **Company Information Display**: Display brief information about the selected company.
- **Article Sentiment Display**: Show articles in which the company appears, including the sentiment associated with the company.
- **Article List**: Provide an option to display all articles mentioning the company.
- **Current Sentiment Value**: Display the current sentiment value of a given company as an average of the sentiment values for a current day.
- **Sentiment and Price Display**: Simultaneously display the sentiment value and the company's adjusted close price in one graph to enhance analysis.
- **Company Connections**: Display how individual companies are connected and which sentiment values maintain these connections, conducting sentimental trends around them. Therefore, the ideal solution could be to use a graph network.

- **Article Sentiment Calculation**: Display the sentiment value of an article based on the average of the individual sentiments of the mentioned companies.
- **Article Access**: Provide access to read the articles so users can perform their analysis and verify the values given by the application. Due to the nature of the data, the application should provide a link to the original article.
- **User Interface**: Ensure the user interface is user-friendly and intuitive to navigate as much as possible.

## 6.2 Non-Functional Requirements

The non-functional requirements are as follows:

- **Modular Processing**: Ensure modular processing to facilitate the easy addition of new analysis components.
- **Data Availability**: Ensure that data is available during updating and processing so that users can access the data at any time.
- **Data Integrity**: Ensure that the cornerstone news data is consistent to provide reliable results.
- **Data Accuracy**: Ensure that the application does not present data noise and is accurate and reliable to provide the best results.
- **Scalability**: Ensure the application is adaptable for more sources and companies.

# 7. Architecture

This chapter will introduce the application's architecture, divided into the backend and frontend, emphasising its distinct roles and functionalities. The backend section delves into four primary services: ETL, NER, SA, and the REST API. Each service is detailed with insights into the technologies and frameworks used for implementation, highlighting their modular design and performance optimisations. The chapter also provides an overview of the database structure, essential phases of ETL, and communication between services crucial for enhancing overall backend performance. We also set the repetition period of the ETL pipeline to 4 hours.

The chapter then moves to the frontend, which is developed in TypeScript using the Angular framework. It describes its role in connecting users to the application's data via the REST API. Various frontend components such as Home, Graphs, CompanyGraph, CompanyInfo, and others are thoroughly explored and their functionality and capabilities for interactive data visualisation in our application are presented.

## 7.1 Introduction

The architecture explanation in this chapter is partitioned into the backend and the frontend. The backend is written in Python and utilises several frameworks, allowing for the separation of responsibilities and facilitating development. The backend is further divided into four essential services, which can be distributed across different servers, thereby achieving better modularity and performance enhancement:

- **Extract Transform Load (ETL)** service is responsible for acquiring, transforming, and storing article data in the database.
- **Named Entity Recognition (NER)** service identifies and classifies company entities in textual data.
- **Sentiment Analysis (SA)** service analyses the sentiment of the text on the entity level, determining whether it is positive, negative, or neutral.
- **Representational State Transfer Application Programming Interface (REST API)** provides access to data stored in the database.

This separation is fundamental when working with large amounts of data, which are subsequently processed using trained artificial intelligence models. The primary service is ETL, which communicates with an external data source during extraction. It collaborates with the NER service in the transformation phase to extract entities from the article, then moves on to communicate with the SA service, which performs sentiment analysis on these extracted entities. Data is continuously saved to files in the extract and transform phases. In the load phase, the results in these files are stored in a database. The Representational State Transfer Application Programming Interface (REST API), running as a separate service, facilitates communication between the database and the frontend.

The frontend is written in TypeScript using the Angular[1] framework. It provides a user interface for interacting with backend and data in the database. Angular communicates with the backend REST API to fetch and display data. Although it is a frontend framework, Angular needs infrastructure such as Docker or similar software to operate on a server. This infrastructure handles deployment and serves the frontend application to users. Therefore, deploying an Angular application requires a server to manage communication with the backend, routing and navigation between pages, and static file serving to users. However, Angular itself does not execute backend logic. The architecture of the application is illustrated in Figure 7.1.

## 7.2 Backend

The backend is divided into several services, each with a distinctly defined role. This section describes each service and the reasons for their division and usage.

### 7.2.1 Extract Transform Load Service

The ETL service is responsible for acquiring data from external sources, transforming it, and storing it in the database. This process includes three phases:

- **Extract:** Acquiring data from the Guardian through Open Platform and saving it to files for further processing.
- **Transform:** Transforming the data, including communication with NER and SA services to extract entities and analyse sentiment. Results are saved to files.
- **Load:** Storing the transformed data from files to the database.

[1] https://www.angular.dev

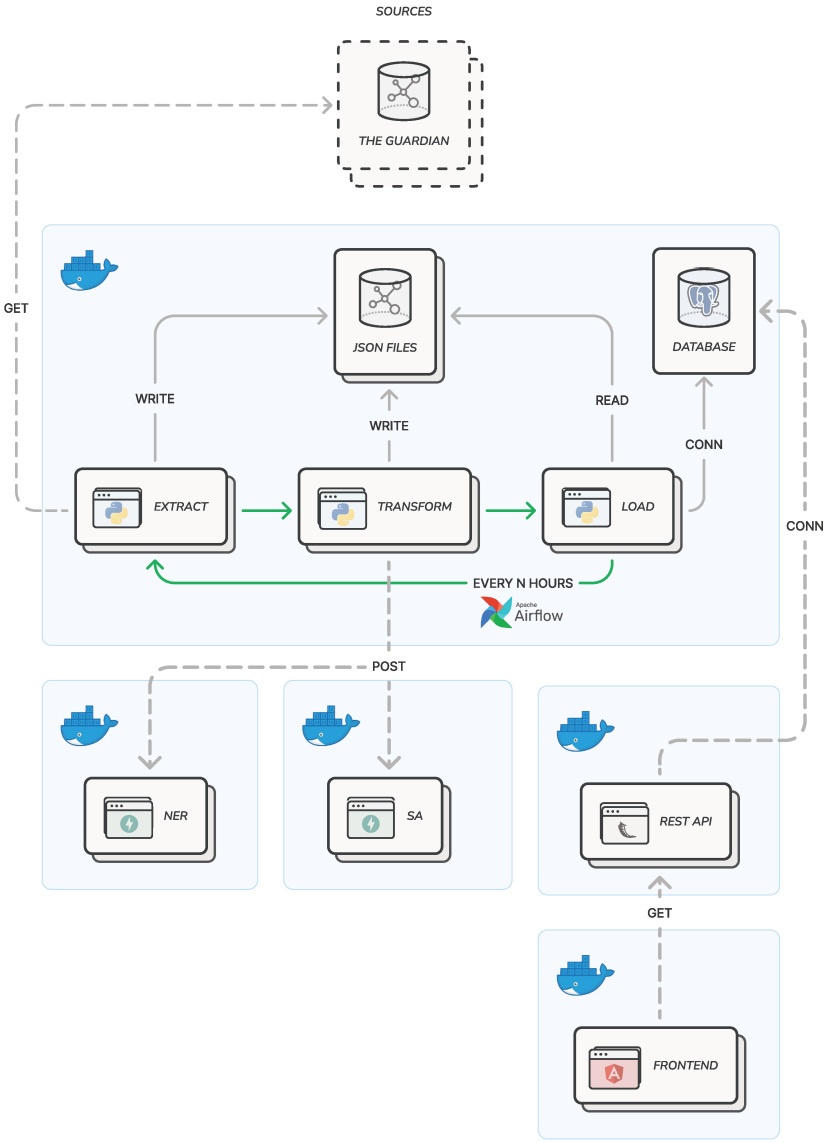



**Figure 7.1** The application's high-level architecture containing individual services and their communication. Where N hours is a variable, which will be set to 4 hours in the current version.

To facilitate the ETL process, the application uses the Apache Airflow[2] framework as a task orchestrator. Airflow is an open-source tool for planning and managing the flow of data tasks. This framework was chosen because it can schedule and execute tasks in a specific order and under certain conditions. It provides a web server that allows us to monitor task status, view logs, and display results. The scheduler triggers tasks based on a defined schedule while the executor executes these tasks. The application uses the local executor, which runs tasks in parallel on the same machine.

This service also includes two PostgreSQL, in short Postgres, databases. The first database manages the operation of Airflow, while the second database stores the results of the ETL process. Additionally, Airflow itself recommends using a Postgres database to store metadata. The choice of Postgres for article-related data is based on its reliability and ability to process large amounts of structured data. It provides a robust relational model ideal for data such as articles, companies, and sentiments. This model allows for efficient data storage in well-defined tables with relationships between them using foreign keys, ensuring data integrity and facilitating complex querying.

Since we are covering a scheduling pipeline for processing text data, which repeats at regular intervals, it is necessary to ensure that data is constantly available. Therefore, replication of all data in the database as temporary copies of tables occurs at the beginning of the Directed Acyclic Graph (DAG) before the extract phase. The original data is deleted from these tables to free up space for new data that will be processed during the current run of the pipeline. By creating copies, we ensure users always have access to data, even during insertion, preventing disruptions. After the load phase, at the end of the DAG, the temporary tables and their data are removed when all the new data are successfully inserted into the original tables, and subsequent additional tasks are finished.

Considering that the data after the transform phase contains only entities as tickers and their associated sentiments, obtaining additional information about the company based on the ticker is necessary. For this purpose, the application uses the Yahoo Finance Python library, which we have used in previous chapters. This library allows obtaining company data such as the name, industry, and website. These pieces of information are then obtained and stored in the database during the load phase after loading the transformed data. Once the data is loaded into the database, we will have information about all the tickers available and will not need to obtain data repeatedly during the transformation. Finally, unnecessary articles that do not contain an entity are removed. This process ensures that only relevant data is stored in the database, reducing unnecessary data storage and improving data quality, see Figure 7.2.

[2]https://www.airflow.apache.org

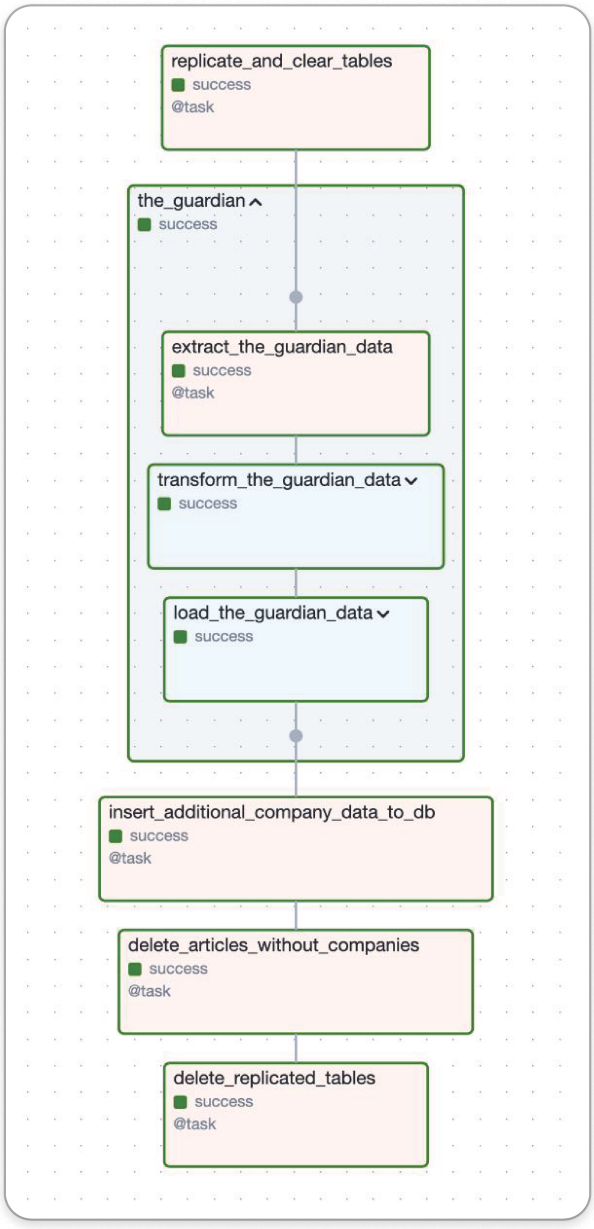



**Figure 7.2** The DAG of Guardian's ETL phases, including four additional tasks for data tables replication and cleaning, insertion of additional company data, deletion of articles without companies, and deletion of replicated tables. Oriented from top to bottom.

### Database Details

As mentioned, Postgres is the primary database for storing data and utilises the relational model. Our database implementation has several tables, as shown in Figure 7.3. In addition, the Database Markup Language[3] (DBML) schema can also be found in the attatchment[4].

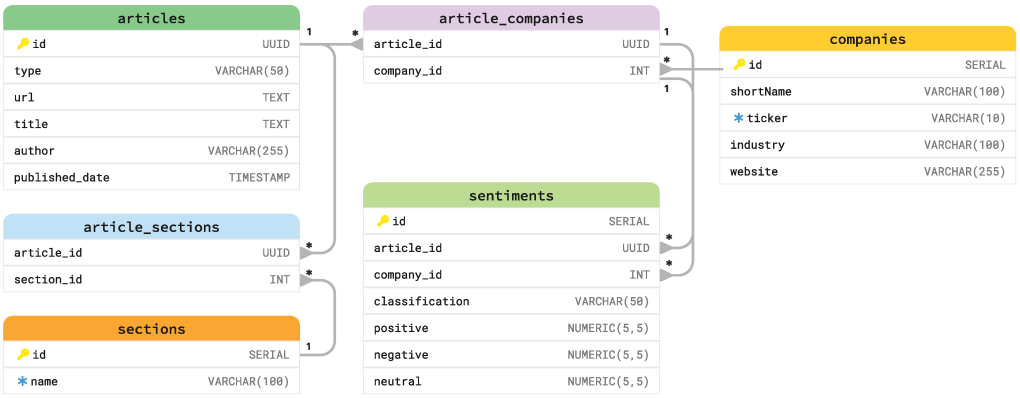



**Figure 7.3** Relational schema of ETL service database for storing article-related data.

Given the extensive utilisation of the join operation in our queries, we have established the three supplementary indexes on columns commonly employed for table joins. The following indexes are designed to improve the performance of queries that join data from these tables and respond to frequent queries more efficiently:

- ***idx_article_companies_article_id_company_id*** for columns *article_id* and *company_id* on the *article_companies* table
- ***idx_sentiments_article_id_company_id*** on the sentiments table for columns *article_id* and *company_id*
- ***idx_article_sections_article_id_section_id*** on the *article_sections* table for columns *article_id* and *section_id*

Suppose we would like to incorporate more types of entities into the database in the future, such as people, events, or products with various relationships between them. In that case, we might consider using a graph database like Neo4j. Nevertheless, the relational model in the current stage of the application version is the most suitable choice for storing current type of data.

[3]https://dbml.dbdiagram.io/home
[4]Located in the directory directory /docs/database/

### Extract Details

During implementation, we focused on making connecting additional sources for extraction effortless. Currently, we use only one source, the Guardian, for reasons outlined in Chapter 3. If we wanted to add more sources, it would be cautious to discuss and consider a filter to determine how similar articles from different sources are to avoid duplicating sentiment values from articles with the same content. Alternatively, we could account for this information in another way, such as weighting the sentiment based on the source's popularity and potential reach, which could impact the market.

Each source creates a separate tasks pipeline in Airflow within a single DAG, containing all phases of the ETL process. This approach allows us to easily add new sources without modifying the existing pipeline. The phases are divided into tasks and task groups, ensuring clarity, comfortable navigation in the code, simple implementation, and the possibility of parallel execution of individual tasks.

Specifically, extraction is performed as a single task for the Guardian source by requesting the Open Platform. The time frame for extracting articles is set to three months. During this period, about $1000$ articles is extracted, focusing on the business and technology sections. The endpoint paginates data in batches of $200$ articles. After determining the number of pages, a task is created for each page to retrieve the data and save it to a file. This method was chosen because the API limits the number of articles we can retrieve simultaneously, making pagination necessary. On the other hand, we can use this guideline to clearly define the data size distribution for individual tasks. Thus, it allows us to monitor the extraction process and ensure parallel execution in the subsequent transform phase.

### Transform Details

The transform phase begins by reading the extracted data stored in files. It simplifies storing the final transformed data without copying and updating existing files. This ensures the data is available even if an error occurs in the subsequent phases and allows for continuous monitoring. Airflow provides cross-communication (X-Coms) data, which is loaded into metadata and can be easily passed between tasks. Although we know about the limitations, it is generally not recommended to transfer extensive data (approximately $1$ GB) using X-Com with a Postgres database supporting Airflow (Laura, 2023). In our case, individual files of $200$ articles with file sizes ranging from $1$ to $2$ MB, making it acceptable with a substantial reserve.

The NER task pulls article content data from X-Coms and sends the content to the NER service API endpoint. The response contains entities, each with its ticker and the substring defining the entity in the text. The result is then pushed

into X-Coms as separate data. It ensures that the NER task results are available for the subsequent SA task.

