Decoding the News with Transformers:

Evaluating Sentiment-Driven Stock Price Prediction

Fredrik Fourong & William Nordansjö

Lund University School of Economics and Management

May 2025

Supervisor: Muhammad Qasim

Abstract

This thesis evaluates an ensemble-based framework for predicting stock prices using pretrained transformer models for sentiment analysis and locally trained transformers for time series forecasting. Sentiment signals were generated from financial news using FinBERT, RoBERTa, and VADER, combined with historical stock price data from Refinitiv Eikon. To enhance label quality, a custom ensemble labeling pipeline was developed, combining weak keyword-based labeling methods with the spaCy transformer-based named entity recognition model, using weighted voting. Manual evaluation of 1,400 article-label pairs ensured a balance between precision and recall. The study focuses on "The Magnificent Seven" companies and evaluates 42 model configurations, combining three sentiment models, two transformer architectures, and seven company-specific datasets. Forecasting was performed using Informer and PatchTST, with hyperparameter tuning. Evaluation included regression and directional metrics as well as a modified Diebold-Mariano test. Due to the inclusion of multiple company-specific models, results are also reported as average scores across the seven companies. Results show that PatchTST consistently outperformed Informer in regression tasks, while Informer achieved marginally higher directional accuracy. These outcomes were partly driven by systematic differences in sentiment strength across models, with FinBERT outputting more optimistic signals than RoBERTa and VADER. Key contributions include the end-to-end use of state-of-the-art transformers, a custom ensemble labeling pipeline, and extensive empirical evaluation.

JEL Classification: G17, C53, C45, C81

Keywords: sentiment analysis, stock prediction, labeling, transformers, deep learning

"Sometimes even the wisest of men and machines can be in error."

— Optimus Prime

Acknowledgments

We would like to extend our heartfelt thanks to our supervisor Qasim for his constant enthusiasm, advice and optimism. In addition, we would like to thank our peers for their support and engagement during the long hours in the computer lab.

Contents

Chapter 1

Introduction

Human emotion is deeply embedded in our everyday lives, whether it is a chef preparing a meal or a painter creating art, emotions often influence and shape our actions. Another area where emotions frequently come to the surface is in the world of finance. During periods of economic stagnation or growth, emotional responses can be observed in the behavior of stock markets, often leading to sharp sell-offs or surges in buying activity. What often tells the crowd about these economic states are the news. The news often include titles which provoke emotion, often referred to as clickbait, and thereby influence how both private and institutional investor behavior. Consequently, the study of emotion or sentiment in financial news in relation to financial markets can be of great interest, and also something that has come to be a lot more common to include.

Natural Language Processing (NLP) is a branch of *Artificial Intelligence* (AI) that uses machine learning to interpret, understand, and generate human language but also predict sentiment (Stryker & Holdsworth, 2025). Tasks such as *Named Entity Recognition* (NER) and sentiment analysis are commonly visited, and models such as FinBERT, RoBERTa, VADER and spaCy are often referred to. This technology serves as the backbone of *Large Language Models* (LLMs) such as OpenAI's ChatGPT. The techniques of NLP and machine learning can benefit from larger amounts of data, and produce robust results in form of predictions or classification and the task of training these models can be quite computationally extensive (Goodfellow et al., 2016, p. 440).

A major breakthrough in the field came with the paper *Attention is All You Need* by Vaswani et al. (2017) introduced the Transformer-architecture which has positively impacted a variety of NLP tasks such as sentiment analysis, machine translation and summarization of language. This new methodology of using

Transformer-based models thus provides a new opportunity to be implemented in the financial NLP and forecasting tasks. The effectiveness of these models has in previous research been steadily been proven to improve on the capacity and predictive power of previous state-of-the-art (SOTA) machine learning models. However, these type of architectures are often deemed as 'data hungry' compared to previous machine learning models such as *Long Short-Term Memory* (LSTM) or *Recurrent Neural Networks* while also being computationally intensive seen to both time and space complexity (Brown et al., 2020).