The SA task pulls article content and entity data from X-Coms. It sends the content and identified entities to the SA service API endpoint, which responds with sentiments for each entity. The results are then pushed into X-Coms as another data collection containing entities and their sentiments for each article.

The results from X-Coms are combined with the original article data from the extract phase, excluding the content that no longer needs to be stored. The combined results are saved into files used in the subsequent load phase. This approach ensures a transparent and efficient processing workflow, which can be defined in task groups containing individual task dependencies, allowing for parallel data processing (see Figure 7.4).

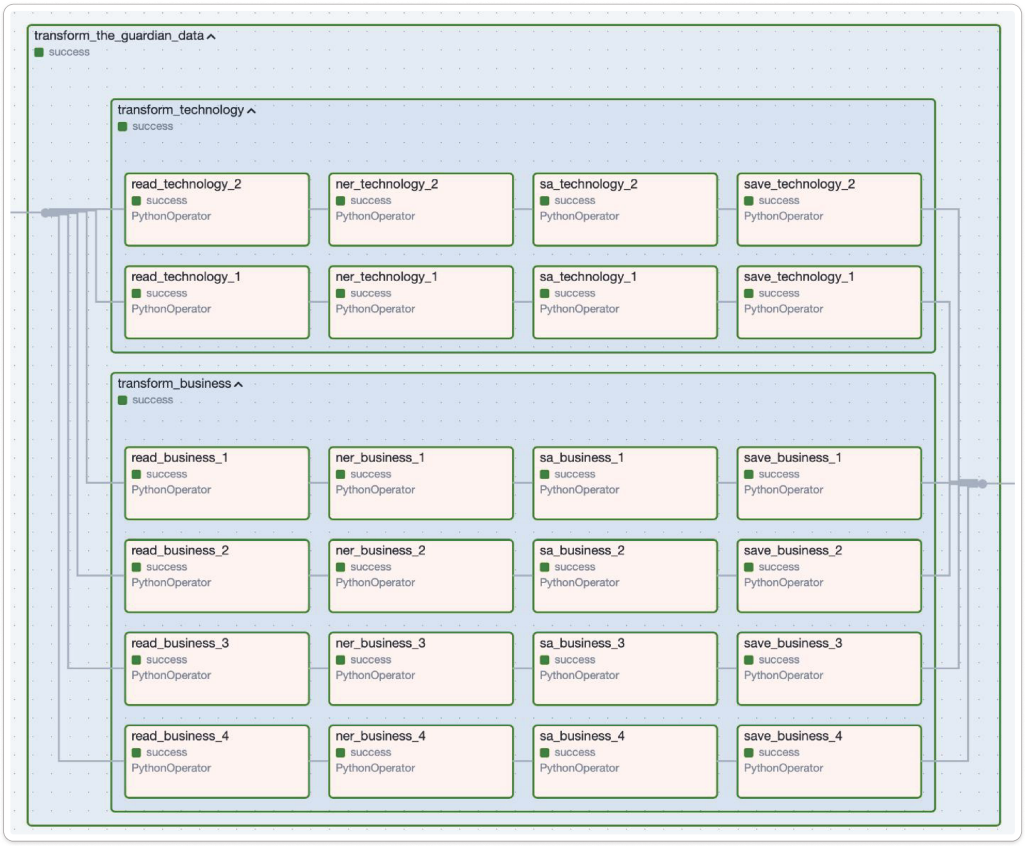



**Figure 7.4** The transform phase task groups and its tasks in the DAG of Guardian's ETL process. Directed from left to right.

### Load Details

The load phase begins with sequentially transformed data bulk loading into the database. In the current version, we do not employ loading in parallel because we have found that the approximate load duration time (see Table 7.5), in addition to a slowdown due to extra indexes, is acceptable, and parallel loading is unnecessary. It could lead to collisions where multiple sources simultaneously write to the same table. This approach could compromise data integrity and lead to database errors, especially when articles containing the same entities are inserted into the table concurrently. While row locking could mitigate this issue, it would slow down the entire process and increase system complexity, which could be counterproductive given the current data volume. Therefore, we prefer sequential data loading from a single source. The entire process of the Load phase is illustrated in Figure 7.5.

After loading data from individual task groups of sections, there is a task to insert additional company data into the database. This task involves inserting previously mentioned data such as short name, industry, and website from Yahoo Finance. Subsequently, there is a task that removes articles that do not contain any tickers. These articles are unusable to us, and their removal helps maintain data cleanliness in the database and reduces unnecessary database load.

| Task & Task Group | Duration [min] |
|---|---|
| replicated_and_clear_tables | 0.02 |
| insert_additional_data | 0.98 |
| delete_articles_without_companie | 0.02 |
| delete_replicated_tables | 0.02 |
| **extract_the_guardian_data** | 0.09 |
| **transform_the_guardian_data** | 44 |
| transform_business | 42 |
| transform_technology | 44 |
| **load_the_guardian_data** | 0.35 |
| load_technology | 0.08 |
| load_business | 0.22 |

**Table 7.1** The approximate duration of individual tasks and task groups in the ETL process for the Guardian source.

According to Tabel 7.1, the transformation of the technology section, even though it contains only two pages (files) of articles, is completed in a similar time as the business section, which is twice as extensive. This is due to the distribution of jobs among workers, of which we have 16 in our setup. The entire DAG work is automatically divided among the workers. However, the transformation results primarily depend on the response times of the NER and SA services. The

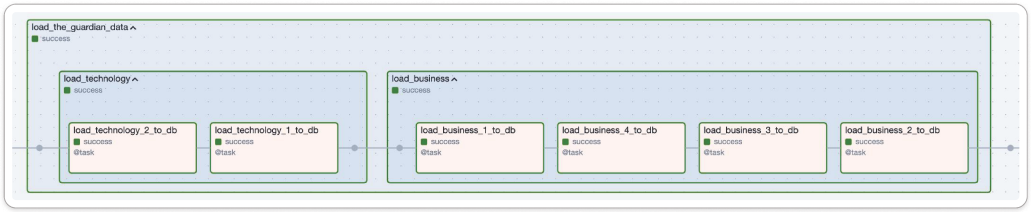



**Figure 7.5** The load phase task groups and its tasks in the DAG of Guardian's ETL process. Directed from left to right.

SA service, in particular, slows down the process owing to the computationally intensive FinABSA-Longer model, which has limited speed due to article content length. The average duration of NER task groups is approximately 4 minutes, while the average duration of SA task groups[5] is 33. The total runtime for a single execution of the entire pipeline, from extraction to loading and removing temporary tables, is approximately 45 minutes. In the following subsection, we will examine the construction of requested services. Lastly, we propose how the DAG distribution might look by adding two more sources, shown in Figure 7.6.

As mentioned, the time frame for extracting articles is set to three months. During this period, the database in the technology and business sections typically contains approximately the number of records indicated in Table 7.2.

| Table | Records |
|---|---|
| *articles* | 543 |
| *companies* | 180 |
| *sections* | 2 |
| *sentiments* | 1,327 |
| *article_companies* | 1,327 |
| *article_sections* | 543 |

**Table 7.2** The approximate number of records in the database tables after three months of time frame extraction employs the Guardian as the source focusing on business and technology sections.

### 7.2.2 Named Entity Recognition Service

The NER service categorises entities in article content data. Its core component is the named entity recognition algorithm, detailed in Chapter 4. We empha-

[5]NER task group in one task instance processes all 200 articles, whereas SA processes only those that contain a ticker after NER is finished.

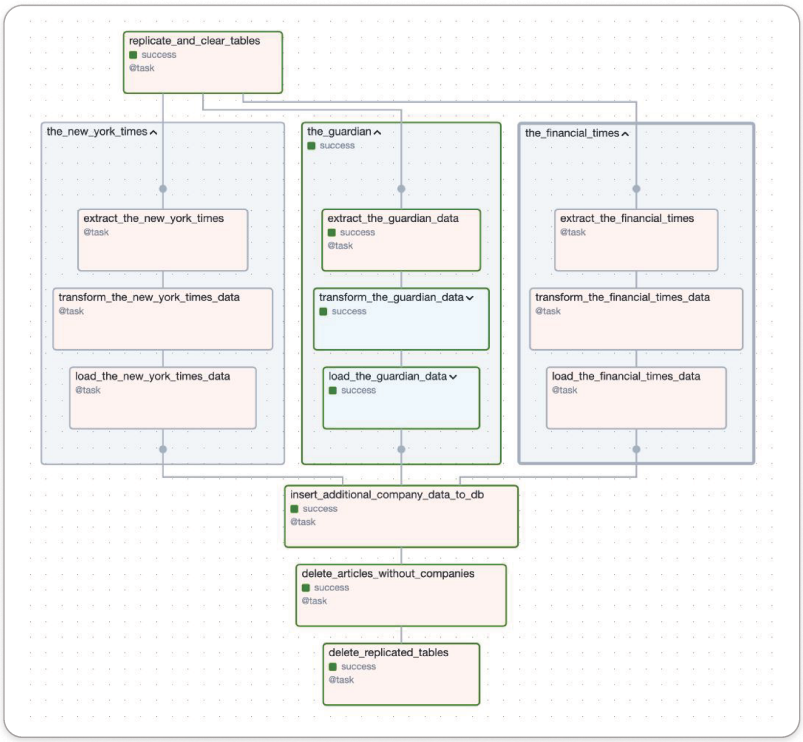



**Figure 7.6** The DAG of ETL phases for multiple sources, including the Guardian, New York Times, and Financial Times.

sise identifying organisation-type entities and filtering to determine companies[6] specifically within the algorithm. This service plays a crucial role in subsequent data processing, as the entities obtained serve as input for the SA service.

This service operates as a FastAPI[7] server, offering a POST request endpoint to extract entities from article content. The output includes a list of entities containing a ticker symbol and a substring representing the entity in the text. Especialy, SA service then processes substring further in analysis. Upon requesting this endpoint, results are returned to the ETL process during its transformation phase before continuing to the SA service. The following Figure 7.7 shows the schema of a POST request to the NER service for extracting entities from article content and its response.

The need for the fastest response times and efficient data handling drove the choice of the FastAPI framework. Its speed, user-friendly nature, and broad user community are the main reasons for its selection. It integrates seamlessly

[6]Reminder that organisations can be considered potential companies. An organisation is defined as a company if it has a ticker.

[7]https://www.fastapi.tiangolo.com

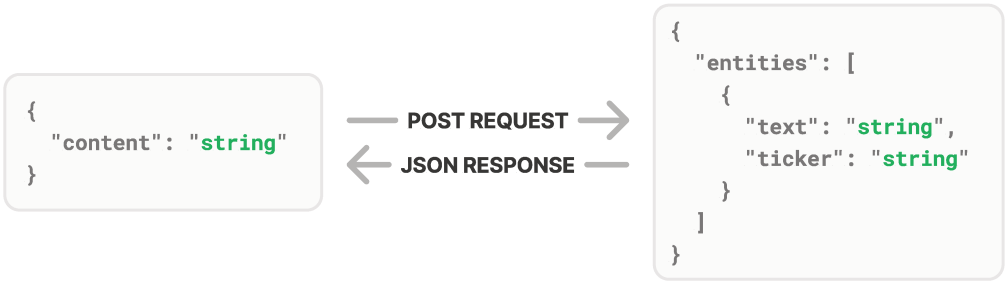



**Figure 7.7** Schema of a POST request to the NER service for extracting entities from article content and its response.

with the Pydantic library[8], allowing data to be defined as Python classes that automatically generate JSON schemas for data validation within the FastAPI framework. Moreover, FastAPI is built on the Asynchronous Server Gateway Interface[9] (ASGI) framework, which is suitable for our querying of Wikidata. The asynchronous implementation of the code allows operations to run independently of the server's main thread, enabling the server to handle more requests and perform other tasks while waiting for a response or processing data retrieved from Wikidata. This is highly desirable for our ETL pipeline, where processing large amounts of data is necessary while maintaining processing speed. Due to the server's limitations on which the NER service runs, the endpoint is handled by two workers for request processing.

### 7.2.3 Sentiment Analysis Service

The SA service analyses the sentiment of text at the entity level, classifying it as positive, negative, or neutral based on the highest sentiment score in one of these categories. The core component of this service is a sentiment analysis algorithm, which is detailed in Chapter 5. This service is crucial for determining the sentiment of entities extracted from the article content by the NER service.

This service operates, like the NER service, as an independent FastAPI server, offering a post request endpoint to analyse the sentiment of entities. The output includes a sentiment score for all categories and a classification for each company in the text. The sentiment scores are a floating-point number between 0 and 1, and their sum is equal to 1. The higher the score, the more the given category is expressed. Thus, sentiment for a given company has a structure that includes classification, positive, neutral, and negative values. The following Figure 7.8

[8]https://docs.pydantic.dev/
[9]https://asgi.readthedocs.io/en/

shows the schema of a post request to the SA service for analysing the sentiment of entities in article content and its response.

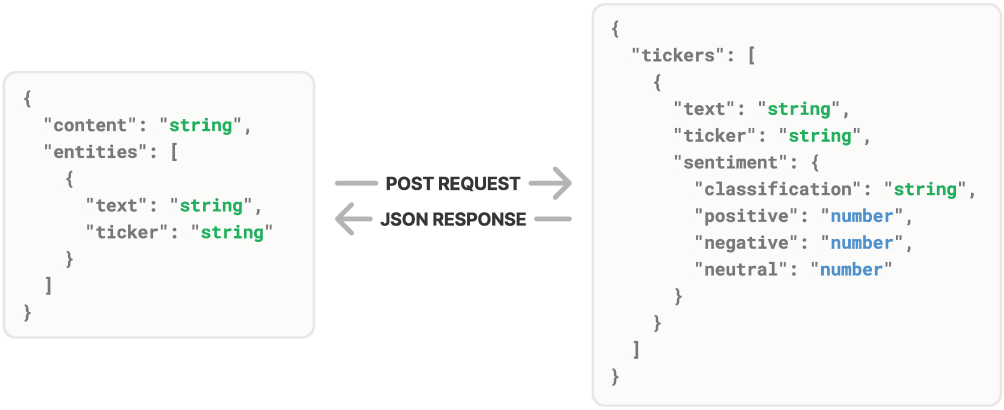



**Figure 7.8** Schema of a POST request to the SA service for sentiment analysis of extracted entities in article content and its reponse.

Like the NER service built on ASGI, the FastAPI framework was also selected for its speed and efficiency. Being built on ASGI enables FastAPI to be implemented asynchronously, which is beneficial for processing data from the NER service. Despite sentiment analysis demanding more computational power, two workers also manage the endpoint. The interval we have set for repeating the DAG is 4 hours, sufficient for processing speed, meaning processing takes 49 minutes and repeats every 4 hours.

### 7.2.4 REST API Service

The REST API service primarily provides access to data stored in the database. It is responsible for handling requests from the frontend and returning the appropriate data to be displayed to users. This service uses Flask[10], a lightweight Web Server Gateway Interface[11] (WSGI) web application microframework. This framework is chosen for its flexibility, simplicity, and ease of extensibility. Most importantly, it provides more satisfactory control and management of schemas and resources than FastAPI, which is crucial for the purposes of this API.

The API handles the connection to the database using a controller, which grants access to all data from the ETL process. It scans whether temporary tables are available in the database during each request, which exists only during the

[10]https://www.flask.palletsprojects.com/en/3.0.x/
[11]https://wsgi.readthedocs.io/en/

ETL process. Furthermore, since the main tables are empty during ETL, it queries the replicated temporary tables. After the ETL process ends, these tables are removed, and the API accesses the main tables containing data after the recent ETL. The API is designed to handle frontend requests and provide responses with data in JSON format. Thus, the output is a JSON file containing data from the database, which is then displayed to users.

The API provides the following endpoints, which we will not specify in detail with images like the previous two services but only summarise. We refer directly to the Swagger documentation, which we specify in more detail in Chapter 8, for a detailed description of schemas and resources. Here is a summary of the endpoints, all of which are in the form of GET requests:

- **Company Info** fetches a company's information from the database, as well as daily high and low prices and market volume from Yahoo Finance.
  - */api/v0/company/{ticker}/info*
- **Company Chart** fetches a company's chart data, such as sentiment, from the database and adjusted close price from Yahoo Finance to be displayed in chart.
  - */api/v0/company/{ticker}/chart*
- **Company Article List** fetches a company's articles from the database.
  - */api/v0/company/{ticker}/articles*
- **Company Graph** fetches a company's graph data from the database to be displayed as a graph network.
  - */api/v0/company/{ticker}/graph*
- **Companies Names and Tickers** fetches all companies' names and tickers from the database.
  - */api/v0/companies/names*
- **Companies Graphs** fetches all companies' graphs from the database to be displayed as a graph network.
  - */api/v0/companies/graphs*

## 7.3 Frontend

The frontend is written using the Angular framework. Its principal function is to provide a user interface for interacting with data from the database through a REST API. In our case, it communicates solely with the backend REST API to retrieve and display data without making any updates. Built on top of TypeScript, Angular adds static typing and other features that simplify the development and implementation of the code. It offers many features, such as a modular architecture, which makes it easy to split an application into smaller parts, including components, services, and routing.

Although Angular is a frontend framework, infrastructure such as Docker or other server software is required to run it on a server. This infrastructure is used to deploy and serve the frontend application to users. Deploying an Angular application requires a server that provides users with static files. It is also built for single-page applications, which is an ideal solution in our case because the application can be more efficient, easily scalable, and maintainable, which is essential for long-term development. As part of the architecture and communication with other services, we create models in the form of interfaces that can be directly mapped to individual endpoints in the REST API, which we can query its endpoints through an HTTP service.

The most significant benefit of Angular is its component architecture. The cornerstone is the component, composed of a TypeScript class, an HTML template, and styles typically in CSS. Components are reusable and can be easily moved to other application parts. This makes deploying new components smooth and effortless, increasing development efficiency. The flowchart navigation of application pages is illustrated in Figure 7.9 below for a better understanding of the frontend structure. The following subsections describe each application component.

### 7.3.1 Home Component

This component serves as the homepage and does not contain any other components. It only displays an introduction describing the application. It includes a navigation bar, allowing users to access other application pages, such as graphs or companies.

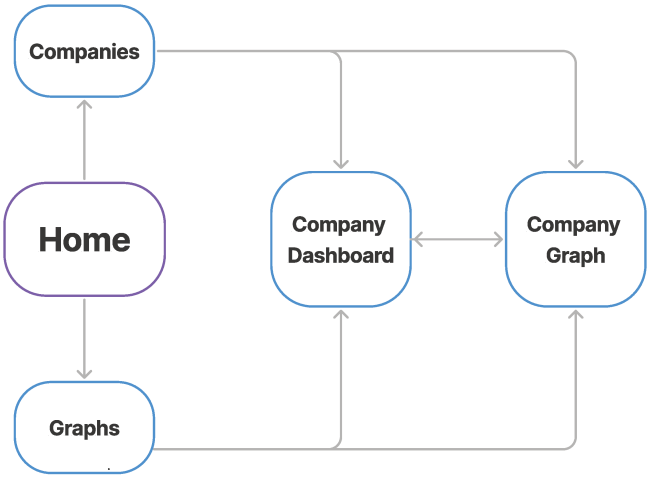



**Figure 7.9** Flowchart navigation of application pages in the frontend.

### 7.3.2 Graphs Component

This component represents a page that displays the connections between individual articles and companies. The 3D Force Graph library, available on GitHub[12], provides the graph visualisation web component based on ThreeJS[13]/WebGL for 3D rendering and d3-force-3d[14] or ngraph[15] for the core physics engine. An edge links an article to a company if the company is mentioned in the article. The edge colour represents the sentiment towards the company in the article. The colour of the articles varies based on their average sentiment towards all mentioned companies, while the company's colour is blue to distinguish it from the articles. Additionally, the edge colour indicates the sentiment associated with the specific company.

The graph can be navigated and displayed, with control panels included for manipulating and modifying the graph visualisation. Users can filter the graph based on the average sentiment of individual articles and adjust the graph size to modify the visualisation. Each article can be viewed on the Guardian or navigated to the company graph or dashboard. The higher the mentioned companies within the article, the higher the article node size. The same applies to the company node. The higher mentioned in articles, the higher the company node size. We can smoothly identify the most trending companies in the articles. The interactive

[12]https://www.github.com/vasturiano/3d-force-graph
[13]https://www.github.com/mrdoob/three.js/
[14]https://www.github.com/vasturiano/d3-force-3d
[15]https://www.github.com/anvaka/ngraph.forcelayout3d

graph allows users to zoom in and out, rotate, and drag nodes. Additionally, graph traversal simulating the visibility of only neighbouring vertices is also implemented. The graph is loaded from the REST API companies graphs endpoint.

### 7.3.3 Companies Component

This component represents a page for searching individual companies' dashboards or graphs. It allows users to search for companies by name or ticker, implemented through Angular's ngx-pipes for efficient data manipulation and sorting by default. After searching, navigating to the selected company's dashboard or company graph component is possible. Like the home component, it also includes a navigation bar from which users can navigate to the graphs or home page. Data are loaded from the REST API companies names and tickers enpoint.

### 7.3.4 CompanyGraph Component

This component serves as a page, similar to the graphs component, but offers a slightly different view of the connections between individual articles and a specific company. The purpose is to reflect sentiment only to the individual company. Colouring in the graph works the same way as before. However, in this case, the article's node colours are based solely on the sentiment towards the specific company and not on the average sentiment of all the companies mentioned. Additionally, it contains control panels similar to those of the graph component. The graph is loaded from the REST API company graph endpoint.

### 7.3.5 CompanyDashboard Component

This component acts as a parent component to load individual child components. As the parent component, it loads data and then passes it to its children through the input decorator. This way, all data transfer is handled in one component, which then distributes it for loading. The child components of the company dashboard are as follows:

- **CompanyInfo Component**: Displays brief information about the company.
- **Charts**: Individual charts are created using the CanvasJS[16] library, which provides simple and efficient implementation within the Angular framework. Data for the charts is loaded from the REST API company chart endpoint. The following charts are included:

[16]https://www.canvasjs.com

  - **StockChart Component**: Displays sentiment and adjusted close price in a stock chart average seniment for each day.
  - **PieChart Component**: Shows the distribution of average sentiment over time in a pie chart.
  - **SplineChart Component**: Illustrates each category of sentiment over time in a spline chart.

- **ArticleList Component**: Lists all articles mentioning the company, including relevant data such as title, date, link to the article, author, section, and sentiment. Data can be sorted and searched using text that applies to data in the table across all attributes. The listing is implemented through the material paginator provided by Angular. Data are loaded from the REST API company article list endpoint.

By utilising the input decorator, data for all three charts is loaded only once, allowing us to clearly and efficiently create at least three charts without loading data separately for each chart. Additionally, data loaded into the CompanyInfo and ArticleList components come from specific endpoints explicitly created for their purpose. This approach allows us to effortlessly and efficiently implement additional components we want to include in the dashboard in the future.

# 8. Development Documentation

Since the algorithms and the concepts of named entity recognition and sentiment analysis applied to each service have been thoroughly described in Chapters 4, 5, and the architecture in Chapter 7, this chapter will only briefly summarize the essential points. For more detailed information, we will refer to the documentations produced by pdoc[1]. This documentation includes extensive functional and module comments and adheres to the PEP8[2] standard. Additionally, Chapter 7 thoroughly describes the ETL process, covering the various steps required to obtain and prepare the data for subsequent analysis. The frontend documentation has not been generated, so we will discuss it in more depth. However, it also contains strong enough comments to describe the functionality.

Our deployment details are not included in this text because the application is available on a server provided by the university. Unified services run on the server, each handled by Docker to ensure independent functionality. Only two server ports are tunnelled, as we only need one to manage ETL service through the Airflow webserver and one to provide access to the frontend. The API services have their OpenAPI specifications in JSON generated by Swagger[3], which can be displayed within Swagger EDITOR[4]. Docker images are available on Docker Hub[5].

## 8.1 Backend

### 8.1.1 Extract Transform Load Service

The extract transform load service is located in the directory */app/backend/extract-transform-load*.

- **Documentation**: */docs/documentation/dev/etl*.
- **Airflow Webserver** is availible on server[6], use Viewer account
  - **login**: *viewer*

[1] https://pdoc3.github.io/pdoc/
[2] https://peps.python.org/pep-0008/
[3] https://swagger.io
[4] https://editor.swagger.io
[5] https://hub.docker.com/repositories/stiborv
[6] Contact me for more information.

  - **password**: *viewer*

- **Main Directories**

  - ***/airflow*** - contains the Airflow DAG loading script to the Airflow and the configuration files
  - ***/dag*** - contains the ETL's DAG, detailed in attached documentation
  - ***/data*** - contains the ETL phases data checkpoints
  - ***/scripts*** - contains Docker scripts for the Airflow and Postgres initialisation.

### 8.1.2 Named Entity Recognition Service

The named entity recognition service is located in the directory */app/backend/-namedentity-recognition*.

- **Documentation**: */docs/documentation/dev/ner*.

  - **OpenAPI**: *openapi.json*

- **Main Directories** are detailed described in the documentation.

- **Spacy Model**: *en_core_web_md*

- **Spacy Entity Linker Model**: version 1.0.3

### 8.1.3 Sentiment Analysis Service

The sentiment analysis service is located in the directory */app/backend/sentiment-analysis*.

- **Documentation**: */docs/documentation/dev/sentiment*.

  - **OpenAPI**: *openapi.json*

- **Main Directories** are detailed described in the documentation.

- **Analysis Model**: *amphora/FinABSA-Longer*

## 8.2 REST API Service

The REST API service is located in the directory */app/backend/rest-api*.

- **Documentation**: */docs/documentation/dev/rest-api*.
  - **OpenAPI**: *openapi.json*
- **Main Directories** are described in the documentation, within resources and schemata, and in OpenAPI documentation.
-

## 8.3 Frontend

The frontend service is located in the directory */app/frontend*. The frontend components are structured in the following way:

### 8.3.1 Components Overview

The frontend components are structured in the following way:

#### HomeComponent

The *HomeComponent* is a simple component with no data mapping. It is dedicated to displaying the home page.

#### CompaniesComponent

The *CompaniesComponent* is dedicated to searching for companies using the *CompanyName* model as a list, which is directly mapped to data from the REST API endpoint */api/v0/companies/names*.

#### DashboardComponent

The *DashboardComponent* is a parent component fetching data for its child components. While data are fetching for each child component separately, the data loading bar is displayed within the separate children. The child components are as follows:

- **ArticleListComponent** - component dedicated to displaying all articles as a list. It is directly mapped to data from the REST API endpoint */api/v0/company/-{ticker}/articles* using the *CompanyArticlesList* model.

- **CompanyInfoComponent** - component dedicated to displaying company information. It is directly mapped to data from the REST:
  - **Endpoint**: */api/v0/company/{ticker}/info*
  - **Model**: *CompanyInfo*
- **PieChartComponent** - component dedicated to displaying the company's daily average sentiment value distribution in a pie chart. It is directly mapped to data from the REST API :
  - **Endpoint**: */api/v0/company/ticker/chart*
  - **Model**: *CompanyChart*
- **SplineChartComponent** - component displaying company sentiment value evolution in a spline chart. It is directly mapped to data from the REST API:
  - **Endpoint**: */api/v0/company/{ticker}/chart*
  - **Model**: *CompanyChart*
- **StockChartComponent** - component dedicated to displaying company price and sentiment. It is directly mapped to data from the REST API:
  - **Endpoint**: */api/v0/company/{ticker}/chart*
  - **Model**: *CompanyChart*

#### GraphsComponent

The *GraphsComponent* is a component dedicated to displaying all companies and articles as one graph. The components are separated into *Graph* and *Graphs* due to the ability to customise the graph and display the data so that the user can easily understand the data. It is directly mapped to data from the REST API:

- **Endpoint**: */api/v0/companies/graphs*
- **Model**: *CompaniesGraphs*

### 8.3.2 Services Overview

The frontend services provide communication with the REST API. Using the *HttpClient* module requesting GET methods to the endpoints using dedicated models to ensure data consistency.

### 8.3.3 Routes

The frontend routes manage navigation through the application. It uses the *RouterModule* to navigate between components. If *ticker* is in the URL, the application will automatically load the company data related to the company with the given *ticker*.

- **HomeComponent**: / or */home*
- **CompaniesComponent**: */companies*
- **GraphsComponent**: */companies/graphs*
- **GraphComponent**: */company/:ticker/graph*
- **DashboardComponent**: */company/:tickerdashboard*

# 9. User Documentation

## 9.1 Home

The home page is the initial page the users see after visiting the website (see Figure 9.1). Users can navigate to the graphs page, which presents a comprehensive network of articles spanning the last three months. Additionally, they have access to the companies page, which provides a detailed listing of companies.

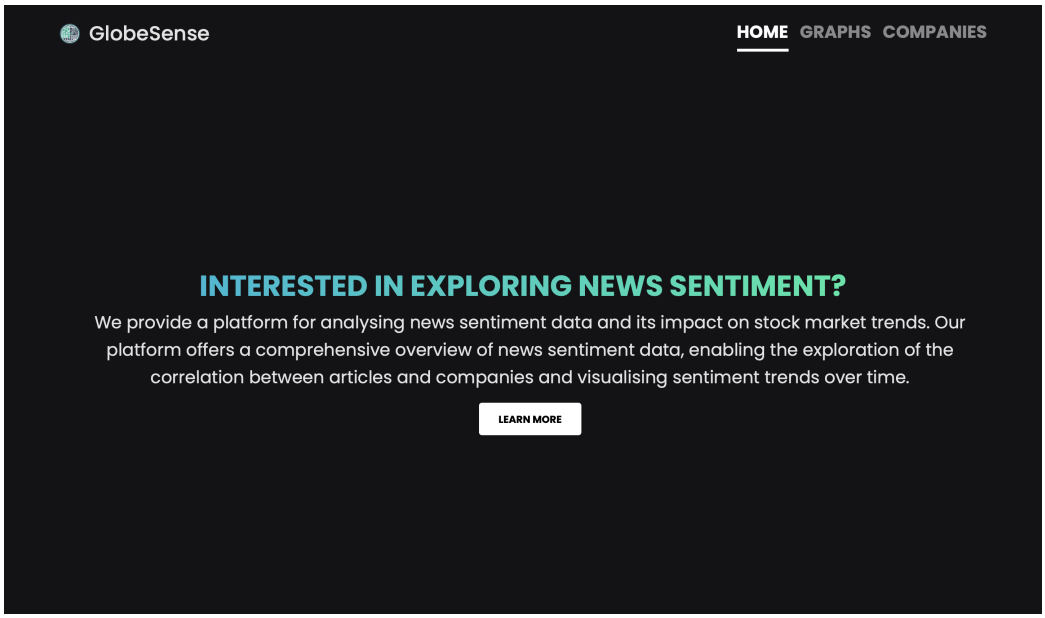



**Figure 9.1** The website's home page briefly describing the project and its purpose.

## 9.2 Graphs

The graphs page displays a network of all articles from the past three months. Each edge represents the sentiment of the company mentioned in the article, indicated by its colour. The article node's colour represents the average sentiment of all connected companies. Hovering over a node reveals its name if it is a company or its title if it is an article (see Figure 9.2). Furthermore, neighbouring nodes are highlighted.

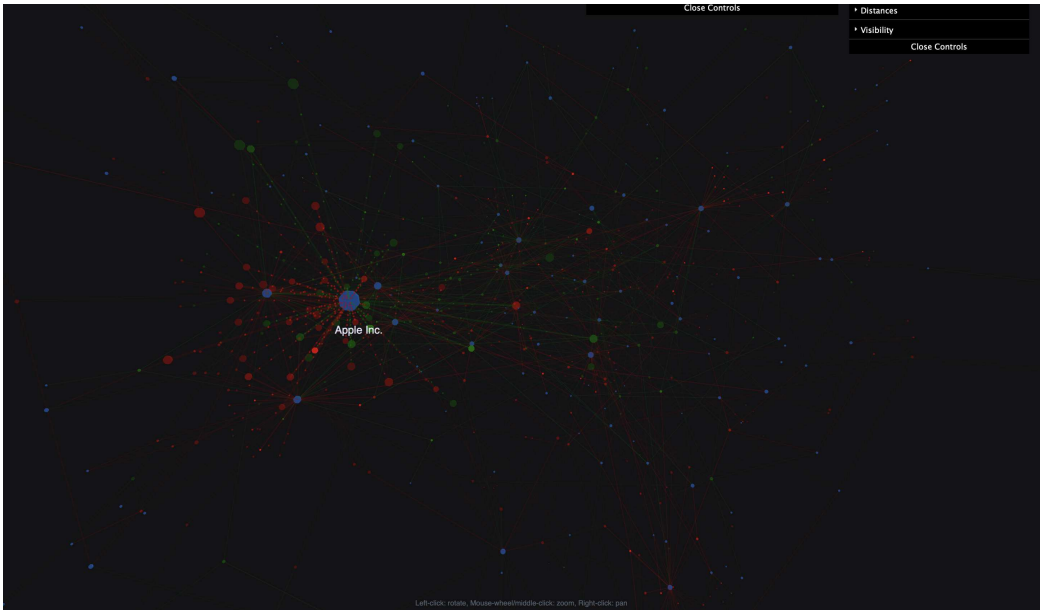



**Figure 9.2** Users can see the network of all articles for the past three months by navigating to the graphs page.

### 9.2.1 Control Panels

The control panels on the upper right side of the page are categorized into two types. The first type is the general control panel, specifically designed for managing the network graphs and encompasses the following features:

- **Distances**
  - **Positive distance** - regulates the distance of the edge with positive sentiment.
  - **Neutral distance** - regulates the distance of the edge with neutral sentiment.
  - **Negative distance** - regulates the distance of the edge with negative sentiment (see Figure 9.3).
- **Visibility**[1]
  - **Show only negative sentiment** - shows only the article nodes with negative sentiment (see Figure 9.4).