The scope of this thesis is centered around three primary objectives and corresponding research questions: (1) to evaluate the informational or sentiment signal generated by three different sentiment models, and (2) to assess the forecasting performance of two transformer architectures, Informer (Zhou et al., 2021) and PatchTST (Nie et al., 2022), when using these signals as input and (3) to investigate the capacity of transformer-based models to extract meaningful patterns from financial time series when sentiment signals are weak or noisy. The research questions are formally described as follows:

- **Research question 1:** How do different generating models compare on providing sentiment signals for financial forecasting?
- **Research question 2:** How do Informer and PatchTST compare in terms of forecasting performance in financial time series in relation to both continuous and discrete outputs?
- **Research question 3:** To what extent do transformer-based models extract meaningful patterns from financial time series when sentiment signals are weak or noisy?

To achieve this we utilize a comprehensive data set of financial news called *Financial News and Stock Price Integration Dataset* (FNSPID) to base our analysis on. The data set made available by Dong et al. (2024) contains 15.7 million financial news records and 29.7 million stock prices for 4,775 S&P500 companies during the years from 1999 through 2023. This dataset is combined with trading data; volume and closing prices, extracted from Refinitiv Eikon. We apply three distinct sentiment label models, FinBERT, RoBERTa and VADER to score the sentiment of each news article. These sentiment scores, together with the technical financial indicators, are then integrated into the transformer models. Due to the scale of FNSPID and various model combinations we limit the analysis to seven companies: Amazon (AMZN), Apple (AAPL), Google (GOOGL), Meta (META), Microsoft (MSFT), NVIDIA (NVDA) and Tesla (TSLA), often referred to as "The Magnificent Seven" (MAG7). These companies are selected due to their high frequency in financial news coverage. The combination of companies, sentiment analysis models and transformer architectures result in a total of 42 experimental configurations. Therefore, in the results chapter, the forecasting scores are averaged across the three sentiment models.

This thesis have the following structure. In the second chapter, we provide an overview of previous research and literature on the three main subjects: NLP, transformer-based learning and data labeling. This chapter aims to a grounded understanding of these fields, while also offering insights into the challenges that may arise when working with similar tasks. The third chapter goes into the technical details of our dataset and the issues discovered within it. The discovery of mislabeled data leads this work to propose a potentially novel pipeline of labeling data. Chapter three concludes with descriptive statistics to illustrate and verify the previously identified issues. The fourth chapter describes the methodology related to the research objectives and formally explains the sentiment analysis models and transformer architectures applied in this thesis. The fifth and final chapter includes the analysis, discussion of results, and the final conclusion, including limitations and suggestions for future research.

The main contributions from this work are:

- We introduce a custom labeling pipeline
- We conduct extensive empirical evaluation across 42 different model configurations, offering insights into their relative performance and robustness
- End-to-end use of SOTA transformer-based models

Our results imply that there are differences between sentiment methods and transformer models, and this effect can be more or less evident depending on the evaluation metrics used. The methods for generating sentiment scores show a clear difference in their restrictiveness and optimism toward each news article, where RoBERTa is the most pessimistic, while FinBERT is the most optimistic. Among the two transformer models, we find that PatchTST performed better in terms of distance and regression metrics, whereas Informer performs slightly better in pseudo classification. This may be due to the different attention mechanisms in the two models, where Informer's attention mechanism may struggle with potentially weak or noisy signals.

Chapter 2

Previous research

In this section, we review previous research that can be leveraged in this thesis and used to support the both methodological and conceptual choices made in this thesis. Research on financial forecasting is extensive. Financial modeling gained academic traction in the 1970s, particularly with the introduction of the Efficient Market Hypothesis (Fama, 1970). While event-based modeling has been explored for decades, advances in computational power have enabled the adoption of machine learning approaches for stock price prediction, ranging from traditional statistical approaches like *Support Vector Machines* (SVM) to more complex neural networks. This section highlight relevant prior studies that form a foundation for informed decisions, particularly regarding both data modeling and the selection of predictive techniques.