[1]Visibility filtering is not available when only neighbouring nodes are visible. Refer to its corresponding company graph page for specific node examination and filtering.

  - **Show only neutral sentiment** - shows only the article nodes with neutral sentiment.
  - **Show only positive sentiment** - shows only the article nodes with positive sentiment.

Another control panel is available for node details, which appears upon left-clicking on an article node. It provides the following options:

- **Article node**
  - **Title** - the title of the article.
  - **Published date** - the date when the article was published.
  - **Author** - the author of the article.
  - **Open article** - the link to the article.
  - **Average sentiment** - the average sentiment of the article is based on the average sentiment of the companies mentioned.
  - **Companies** - the list of companies mentioned in the article with their sentiment. It expands on details about the company and its sentiment in the article (see Figure 9.5).
- **Company node**
  - **Company information** - the details about the company, such as ticker and link to the company's dashboard or graph page.
  - **Articles** - the list of articles in which the company is mentioned with the sentiment of the company in the article (see Figure 9.6).

### 9.2.2 Actions

- **Left click** on a node involves zoom-in, then it will show only the neighbours of the clicked node.
- **Right click** on a node will show all nodes back.
- **Hover node** shows a name of the company or the title of the article and highlights the neighbours.

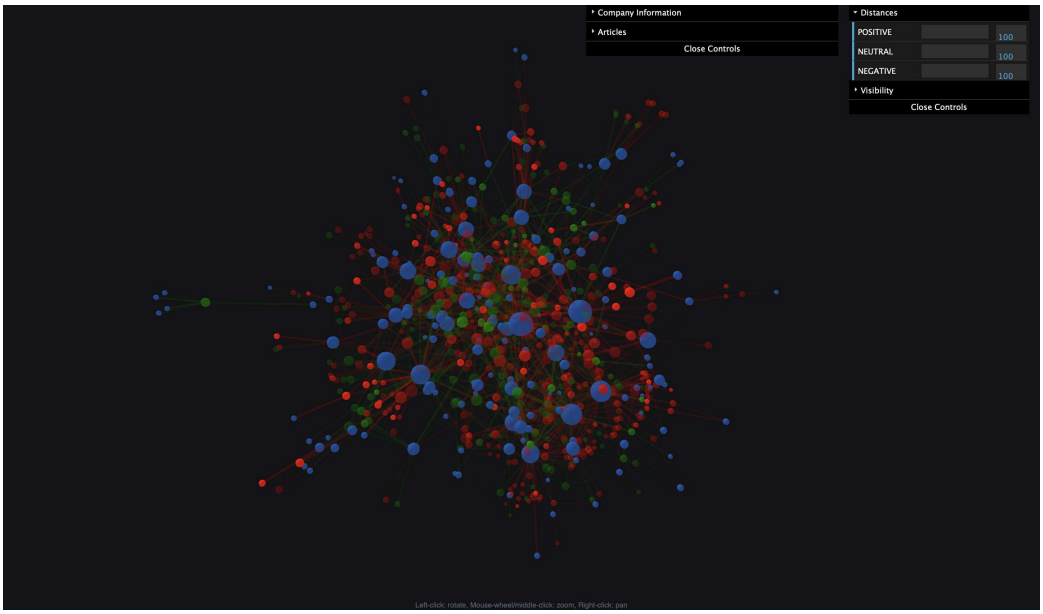



**Figure 9.3** The control panel managing edge distances. This involves selecting edges with sentiments and regulating them with values ranging from 100 to 3000. The image shows the network with the edge distances set to 100.

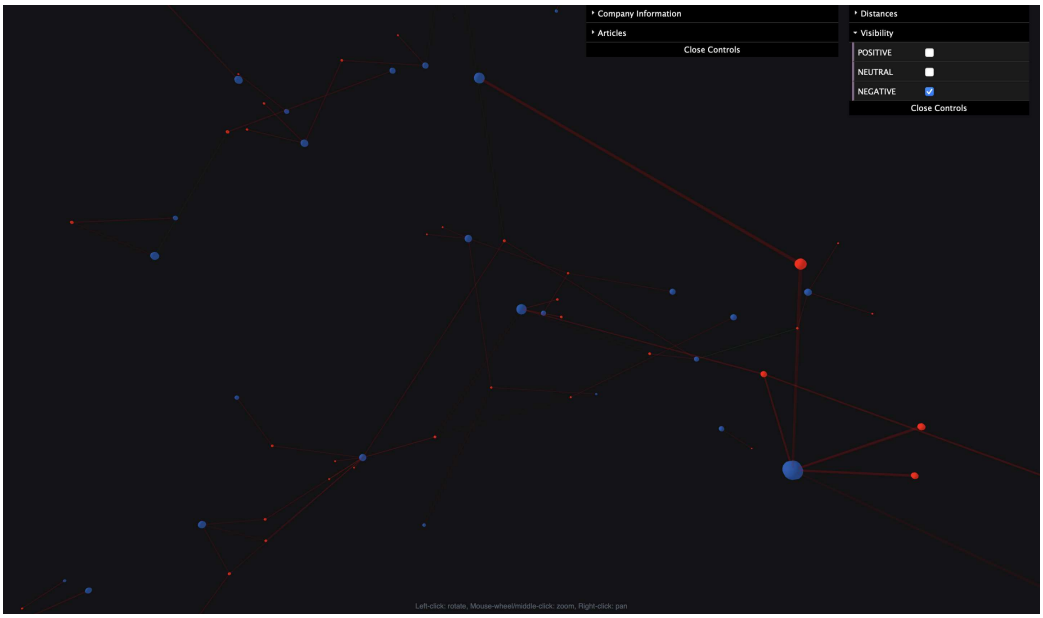



**Figure 9.4** After right-clicking, all nodes will be displayed back. The neutral and positive sentiments of the article nodes were turned off through the control panel. If there is an edge to the article node, then it remains to be visible.

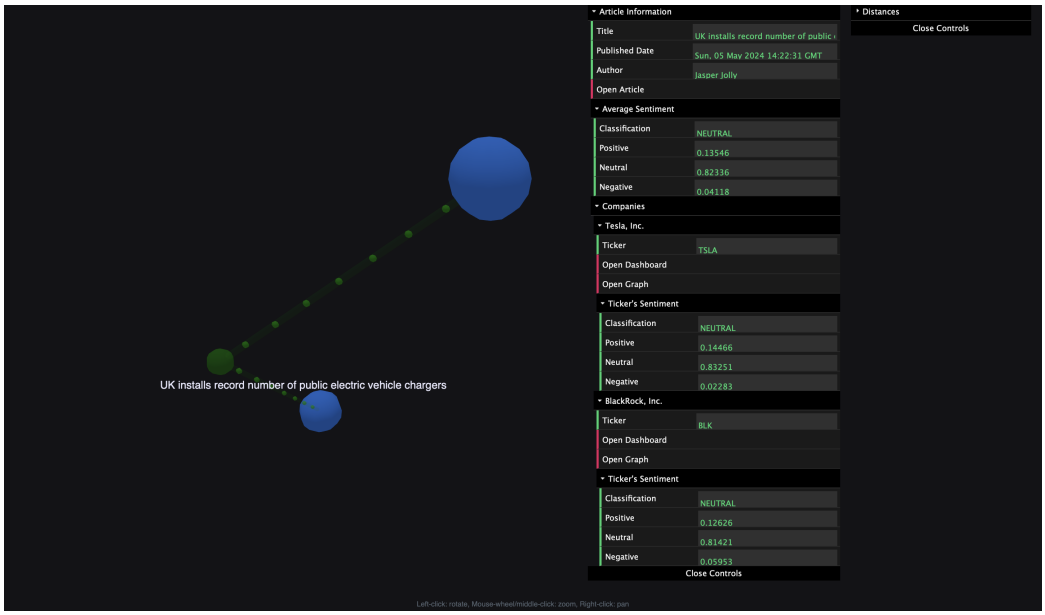



**Figure 9.5** After left-clicking on an article node, the control panel could be expanded with further information, such as listing all companies mentioned in the article with their sentiment as well as the average sentiment of the article and more detailed information about the title, date, author, and link to the article.

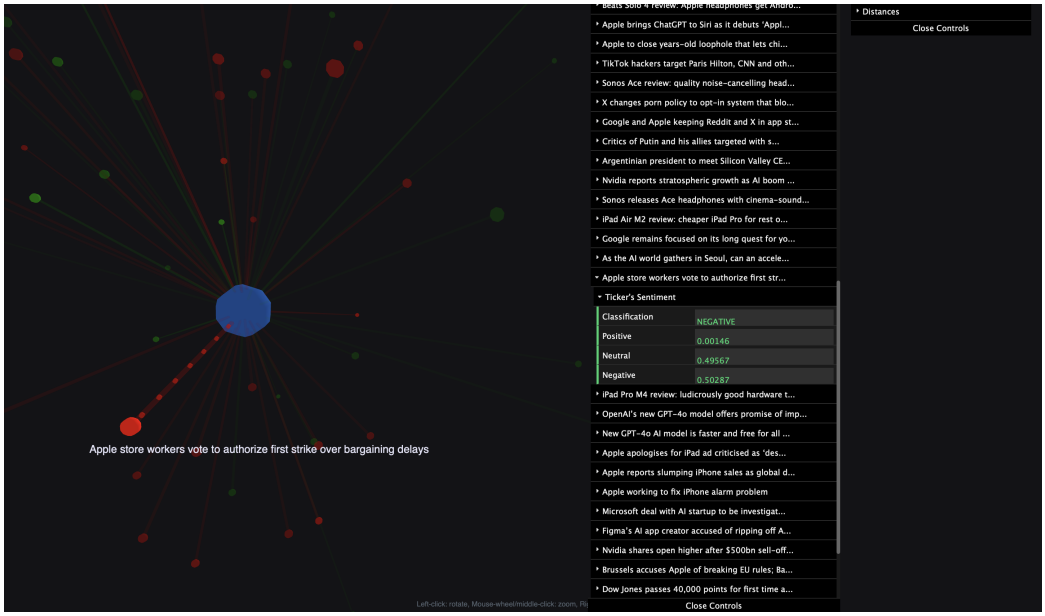



**Figure 9.6** Apple Inc. node with its neighbours after left click. The list of articles in which it is mentioned additionally provides the values of its sentiment, which is Apple's sentiment in that article. The article's colour indicates a negative average sentiment.

## 9.3 Companies

The companies page allows the users to search for a company by its name or stock symbol. The users can also navigate to the company's dashboard or graph page (see Figure 9.7).

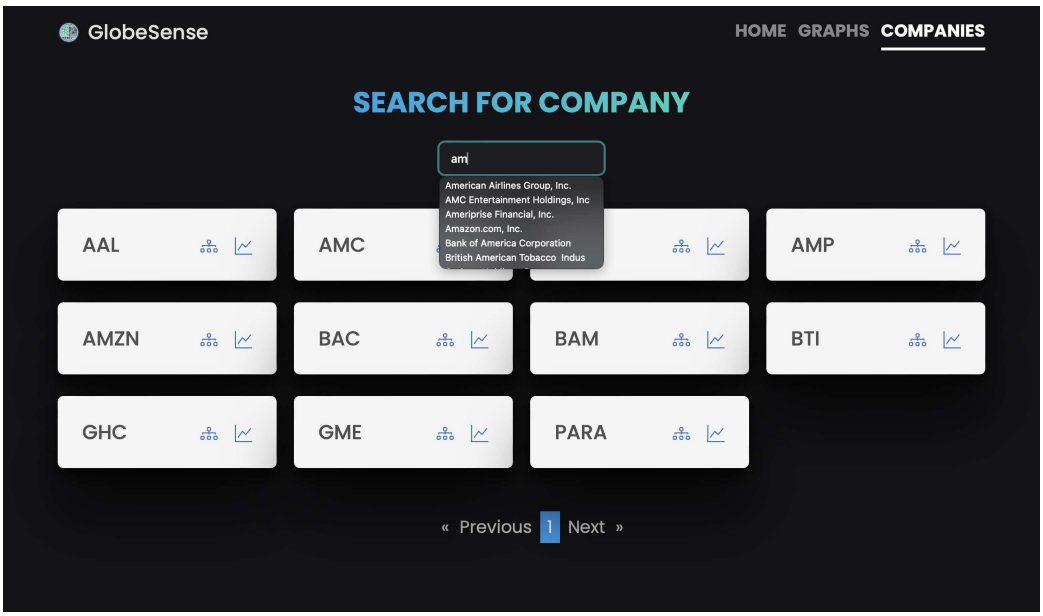



**Figure 9.7** Companies search allows the users to search for a company by its name or stock symbol. The image shows the search for "am".

## 9.4 Company Dashboard

The company dashboard offers users comprehensive information about the company through multiple components (see Figure 9.8).

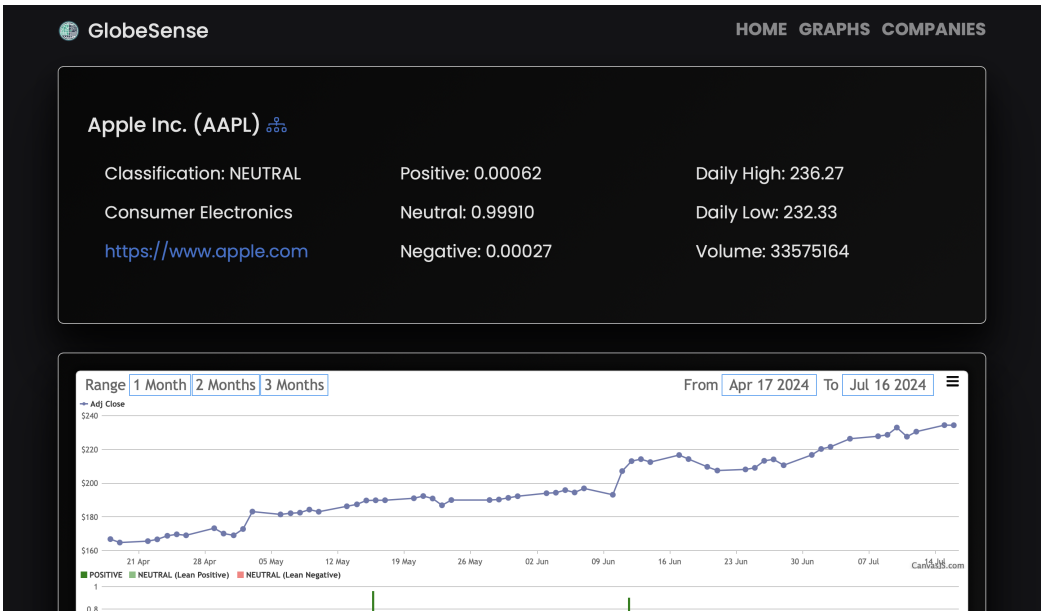



**Figure 9.8** Apple Inc. dashboard overview.

### 9.4.1 Company Information

The company information component includes the company's name, stock symbol, and a graph link. It provides average daily sentiment values categorised as positive, neutral, or negative, updated every 4 hours. Additionally, it displays Yahoo Finance data, including daily high, low, and volume, which is refreshed with each visit (see Figure 9.9).

### 9.4.2 Stock Chart

The stock chart component shows the adjusted close price of the company for the past three months. The users can zoom in and out by selecting the time range (see Figure 9.10). The chart is divided into two parts; the upper part shows the price, and the lower part shows the daily average value of the company's sentiments in the articles for a given day.


**Figure 9.9** Apple Inc. company information.

Sentiments in the second part can be hidden by clicking on the sentiment legend. Hovering over the chart shows the date, stock price, and sentiment value for a given day (see Figure 9.11).


**Figure 9.10** Apple Inc. stock chart in default view.

### 9.4.3 Spline Chart

The spline chart component demonstrates the evolution of the daily average sentiment value over the past three months. Users can hide the sentiment spline like in the stock chart by clicking on the sentiment legend. Hovering over the chart shows a given day's date and sentiment value (see Figure 9.12).


**Figure 9.11** Apple Inc. stock chart in two months range with hidden sentiment values except neutral (lean negative).

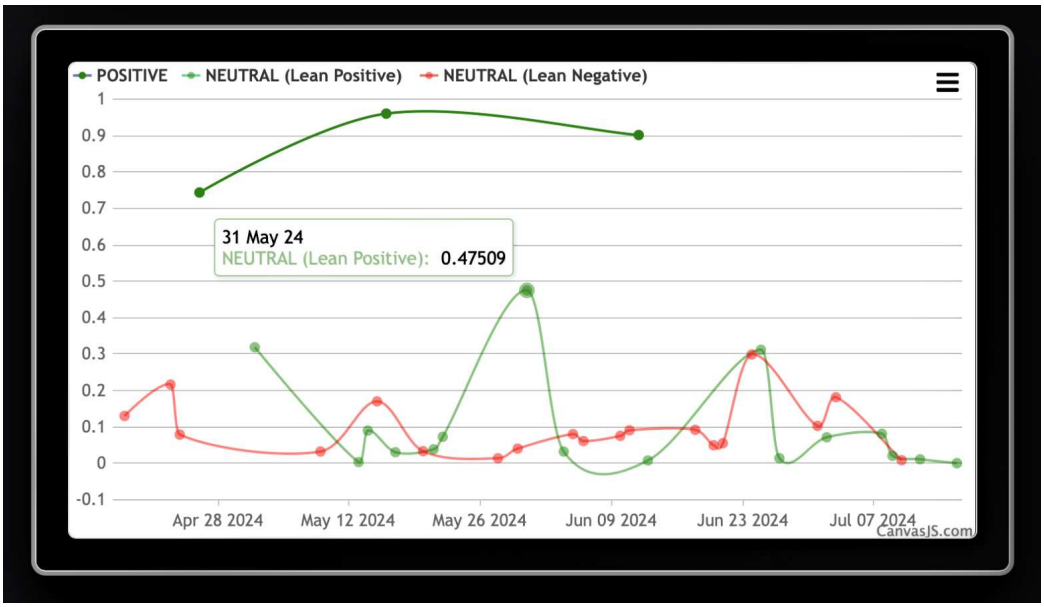



**Figure 9.12** Apple Inc. spline chart displaying the daily average sentiment value evolution for the past three months.

### 9.4.4 Pie Chart

The pie chart component illustrates the daily average sentiment values distribution for the past three months (see Figure 9.13).

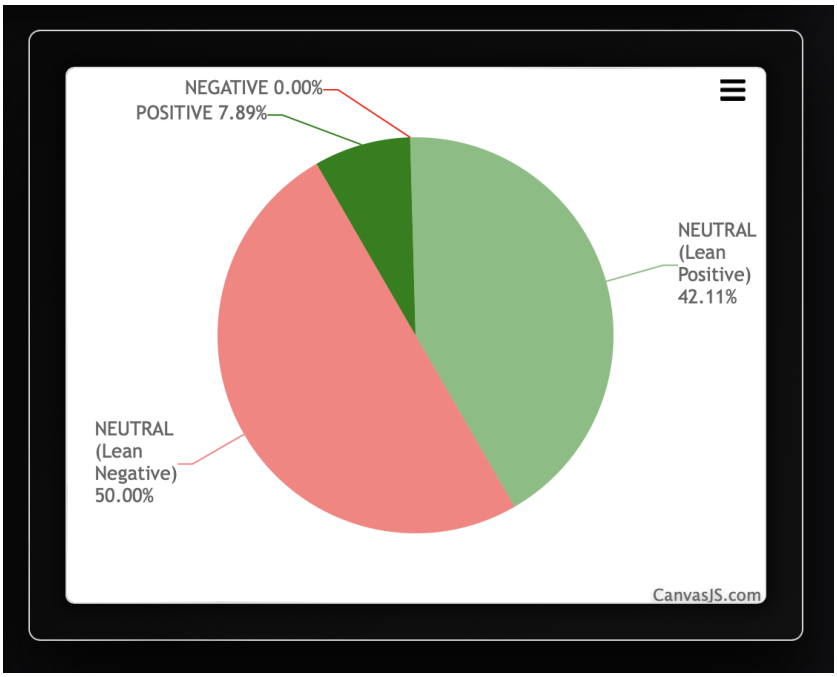



**Figure 9.13** Apple Inc. pie chart showing the daily average sentiment values distribution for the past three months.

### 9.4.5 Article List

The article list component shows the list of articles in which the company is mentioned, along with the sentiment of the company in the article. Users can navigate to the article page by clicking on the article source website.

It also provides the company's sentiment value in the article, which is the company's sentiment in that article. It is a classic pagination where users can navigate to the next or previous page. Overall, column ordering is allowed by clicking on the column name. Users can also filter the articles by the search terms in the search bar. Thus, the articles can be searched by title, author, type, specific sentiment classification, and more. See Figures 9.14, 9.15, and 9.16.

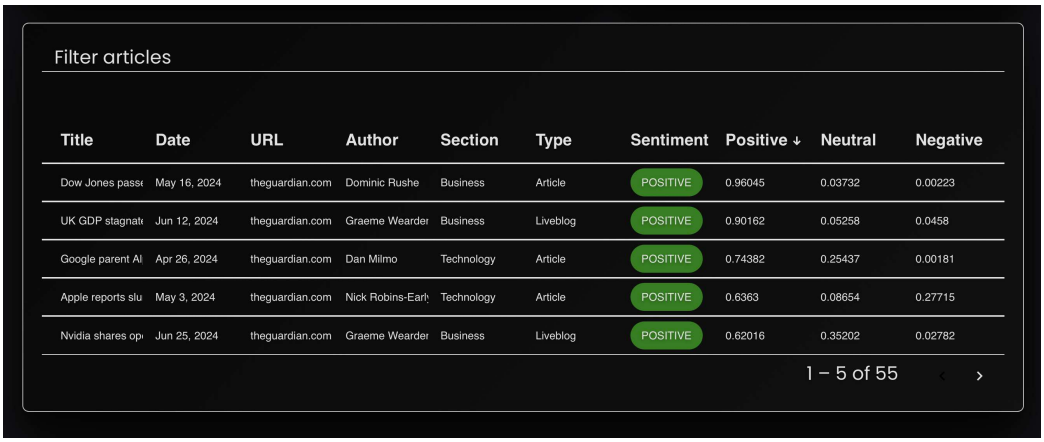



**Figure 9.14** Apple Inc. article list displaying the list of articles in which the company is mentioned with the sentiment of the company in the article in descending order by the positive sentiment value.

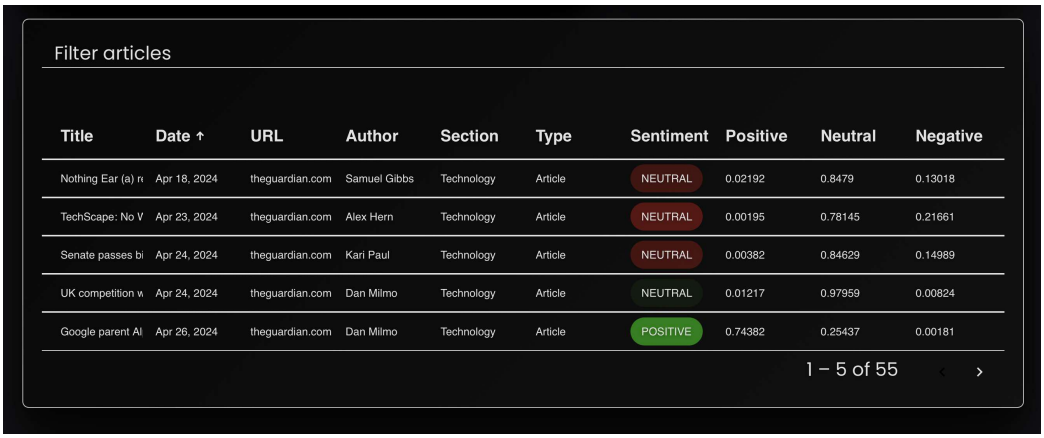



**Figure 9.15** Apple Inc. article list depicting the list of articles in which the company is mentioned with the sentiment of the company in the article in the ascending order by the date.

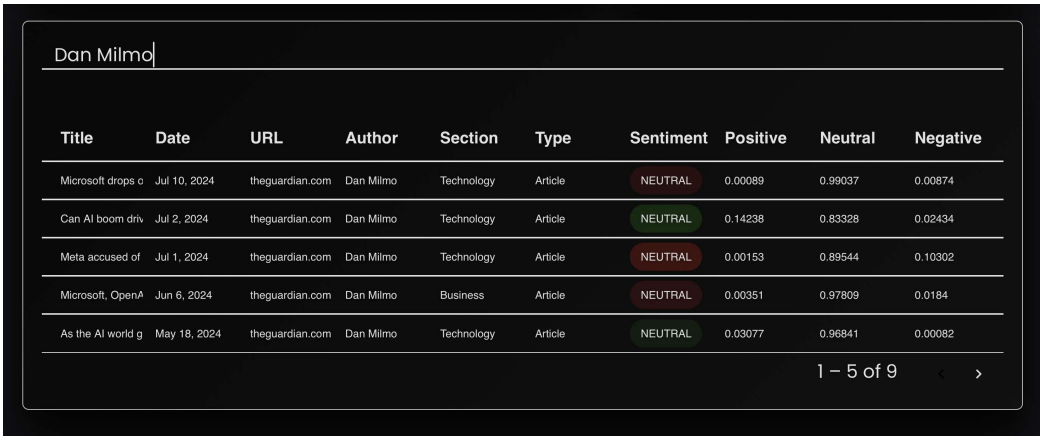



**Figure 9.16** Apple Inc. article list portraying the list of articles in which the company is mentioned with the sentiment of the company in the article filtered by the search term “Dan Milmo” to find all articles written by the author.

## 9.5 Company Graph

The company graph page shows the network of all articles in which the company has been mentioned for the past three months. The edge shows by its colour the sentiment of the company within the article, which node is coloured by the sentiment associated with the company, unlike companies graphs where the average sentiment of all associated companies colours articles. Actions are the same as on the general graph page.

Hovering over a node displays its name in the case of a company and its title in the case of an article. In this case, it highlights only the article and company node, including the edge, after hovering over the article. Hovering the company node just increases (highlights) its size but not relationships with all articles, which is unnecessary in this projection type. The user can navigate the article page by clicking the article node (see Figure 9.17).

The general control panel has a purpose similar to that of the graphs page. Visibility and distances remain the same. The company and article node control panel provides the following details:

- **Article node**
  - **Title** - the title of the article.
  - **Published date** - the date when the article was published.
  - **Author** - the author of the article.
  - **Open article** - the link to the article.

  - **Sentiment** - the sentiment of the company in the article.

- **Company node**
  - **Company information** - the details about the company, such as a ticker and link to the company's dashboard. Additionally, it presents the average daily sentiment of the company.
  - **Articles** - the list of articles in which the company is mentioned with the sentiment of the company in the article.


**Figure 9.17** Apple Inc. company graph displaying the network of all articles mentioned in the company for the past three months. The state after clicking on the company node and hovering over article node is illustrated.

# Conclusion

This thesis aimed to provide users with a tool that allows them to monitor and analyse the flow of information about emotional overtones as a fundamental aspect of market decisions. This objective was achieved by developing an application that extracts data from the Guardian, a British daily newspaper, analyses the sentiment of the extracted entities and provides this data as an indicator of potential future influences on a company's stock price. Acquiring reliable data made the development demanding, as obtaining data directly from providers rather than third-party sources is more challenging.

We investigated the most suitable approach for extracting company entities and their tickers. Building on the Spacy Entity Linker library, we demonstrated a flexible method for extracting information regarding a given entity. Using our approach to entity recognition and the subsequent employment of the FinABSA model, we achieved a $92\%$ success rate in sentiment evaluation when testing the reduced FinEntity dataset, primarily due to FinABSA. We then provided a short demonstration of using sentiment within the stock market.

Users have continuous access to ensure reliable data results, even during updates and processing. The application prioritises accuracy and reliability by minimising data noise. The application also allows users to verify and test historical data to see how our algorithms evaluated previous sentiment data and how the market reacted, helping them consider the values our application will evaluate. Additionally, it is adaptable to various sources, expanding the range of companies it can cover.

We designed graph network visualisation, allowing users to visualise connections between companies and news articles. The impact of news sentiment on a company's stock price is also visualised within the stock chart, supported by spline and pie charts displaying additional analyses. The application is designed to be user-friendly and intuitive, allowing users to navigate to the articles and perform their analysis to verify the values provided by the application. It is also organised with a modular architecture, enabling the manageable integration of new analysis components or models through its multi-service structure. Both of the application's leading algorithms were tested on historical data.

In addition, we acknowledge that the sentiment analysis algorithm could be faster, but this is primarily due to the FinABSA model, which takes much time due to the length of textual data. However, this is currently sufficient as the model evaluates the data with impressive accuracy, and there is enough time between ETL intervals to execute it, allowing us to provide results anytime. New

types of data sources, such as social media, could be integrated to gain additional sentiment insights, or other text data sources covering publicly traded companies could be included.

# Bibliography


Piryani, R. et al. (2017). *Analytical mapping of opinion mining and sentiment analysis research during 2000–2015*. In: *Information Processing & Management* 53.1, pp. 122–150. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2016.07.001. URL: https://www.sciencedirect.com/science/article/pii/S030645731630245X.

Saunders, Danielle (2020). *Domain adaptation for neural machine translation*. PhD thesis. Apollo - University of Cambridge Repository. DOI: 10.17863/CAM.66458. URL: https://www.repository.cam.ac.uk/handle/1810/319335.

Liu, Bing (2022). *Sentiment analysis and opinion mining*. In: Springer Nature. Chap. 3, pp. 31–36.

Wankhade, Mayur et al. (Aug. 2022). *A survey on sentiment analysis methods, applications, and challenges*. In: *Artificial Intelligence Review* 55.7, pp. 5731–5780. ISSN: 1573-7462. DOI: 10.1007/s10462-022-10144-1. URL: https://doi.org/10.1007/s10462-022-10144-1.

Mary, A. Jenifer Jothi et al. (2017). *Jen-Ton: A framework to enhance the accuracy of aspect level sentiment analysis in big data*. In: *2017 International Conference on Inventive Computing and Informatics (ICICI)*, pp. 452–457. URL: https://api.semanticscholar.org/CorpusID:44112128.

Wang, Yequan et al. (2019). *Aspect-level Sentiment Analysis using AS-Capsules*. In: *The World Wide Web Conference*. WWW '19. New York, NY, USA: Association for Computing Machinery, 2033–2044. ISBN: 9781450366748. DOI: 10.1145/3308558.3313750. URL: https://doi.org/10.1145/3308558.3313750.

Rønningstad, Egil et al. (Oct. 2022). *Entity-Level Sentiment Analysis (ELSA): An Exploratory Task Survey*. In: *Proceedings of the 29th International Conference on Computational Linguistics*. Ed. by Nicoletta Calzolari et al. Gyeongju, Republic of Korea: International Committee on Computational Linguistics, pp. 6773–6783. URL: https://aclanthology.org/2022.coling-1.589.

Liu, Bing (2015). *Aspect and Entity Extraction*. In: *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. Cambridge University Press, 137–188.

Zhang, Lei et al. (2014). *Aspect and Entity Extraction for Opinion Mining*. In: *Data Mining and Knowledge Discovery for Big Data: Methodologies, Challenge and Opportunities*. Ed. by Wesley W. Chu. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–40. ISBN: 978-3-642-40837-3. DOI: 10.1007/978-3-642-40837-3_1. URL: https://doi.org/10.1007/978-3-642-40837-3_1.

Ramshaw, Lance et al. (1995). *Text Chunking using Transformation-Based Learning*. In: *Third Workshop on Very Large Corpora*. URL: https://aclanthology.org/W95-0107.

Keraghel, Imed et al. (2024). *A survey on recent advances in named entity recognition*. arXiv: 2401.10825 [cs.CL].

Sinaga, Kristina P. et al. (2020). *Unsupervised K-Means Clustering Algorithm*. In: *IEEE Access* 8, pp. 80716–80727. DOI: 10.1109/ACCESS.2020.2988796.

Wang, Lipo (2005). *Support vector machines: theory and applications*. Vol. 177. Springer Science & Business Media.

Sutton, Charles et al. (2012). *An introduction to conditional random fields*. In: *Foundations and Trends® in Machine Learning* 4.4, pp. 267–373.

Berger, Adam et al. (1996). *A maximum entropy approach to natural language processing*. In: *Computational linguistics* 22.1, pp. 39–71.

Eddy, Sean R (1996). *Hidden Markov models*. In: *Current Opinion in Structural Biology* 6.3, pp. 361–365. ISSN: 0959-440X. DOI: https://doi.org/10.1016/S0959-440X(96)80056-X. URL: https://www.sciencedirect.com/science/article/pii/S0959440X9680056X.

Mikolov, Tomas et al. (2013). *Efficient estimation of word representations in vector space*. In: *arXiv preprint arXiv:1301.3781*.

Aizawa, Akiko (2003). *An information-theoretic perspective of tf–idf measures*. In: *Information Processing & Management* 39.1, pp. 45–65. ISSN: 0306-4573. DOI: https://doi.org/10.1016/S0306-4573(02)00021-3. URL: https://www.sciencedirect.com/science/article/pii/S0306457302000213.

Joulin, Armand et al. (2016). *Bag of Tricks for Efficient Text Classification*. arXiv: 1607.01759 [cs.CL].

Pennington, Jeffrey et al. (Oct. 2014). *GloVe: Global Vectors for Word Representation*. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Alessandro Moschitti et al. Doha, Qatar: Association for Computational Linguistics, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

Peters, Matthew E. et al. (2018). *Deep contextualized word representations*. arXiv: 1802.05365 [cs.CL].

Singh, Medha (July 2022). *After crypto, Stocktwits takes aim at stocks trading*. In: *Reuters*. URL: https://www.reuters.com/technology/after-crypto-stocktwits-takes-aim-stocks-trading-2022-07-19/ (visited on 12/20/2023).