2.1 Natural Language Processing

NLP is an interdisciplinary field that spans psychology, cognitive and neural sciences, as well as data and computer science. The field began in the 1950s as a core area of AI. Early research revealed that language analysis presents a significant challenge due to the inherent ambiguity of human language as words can carry multiple meanings and also due to variability introduced by human errors, such as misspellings. In the 1980s, rule-based approaches began to give way to statistical methods, such as machine learning, which instead focused on learning the statistical distributions of linguistic patterns. Since the late 2000s, deep learning has emerged as the superior approach, leveraging neural networks capable of capturing far more complex relationships than earlier methods (Zhang & Teng, 2021, pp. 3–4). NLP explores the automation of language processing and generation. Its applications range from simple tasks like string matching and

tokenization to advanced systems, such as ChatGPT, that use neural networks to summarize content or translate between languages. This thesis includes several different NLP tasks such as tokenization and sentiment scoring, but mainly deriving from the outputs provided by external models explained in the next subsection.

Bollen et al. (2011) explored the field of behavioral economics to study how emotions affect individual behavior and decision-making. The data used in this investigations consisted of text content from large-scale twitter feeds, which was analyzed for its correlation to the value of the Dow Jones Industrial Average (DJIA) over time. Two methods were employed to track the sentiment of the collective mood. To support their analysis a Granger Causality Analysis and Self-organizing Fuzzy Neural Network were used to test their hypothesis that mood affects the DJIA closing price. Their results indicated that accuracy could be significantly improved by specific public mood dimensions, ultimately achieving an accuracy of 87.6% in predicting daily changes of the DJIA closing price. Thus proving that collective moods can improve stock market prediction accuracy. This was one of the first studies to suggest that social media, beyond traditional financial news, could be a valuable data source.

Another study that supports the use of textual information from news sources was conducted by Schumaker & Chen (2009) who introduced a predictive model based on over 9000 financial news articles and 10 million stock quotes. The method involved several different NLP techniques such as *Bag of Words*, *Noun Phrases* and *Named Entities* which were applied using a SVM to predict stock prices. They found that the model combining both variables derived from textual data and numerical variables had the best accuracy with a *Mean Squared Error* (MSE) of 0.04261 and 57.1% directional accuracy of the future stock price. In a simulated trading study this model also achieved the highest return with a 2.06% return on investment. Their research reinforces the idea that stock predictions should not rely solely on numerical data like opening prices or trading volume but should also consider incorporating textual information from news sources.

In NLP however, it can be crucial to understand and acknowledge contextual biases that can arise as words and language can have drastically different meanings. To address this matter, Loughran & McDonald (2011) investigated whether a common word list used in psychology and sociology could be applied in a financial context. Previous research had suggested that negative word counts can reliably reflect the sentiment of text. The use of Harvard Psychological Dictionary (H4N) was commonly employed word list in classifying whether a word is positive or negative. By evaluating 50,115 annual reports (10-Ks), they provided evidence that this list approximately 73.8% of the negative word counts in the word list were not considered negative in a financial context. The authors constructed a new list of 2,337 words that describe negative words in the financial context, and the list is referred to as Fin-Neg. One of their

recommendations was to use term weighting when calculating word counts, but also that financial research should be careful when using word classification techniques that come from outside the financial domain. This is particularly relevant to our research, as it highlights the need for financial-specific sentiment analysis rather than relying on generic models, thus leading us into the next topic.