Cui, Xin et al. (2024). *Embedded Value in Bloomberg News & Social Sentiment Data*. Received via email from Bloomberg L.P. on 2.2.2024. URL: https://data.bloomberglp.com/promo/sites/12/725454457_EDFSentimentWP.pdf (visited on 02/02/2024).

Gu, Chen et al. (2020). *Informational role of social media: Evidence from Twitter sentiment*. In: *Journal of Banking & Finance* 121, p. 105969. ISSN: 0378-4266. DOI: https://doi.org/10.1016/j.jbankfin.2020.105969. URL: https://www.sciencedirect.com/science/article/pii/S0378426620302314.

Araci, Dogu (2019). *FinBERT: Financial Sentiment Analysis with Pre-trained Language Models*. arXiv: 1908.10063 [cs.CL].

Tang, Yixuan et al. (Dec. 2023). *FinEntity: Entity-level Sentiment Classification for Financial Texts*. In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor et al. Singapore: Association for Computational Linguistics, pp. 15465–15471. DOI: 10.18653/v1/2023.emnlp-main.956. URL: https://aclanthology.org/2023.emnlp-main.956.

Zhao, Lingyun et al. (2021). *A BERT based Sentiment Analysis and Key Entity Detection Approach for Online Financial Texts*. In: *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 1233–1238. DOI: 10.1109/CSCWD49262.2021.9437616.

Liu, Yinhan et al. (2019b). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. arXiv: 1907.11692 [cs.CL].

Liu, Shanshan et al. (2019a). *Neural Machine Reading Comprehension: Methods and Trends*. In: *Applied Sciences* 9.18. ISSN: 2076-3417. DOI: 10.3390/app9183698. URL: https://www.mdpi.com/2076-3417/9/18/3698.

Son, Guijin et al. (2023). *Removing non-stationary knowledge from pre-trained language models for entity-level sentiment classification in finance*. In: *arXiv preprint arXiv:2301.03136*.

Raffel, Colin et al. (2020). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. In: *Journal of Machine Learning Research* 21.140, pp. 1–67. URL: http://jmlr.org/papers/v21/20-074.html.

Sinha, Ankur et al. (2022). *SEntFiN 1.0: Entity-aware sentiment analysis for financial news*. In: *Journal of the Association for Information Science and Technology* 73.9, pp. 1314–1335. DOI: https://doi.org/10.1002/asi.24634. eprint: https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/asi.24634. URL: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.24634.

Li, Xiaodong et al. (2014). *News impact on stock price return via sentiment analysis*. In: *Knowledge-Based Systems* 69, pp. 14–23. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2014.04.022. URL: https://www.sciencedirect.com/science/article/pii/S0950705114001440.

Wan, Xingchen et al. (Feb. 2021). *Sentiment correlation in financial news networks and associated market movements*. In: *Scientific Reports* 11.1, p. 3062. ISSN: 2045-

2322. DOI: 10.1038/s41598-021-82338-6. URL: https://doi.org/10.1038/s41598-021-82338-6.

Khedr, Ayman E et al. (2017). *Predicting stock market behavior using data mining technique and news sentiment analysis*. In: *International Journal of Intelligent Systems and Applications* 9.7, p. 22.

Stempel, Jonathan (Dec. 2023). *NY Times sues OpenAI, Microsoft for infringing copyrighted works*. URL: https://www.reuters.com/legal/transactional/ny-times-sues-openai-microsoft-infringing-copyrighted-work-2023-12-27/ (visited on 05/15/2024).

Bergen, Mark (Dec. 2023). *New York Times Sues Microsoft and OpenAI for Copyright Infringement*. URL: https://www.bloomberg.com/news/articles/2023-12-27/new-york-times-sues-microsoft-and-openai-for-copyright-infringement (visited on 05/15/2024).

Robertson, Katie (Apr. 2024). *8 Daily Newspapers Sue OpenAI and Microsoft Over A.I.* URL: https://www.nytimes.com/2024/04/30/business/media/newspapers-sued-microsoft-openai.html (visited on 05/15/2024).

Guardian, The (Apr. 2024a). *Eight US newspapers sue OpenAI and Microsoft for copyright infringement*. In: *Guardian*. (Visited on 05/15/2024).

Marc (Feb. 2024). *How do RSS feeds work?* URL: https://rss.com/blog/how-do-rss-feeds-work/ (visited on 07/15/2024).

Sajid, Md (July 2023). *Difference Between RSS and ATOM*. URL: https://www.tutorialspoint.com/difference-between-rss-and-atom (visited on 07/15/2024).

Finnhub (n.d.). *Finnhub - Free realtime APIs for stock, forex and cryptocurrency*. URL: https://finnhub.io/docs/api/company-news (visited on 05/16/2024).

Guardian, The (2024b). *UK's enemies could use AI deepfakes to try to rig election, says James Cleverly*. Article discussing the potential use of AI-generated deepfakes in elections, featuring statements from James Cleverly, the UK's Home Secretary. URL: https://www.theguardian.com/uk-news/2024/feb/25/uks-enemies-could-use-ai-deepfakes-to-try-to-rig-election-says-james-cleverly (visited on 04/22/2024).

Levenshtein, Vladimir I et al. (1966). *Binary codes capable of correcting deletions, insertions, and reversals*. In: *Soviet physics doklady*. Vol. 10. 8. Soviet Union, pp. 707–710.

Wikidata (2024). *Wikidata:Glossary - Wikidata*. URL: https://www.wikidata.org/wiki/Wikidata:Glossary (visited on 06/09/2024).

Quantstart (2024). *Understanding equities data*. URL: https://quantstart.com/articles/understanding-equities-data/ (visited on 05/15/2024)

Laura (Mar. 2023). *Using Airflow XComs - Laura*. URL: https://medium.com/@laura-fyi/using-airflow-xcoms-4703ea5e534a (visited on 05/12/2024).

# Acronyms

**API** Application Programming Interface. 20, 22–25, 27, 67, 73, 74, 79

**BERT** Bidirectional Encoder Representations from Transformers. 18

**BIO** Beginning-Inside-Outside. 8–10

**BMEWO** Beginning-Middle-End-Whole-Outside. 9, 10

**CNN** Convolutional Neural Network. 14

**CRF** Conditional Random Field. 13

**DAG** Directed Acyclic Graph. 64, 65, 67–71, 73, 80

**ELMo** Embeddings from Language Model. 14

**ETL** Extract Transform Load. 61, 62, 64–74, 79, 80

**GloVe** Global Vectors for Word Representation. 14

**HMM** Hidden Markov Model. 13

**IO** Inside-Outside. 9, 10

**IOB** Inside-Outside-Beginning. 8

**ME** Maximum Entropy. 13

**MRC** Machine Reading Comprehension. 18

**NER** Named Entity Recognition. 8, 61, 62, 67–73

**NLP** Natural Language Processing. 5, 8

**REST API** Representational State Transfer Application Programming Interface. 61, 62, 73, 75, 77, 78, 81, 82

**RNN** Recurrent Neural Network. 14

**RSS** Really Simple Syndication. 20, 22, 23

**SA** Sentiment Analysis. 5, 61, 62, 68–73

**SML** Supervised Machine Learning. 16

**SVM** Support Vector Machine. 13

**T5** Text-to-Text Transfer Transformer. 18

**TF-IDF** Term Frequency-Inverse Document Frequency. 14, 34

# A. Textual Data

This appendix presents daily news acticles volume of the selected companies, as discussed in Sections 3.4 and 3.5.

## A.1 Third-party Data Providers

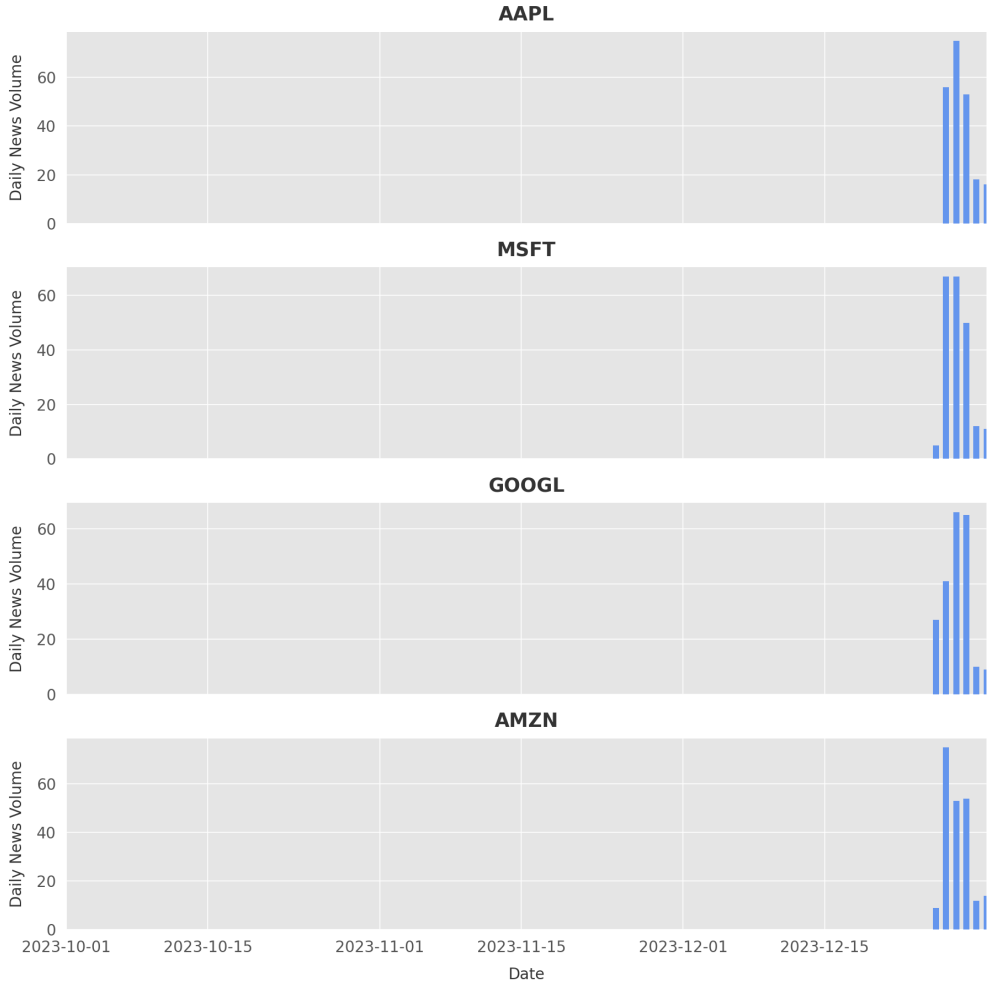



**Figure A.1** Finnhub daily news articles volume of Apple Inc. (AAPL), Microsoft Corp. (MSFT), Alphabet Inc. (GOOGL), and Amazon.com Inc. (AMZN) for the fourth quarter of 2023

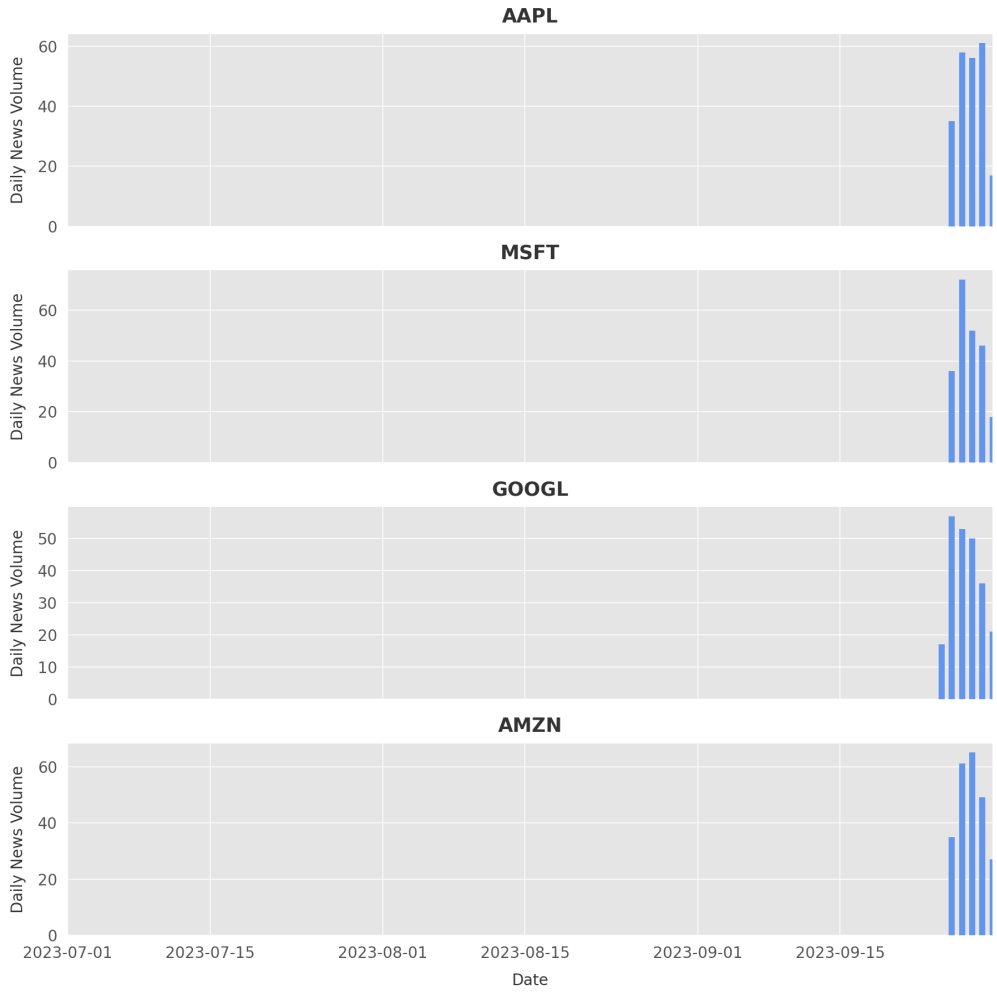



**Figure A.2** Finnhub daily news articles volume of Apple Inc. (AAPL), Microsoft Corp. (MSFT), Alphabet Inc. (GOOGL), and Amazon.com Inc. (AMZN) for the third quarter of 2023

## A.2 First-party Data Providers

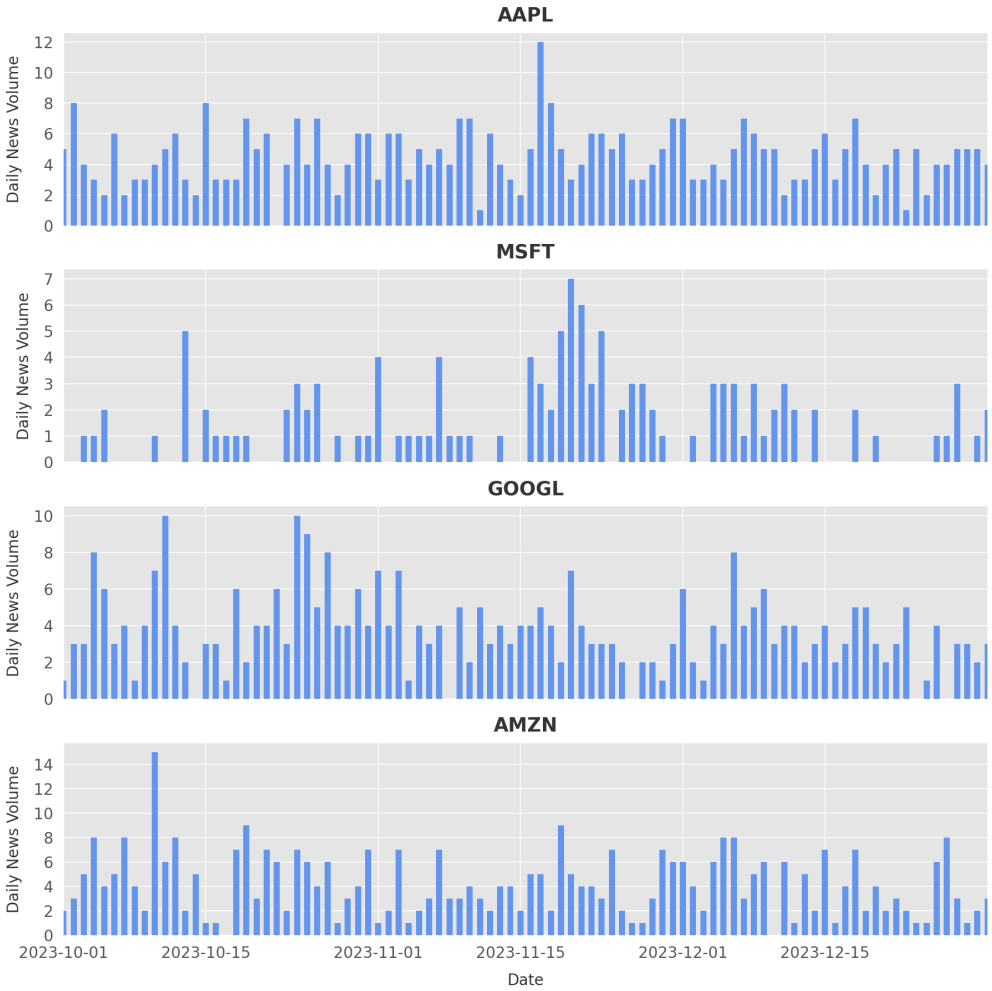



**Figure A.3** The Guardian daily news articles volume of Apple Inc. (AAPL), Microsoft Corp. (MSFT), Alphabet Inc. (GOOGL), Amazon.com Inc. (AMZN) for the fourth quarter of 2023

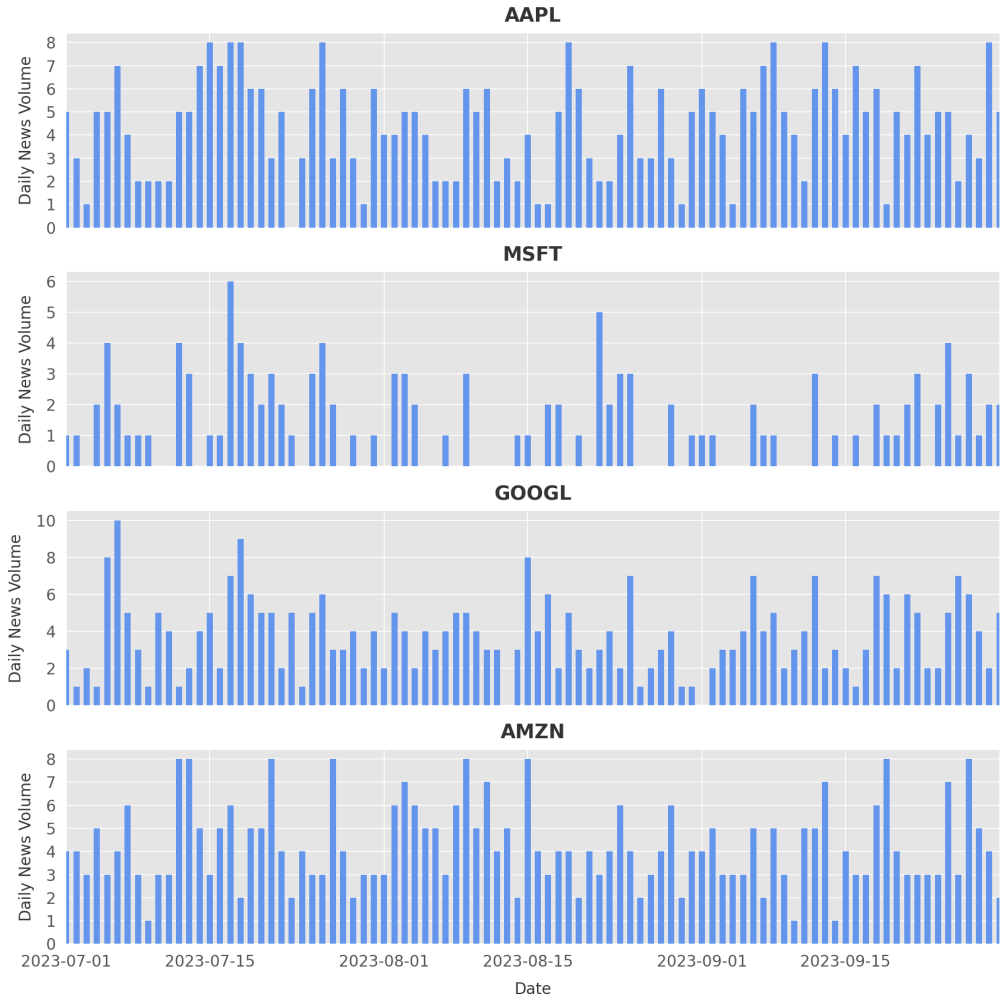



**Figure A.4** The Guardian daily news articles volume of Apple Inc. (AAPL), Microsoft Corp. (MSFT), Alphabet Inc. (GOOGL), and Amazon.com Inc. (AMZN) for the third quarter of 2023

# B. SPARQL Wrapper

This appendix contains the implementation of the SPARQL queries discussed in Section 4.4.3. The queries are divided into three parts, each representing a different approach to retrieving the ticker symbol of the selected entities. To ensure consistency in the text when comparing results with the naive approach methods, we will refer to the QID (*?id*) and Label (*?idLabel*) related to the Organization entity, as well as Ticker (*?ticker*) and Stock Exchange (*?exchangeLabels*) in the query result tables.

## B.1 Query 1: Direct ticker retrieval

**Listing B.1** SPARQL Query 1: Retrieve entity information for entities directly with the *stock exchange* property.

```
SELECT DISTINCT
  ?id             # Selects the entity ID
  ?idLabel        # Selects the label of the entity
  ?exchangesLabel # Selects the label of the exchange
  ?ticker         # Selects the ticker symbol

WHERE {
  # Retrieves labels in English
  SERVICE wikibase:label {
    bd:serviceParam wikibase:language
        "[AUTO_LANGUAGE],en".
  }

  # Specifies the QIDs of the entities
  VALUES ?id {
    wd:Q11463    # Adobe
    wd:Q3884     # Amazon
    wd:Q95       # Google
    wd:Q37156    # IBM
    wd:Q18811574 # Meta
    wd:Q2283     # Microsoft
    wd:Q48938223 # TikTok
    wd:Q21708200 # OpenAI
    wd:Q209330   # Instagram
  }

  # Specifies the QIDs of the stock exchanges
```

```
  VALUES ?exchanges {
    wd:Q82059     # NASDAQ
    wd:Q13677     # NYSE
  }

  # Matches entities with stock exchange property
  ?id p:P414 ?exchange.

  # Filters the exchanges to those specified
  # and retrieves the ticker symbol
  ?exchange ps:P414 ?exchanges;
            pq:P249 ?ticker.

  # Filters tickers without an end time
  FILTER NOT EXISTS {
      ?exchange pq:P582 ?endTime.
  }
}
```

## B.2 Query 2: Owner-based ticker retrieval

**Listing B.2** SPARQL Query 2: Retrieve entity information for remaining entities with the *owned by* property.

```
SELECT DISTINCT
  ?id             # Selects the entity ID
  ?idLabel        # Selects the label of the entity
  ?exchangesLabel # Selects the label of the exchange
  ?ticker         # Selects the ticker symbol

WHERE {
  # Retrieves labels in English
  SERVICE wikibase:label {
    bd:serviceParam wikibase:language
        "[AUTO_LANGUAGE],en".
  }

  # Specifies the QIDs of the remaining entities
  VALUES ?id {
    wd:Q95       # Google
    wd:Q18811574 # Meta
    wd:Q48938223 # TikTok
    wd:Q21708200 # OpenAI
    wd:Q209330   # Instagram
   }

  # Specifies the QIDs of the stock exchanges
```

```
  VALUES ?exchanges {
    wd:Q82059     # NASDAQ
    wd:Q13677     # NYSE
  }

  # Matches entities with owner property
  ?id wdt:P127 ?owner.
  ?owner p:P414 ?exchange.

  # Filters the exchanges to those specified
  # and retrieves the ticker symbol
  ?exchange ps:P414 ?exchanges;
            pq:P249 ?ticker.

  # Filters tickers without an end time
  FILTER NOT EXISTS {
      ?exchange pq:P582 ?endTime.
  }
}
```

## B.3 Query 3: Differentiated ticker retrieval

**Listing B.3** SPARQL Query 3: Retrieve entity information for remaining entities with the *different from* property.

```
SELECT DISTINCT
  ?id             # Selects the entity ID
  ?idLabel        # Selects the label of the entity
  ?exchangesLabel # Selects the label of the exchange
  ?ticker         # Selects the ticker symbol

WHERE {
  # Retrieves labels in English
  SERVICE wikibase:label {
  bd:serviceParam wikibase:language
    "[AUTO_LANGUAGE],en".
  }

  # Specifies the QIDs of the remaining entities
  VALUES ?id {
  wd:Q18811574 # Meta
  wd:Q48938223 # TikTok
  wd:Q21708200 # OpenAI
  }

  # Specifies the QIDs of the stock exchanges
  VALUES ?exchanges {
```

```
  wd:Q82059     # NASDAQ
  wd:Q13677     # NYSE
  }

  # Matches entities with the 'different from' property
  ?id wdt:P1889 ?differs.
  ?differs p:P414 ?exchange.

  # Filters the exchanges to those specified
  # and retrieves the ticker symbol
  ?exchange ps:P414 ?exchanges;
       pq:P249 ?ticker.

  # Filters tickers without an end time
  FILTER NOT EXISTS {
    ?exchange pq:P582 ?endTime.
  }
}
```

# C. Sentiment and Adjusted Close Price

This appendix presents the outcomes of the sentiment based hold strategy outlined in Section 5.4.1, along with the analysis of the correlation between sentiment and the adjusted close prices of selected companies, as detailed in Section 5.4.2.

## C.1 Hold Strategy

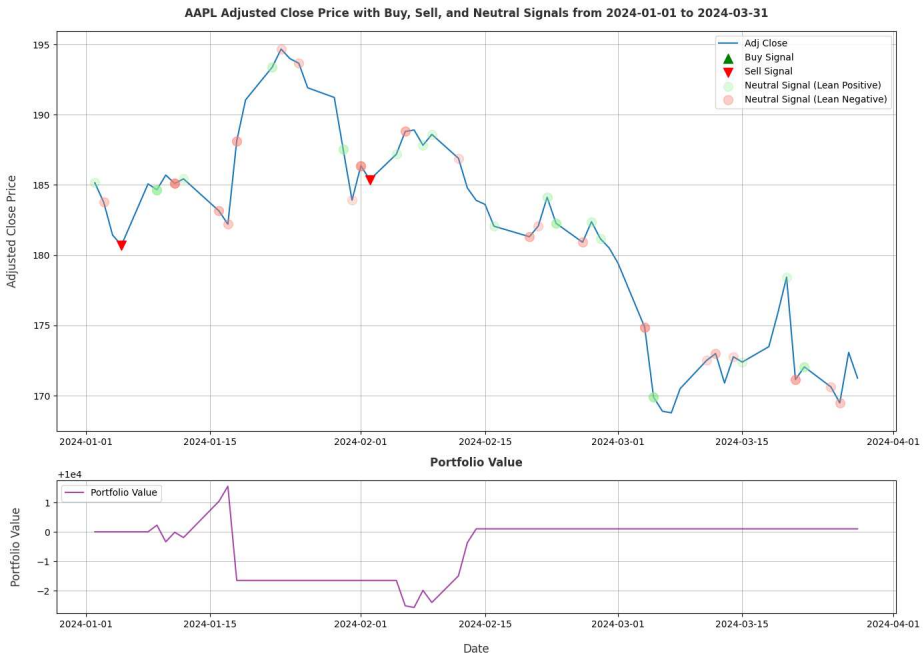



**Figure C.1** Apple Inc. (AAPL) adjusted close price with buy, sell and neutral signals in the first quarter of 2024. Hold strategy based on sentiment signals.

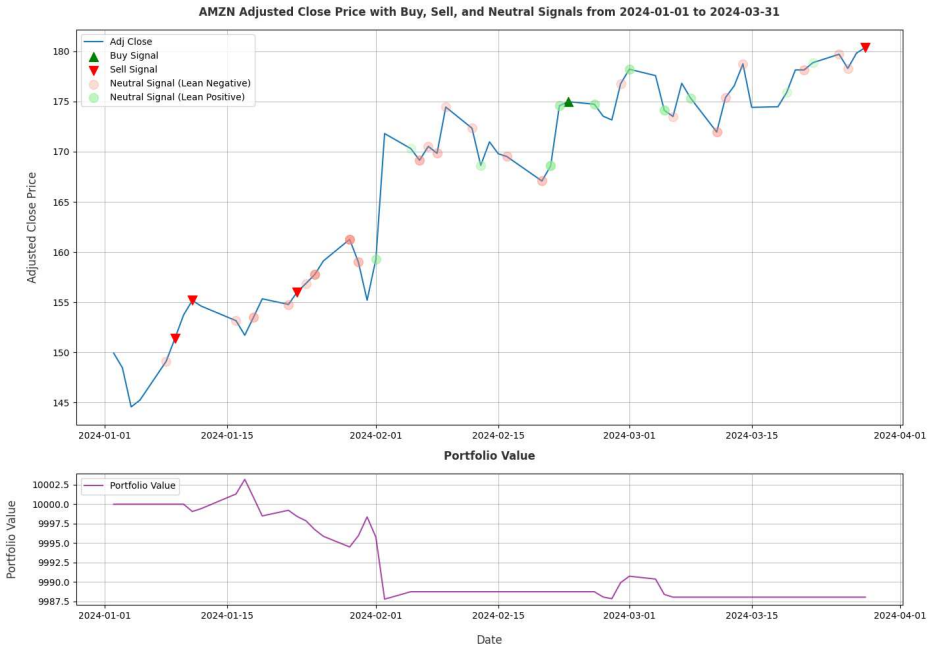



**Figure C.2** Amazon.com Inc. (AMZN) adjusted close price with buy, sell and neutral signals in the first quarter of 2024. Hold strategy based on sentiment signals.

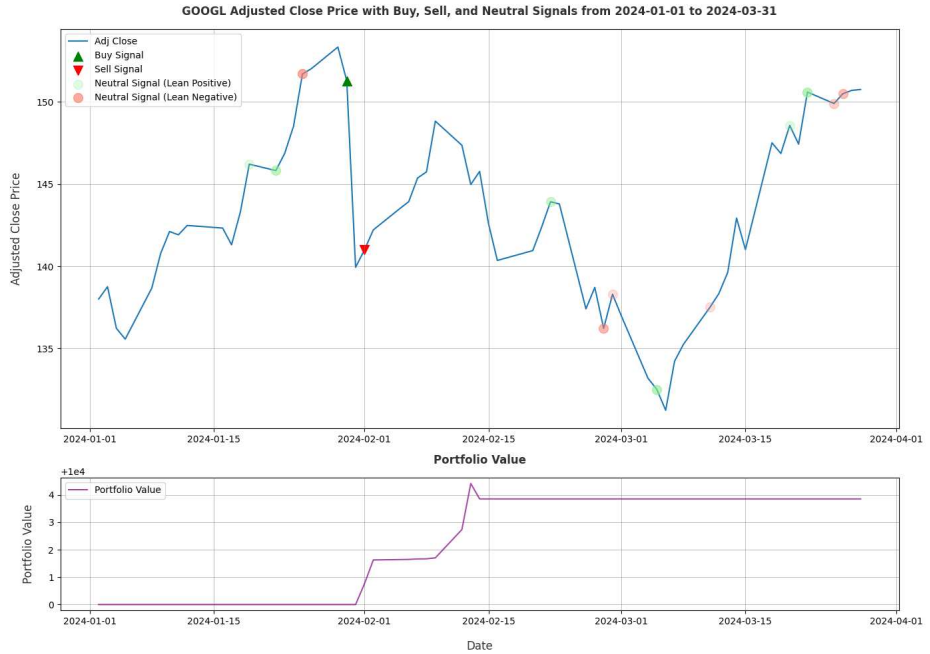



**Figure C.3** Alphabet Inc. (GOOGL) adjusted close price with buy, sell and neutral signals in the first quarter of 2024. Hold strategy based on sentiment signals.

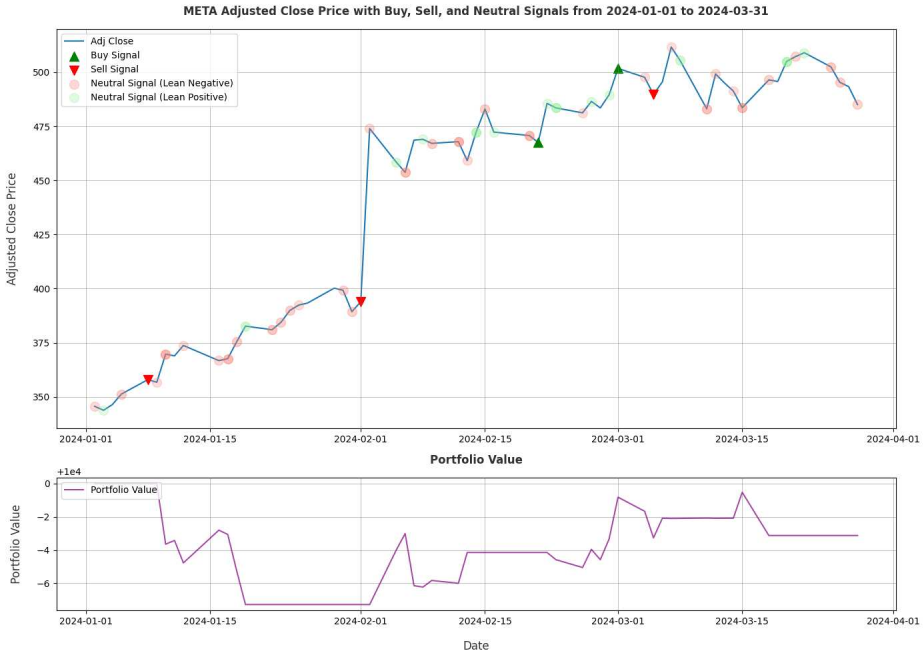



**Figure C.4** Meta Platforms Inc. (META) adjusted close price with buy, sell and neutral signals in the first quarter of 2024. Hold strategy based on sentiment signals.