Simkin & Roychowdhury (2014) investigated the decay of user attention on web articles such as blog posts and news articles over time. To do this, they analyzed IP logs from more than a hundred websites. Their findings revealed that article views typically follow a decaying power-law relative to the time since publication. The decay exponents ranged from 0.6 to 3.2, and most commonly around 1.5. The authors argued that the pattern of attention decay is best explained by a visibility-factors as articles are pushed further from the front page their exposure and access rate declines. To support their argument, they conducted empirical tests, including a controlled link-rotation experiment, which demonstrated a clear drop in click probability based on link position. Their theory aligned well with observed power-law patterns and provided a more consistent explanation for attention decay across platforms. This research is highly relevant to this study, as it offers important insights into how visibility and time affect the news engagement, which is directly related to how sentiment signals from news articles may fade in predictive models for stock prices. It also supports the idea of using time-decay functions or weighting schemes when incorporating older news into NLP models.

Building on previous research that incorporated textual information into models, it is also important to recognize that sentiment can be influenced by contextual biases. In this context Araci (2019) introduced FinBERT, a *Bidirectional Encoder Representations from Transformers* (BERT)-based model, fine-tuned specifically for financial sentiment analysis. Araci highlighted the challenges of financial sentiment analysis, primarily due to domain-specific language and a lack of labeled data. The goal was to develop a language model capable of handling various NLP tasks within the financial domain, thereby facilitating further research in the field. To achieve this, Araci pre-trained Google's original BERT model on three financial-domain collection of documentse: TRC2-Financial, Financial PhraseBank, and FiQA Sentiment. The model's performance was evaluated against three baseline methods, including LSTM classifiers using GloVe and ELMo embeddings, as well as a ULMFiT classifier. The findings demonstrated improvements across all evaluated metrics compared to the existing SOTA on two financial sentiment analysis datasets. Notably, FinBERT achieved superior performance even when fine-tuned on a smaller training set and with only partial model adjustments, outperforming prior machine learning approaches.This study directly addressed the contextual issues raised by Loughran & McDonald (2011), offering a more effective approach to processing financial text and supporting our decision to use FinBERT.

2.2 Transformer-based learning

Recent years have seen massive strides in machine learning methods. One of the most significant advancements, however, was the Transformer model by Vaswani et al. (2017), who, in their influential article *Attention is All You Need*, proposed a new model in order to improve text sequencing. Earlier models that have been widely used and considered SOTA are the LSTM and *Gated Recurrent Units* (GRUs). These recurrent models generates a sequence of hidden states which are functions of previous states and inputs, which limits their ability to capture long-term dependencies due to memory constraints. Although recent work in deep learning has made advancements in these type of architectures, particularly regarding computational efficiency, limitations still remain. The Transformer is a type of network architecture that, unlike previous models, eliminates recurrence and convolutions entirely, instead relying solely on attention mechanisms. This mechanism allows modeling dependent data regardless of distance, and therefore can draw global context modeling between input and output. By implementing this new approach, the authors significantly reduced computational cost relative to translation quality. This innovation opens an interesting possibility to use transformers in scenarios of smaller scale, making them suitable for the context of this thesis.

Transformers have since been adapted to time-series and one of the earlier works was conducted by Zerveas et al. (2020) who proposed a transformer-based approach in a multivariate time series context. The authors investigated the use of a transformer encoder for tasks including unsupervised representation learning, time series regression and classification. They developed a framework which allows for broader application by leveraging unlabeled data and then use the trained transformer model to extract vector representations of the time series through an input denoising objective. They applied their methodology on several public datasets and demonstrated that transformers are superior compared to previously established models, even with smaller datasets. Although this study was primarily focused on unsupervised learning, this thesis draws additional support in the use of transformer-based methods, due to their superiority compared to current SOTA methods.

Adding to the previous research on transformers, Zhou et al. (2021) addressed the high computational costs associated with the strong predictive capabilities of transformer models. Transformers involve a quadratic time complexity and high memory usage per layer, and also perform step-by-step inference. This ultimately limits the scalability of models, as stacking several layers means multiplying the time complexity by the number of layers. Additionally, dynamic decoding slows down prediction speed when generating longer output sequences. To tackle these issues, the authors proposed a new type of self-attention mechanism, namely a ProbSparse Self-Attention, which reduces the quadratic time complexity

and memory usage to $\mathcal{O}(L \log L)$. The self-attention performs a distillation process that filters out weaker attention scores in each of the stacking layers which also reduce the total space complexity. In regard to the decoding process, they introduced a more efficient generative decoding approach that requires only a one-step forward operation. To evaluate their model, they tested it on four large-scale datasets, two real-world and two benchmark datasets. The results showed significant improvements over existing methods for long sequence time-series forecasting tasks.