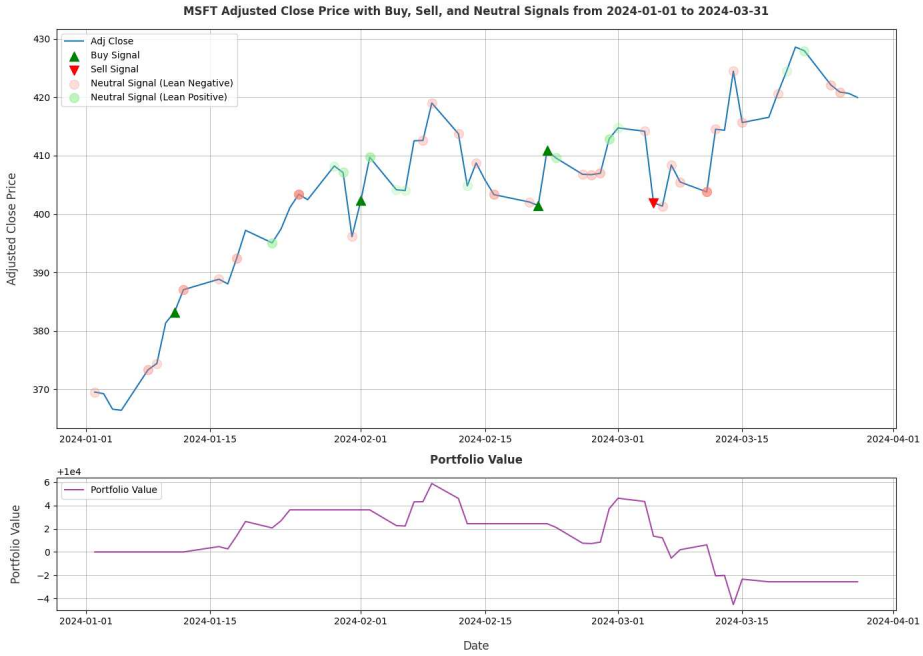



**Figure C.5** Microsoft Corp. (MSFT) adjusted close price with buy, sell and neutral signals in the first quarter of 2024. Hold strategy based on sentiment signals.

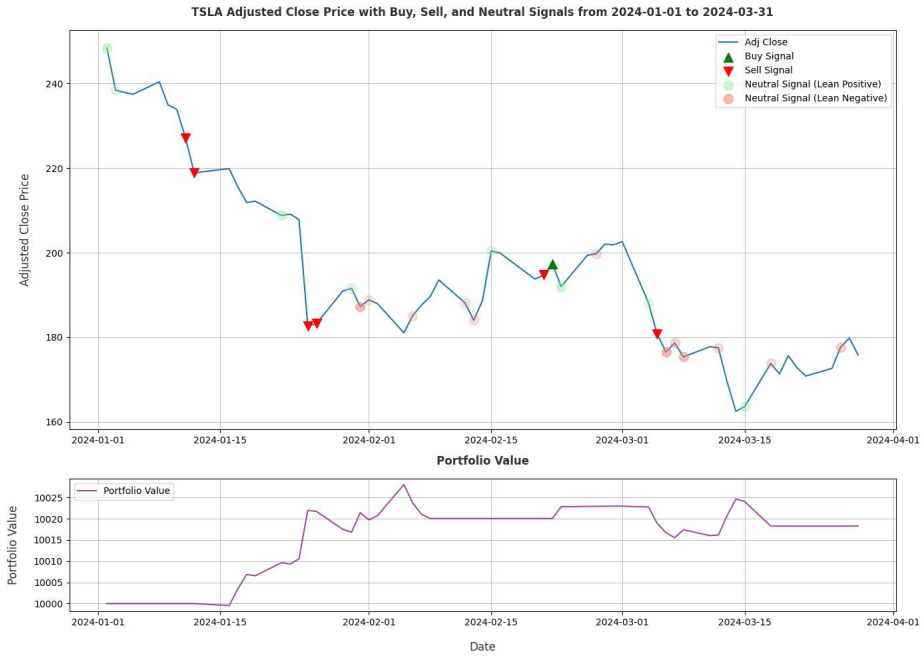



**Figure C.6** Tesla Inc. (TSLA) adjusted close price with buy, sell and neutral signals in the first quarter of 2024. Hold strategy based on sentiment signals.

## C.2 Correlation

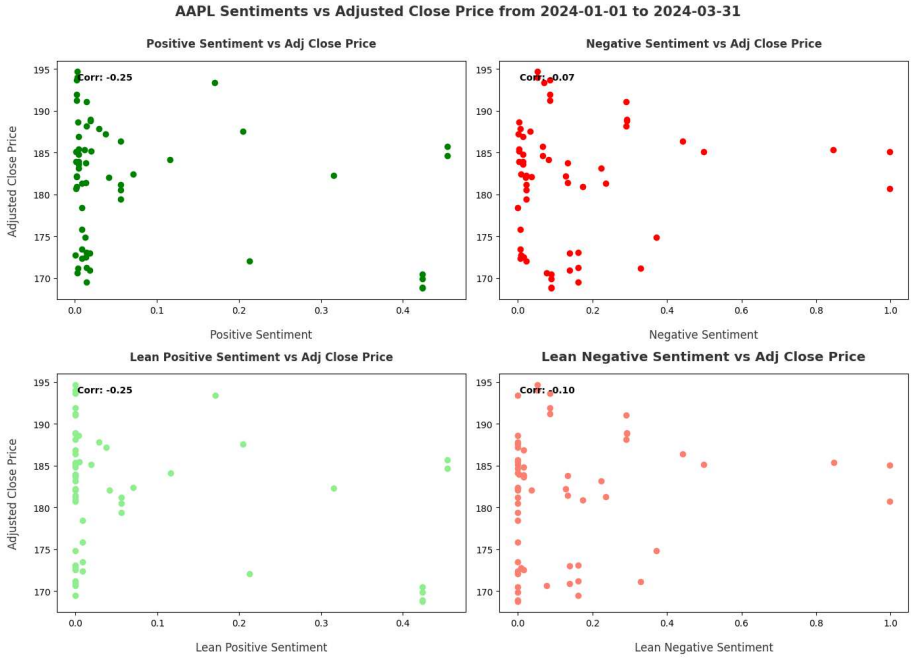



**Figure C.7** Apple Inc. (AAPL) sentiment correlation with adjusted close price in the first quarter of 2024.

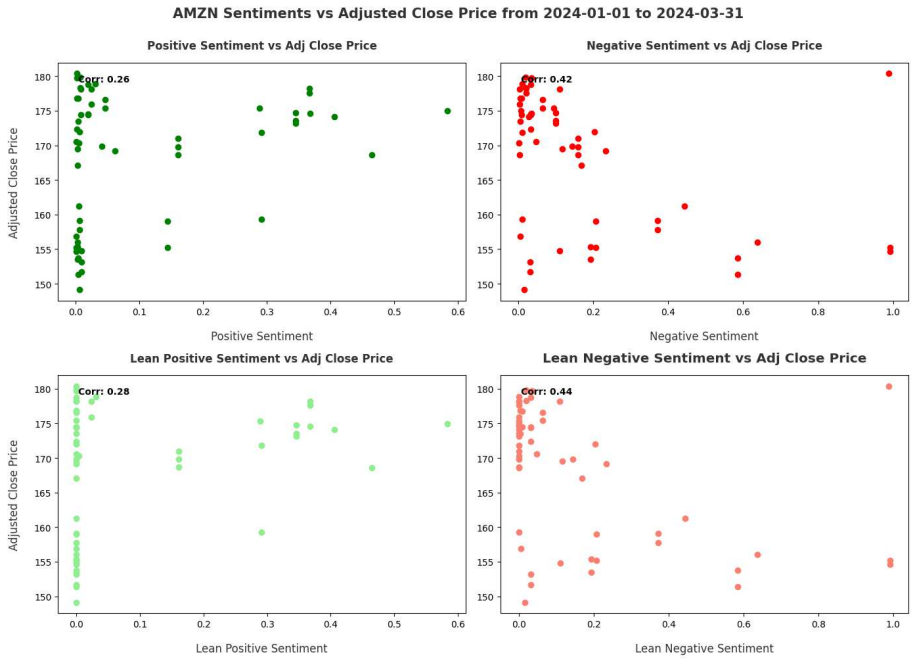



**Figure C.8** Amazon.com Inc. (AMZN) sentiment correlation with adjusted close price in the first quarter of 2024.

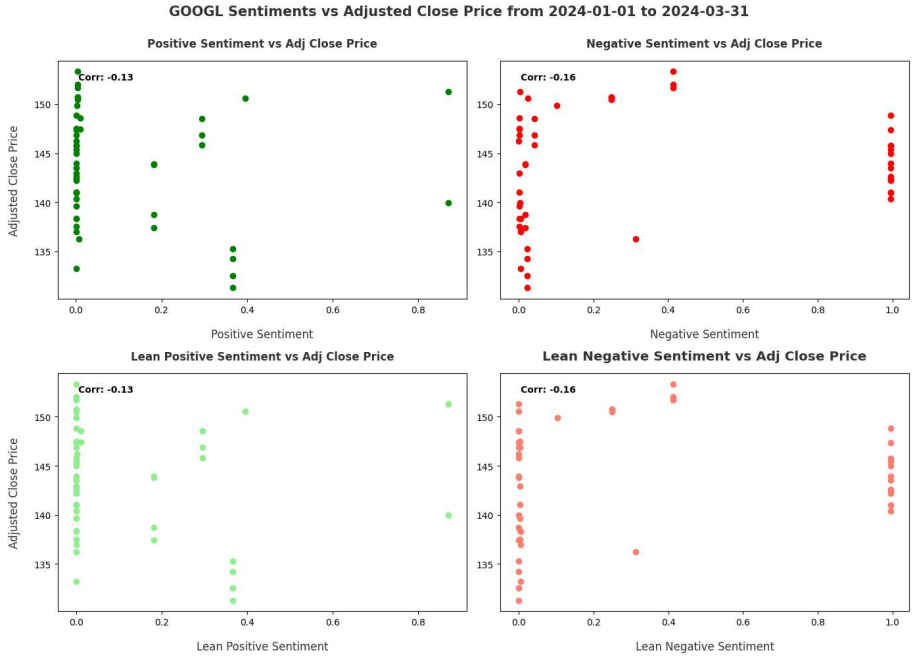



**Figure C.9** Alphabet Inc. (GOOGL) sentiment correlation with adjusted close price in the first quarter of 2024.

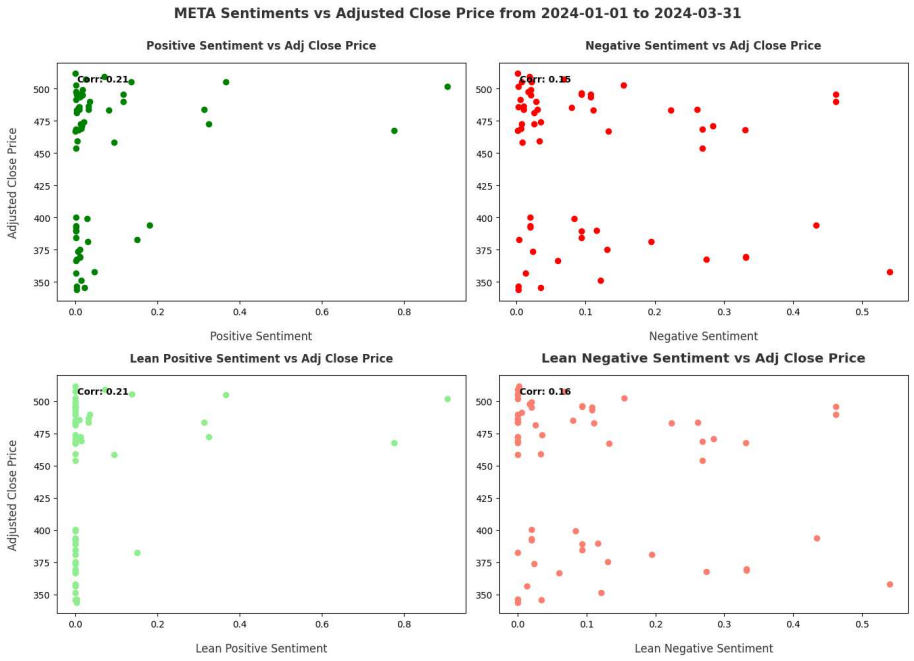



**Figure C.10** Meta Platforms Inc. (META) sentiment correlation with adjusted close price in the first quarter of 2024.

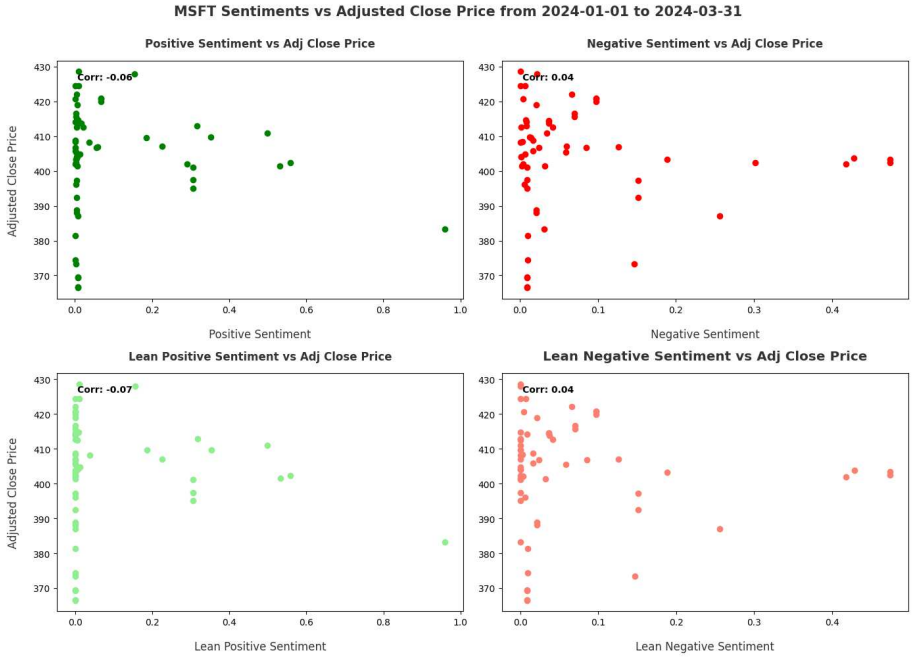



**Figure C.11** Microsoft Corp. (MSFT) sentiment correlation with adjusted close price in the first quarter of 2024.

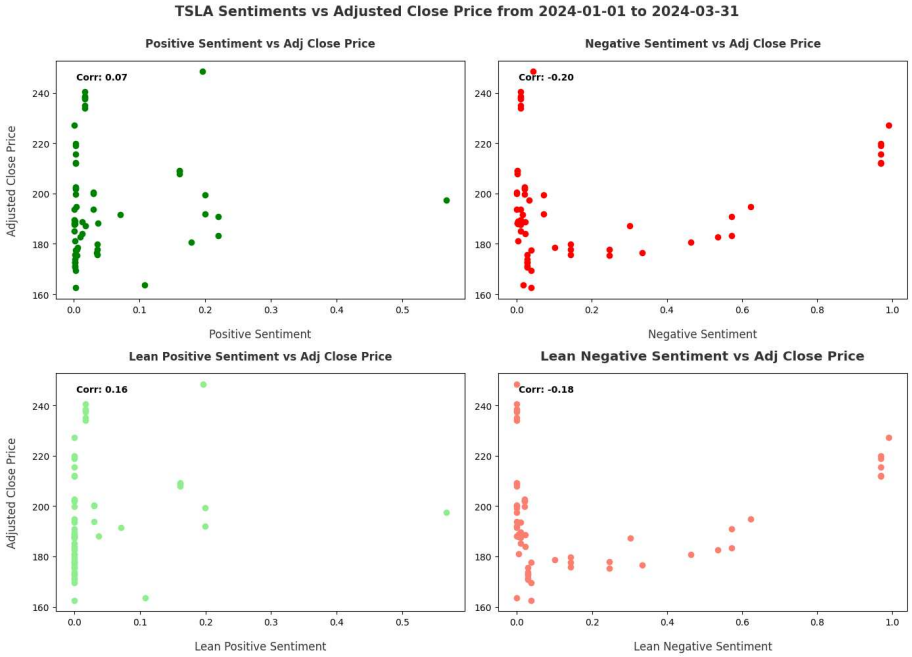



**Figure C.12** Tesla Inc. (TSLA) sentiment correlation with adjusted close price in the first quarter of 2024.