Another study that proposed an alternative solution to the computational limitations of transformers was conducted by Nie et al. (2022), who aimed to redesign transformer-based models in time-series forecasting and self-supervised representation learning. The new architecture is composed by two parts. The first involves slicing time-series into several *patches* which then enters the Transformer as tokens. The second introduces channel independence, where every channel contains a single univariate time-series, and all channels share the same embedding and transformer weights. This redesign comes with several benefits: patching provides that possibility to save local semantic information in embeddings but also that the computation and memory usage is reduced quadratically and the model can deal with longer history in time-series. The authors demonstrated significant improvements in accuracy compared to previous SOTA transformer-based models.

A recent study implementing transformers in sentiment analysis tasks was authored by Gu et al. (2024). In this paper, the authors proposed a combined FinBERT-LSTM model to predict stock prices, leveraging information from news articles related to financial assets. The dataset consisted of a large collection of 843,062 news articles ranging from 2009 through 2020, including the article title, URL, and related stock ticker, which were then combined with numerical data on closing price and volume. The FinBERT component classified article titles into sentiment categories (negative, neutral, or positive) transformed into probabilities using a softmax function, ensuring the sentiment scores ranged between 0 and 1 and summed to 1 across all categories. These scores were subsequently averaged on a daily basis to align with financial data. In addition to the FinBERT-LSTM model, the study also evaluated a standard LSTM network and a *Deep Neural Network* (DNN) as benchmarks for comparison. The performance of the models was evaluated using *Mean Absolute Error* (MAE), *Mean Absolute Percentage Error* (MAPE), and accuracy. The results showed that the proposed FinBERT-LSTM model achieved the best performance, with an MAE of 173.67 and an accuracy of 95.5%. This was significantly better than the vanilla LSTM (MAE: 183.36, Accuracy: 92.8%) and especially the DNN model (MAE: 489.32, Accuracy: 78.0%). These results clearly indicate that integrating FinBERT significantly enhanced the predictive capability of the model.

2.3 Data Labeling

Supervised learning is one of the most widely used approaches in statistics and machine learning, relying on labeled datasets to train predictive models. However, obtaining high-quality labeled data is often a significant bottleneck, particularly in domains requiring domain-specific knowledge or manual annotation. Although labels are not used as supervisory signals in this thesis, it is wise to gain some understanding of applications related to labeling. In this section, we review existing literature on data labeling strategies, focusing on both manual and automated techniques. While supervised learning requires labels, the process of generating those labels can itself rely on unsupervised or semi-supervised approaches, especially when dealing with large-scale unstructured data, as is the case our work.

Data labeling is a challenging and time-extensive task. In an empirical study conducted by Fredriksson et al. (2020), researchers investigated the issues and challenges that comes with labeling data. To do this, they interviewed data scientists at a worldwide telecommunication provider and one specialized in labeling. The study found that some of the more common techniques for labeling data included crowd sourcing and active learning. Active learning labels samples by the level of informativeness, which is done by query strategies which can differ based on the task at hand. The process is iterative and requires repetition where a model is trained and evaluated until a learner is satisfied. Crowdsourcing involves allowing a third-party contributors and relies heavily on human labor, for example, one company labeled data using 1,000 people. One of the key challenges with crowd sourcing is validating the quality of labels and the annotator, as the task at hand can rely on domain knowledge. The study also discussed the potential of semi-supervised learning, which can be employed in the event where only a fraction of the data is labeled. The labeled portion is used to train a model, which then label the remaining data. Common techniques in this area include the *Expectation-Maximization* algorithm and models such as SVMs. To improve label quality, the authors suggested techniques like repeated labeling and incorporating machine learning methods to handle noisy labels. Ultimately, while data labeling remains a resource-intensive component of supervised learning, the study highlighted that adopting systematic strategies such as active learning, crowdsourcing with quality control, or semi-supervised learning, can help mitigate these challenges.

Another study investigating data labeling for supervised learning was conducted by Shelar et al. (2020). The study focused on the use of NLP for information extraction from unstructured text, with a particular emphasis on NER. NER plays an essential role in semantic analysis, as it allows for the identification of combinations of entities that constitute a single meaningful unit. These entities can be anything from a geographical location to time, it is also possible to include custom entities or use more developed NER models for complex extraction tasks. The paper examined several established NLP libraries: `spaCy`,

`Apache OpenNLP` and `TensorFlow`. All of which can be adjusted depending on the task. To evaluate these tools, the authors compared them in terms of time, accuracy and overall performance, using IBM's dataset from Advanced Search Panel. Upon evaluation, the authors found that spaCy outperformed both TensorFlow/Keras and Apache OpenNLP. spaCy was faster than OpenNLP and equally fast as TensorFlow, but provided higher prediction accuracy than both. However, spaCy has the drawback of being a significantly larger model compared to the others which increases computational requirements. This research supports our decision to use the spaCy transformer in this work.

Chapter 3

Data

This chapter presents the data used in the study, including its strengths, limitations and the novel approach developed to address the labeling issue. The chapter is structured by firstly presenting the original datasets with a brief description of their characteristics in Section 3.1 for the textual part and Section 3.2 for the numerical. The bulk of the chapter is in large dedicated to our method of dealing with the labeling issues inherent in the original data, in Section 3.3 we describe the methodology, construction and evaluation of our labeling solution, as well as the data pipeline related part of the sentiment scoring as well as the incorporation of the numerical stock data. Finally Section 3.4 presents an overview of the final dataset including the temporal scope and overall descriptive statistics.

3.1 FNSPID: Financial News and Stock Price Integration Dataset

The primary data source for this thesis is the *Comprehensive Financial News Dataset in Time Series* (FNSPID) dataset created by Dong et al. (2024) which consists of two parts: Firstly a textual component comprising of financial news articles and secondly a numerical component of historical stock prices and trading volumes[1]. Both parts are in time series format covering the period from 1998 to 2024, concern the same 4775 S&P500 companies and are time aligned with each other. The textual part of the dataset contains 15.7 million news articles arranged by date and symbol of the related stock. Along with the articles and their titles, summarizations of the articles, made using various methods such as Luhn and LSA are provided sporadically. In addition, information such as URLs, author and publisher are sometimes

[1]Due to issues with some stocks in the dataset, the numerical part was substituted in its entirety with an alternative data source described below.

available but are considered outside the scope of this study. This two part structure is designed to be used for sentiment analysis and is ideal for our application, as we can determine sentiment for articles relating to a specific stock and then connect them via date to stock price data.

As the original dataset concerns 4775 stocks, we decided to restrict our scope to only investigate a group of stocks commonly referred to as the MAG7 (Wang, 2025), which are large American companies in the technology industry. Another unifying trait of these companies is that they have all seen major success in terms of evaluation of their stock over the last couple of decades, and have been frequently discussed in financial news, which makes them particularly interesting for our thesis.

Table 3.1: Magnificent Seven Stock Tickers

Stock	**Stock Ticker**	**Historical/Alternative**
Apple	AAPL	
Microsoft	MSFT	
Nvidia	NVDA	
Alphabet	GOOGL	GOOG
Amazon	AMZN	
Meta	META	FB
Tesla	TSLA	

3.2 Refinitiv Eikon

In order to substitute the numerical component of FNSPID, the relevant data was instead retrieved from a Refinitiv Eikon terminal (Refinitiv, 2025), which is a financial data platform that was accessed through the LUSEM library Data Science Lab on 28th of April 2025.

The numerical part of FNSPID was originally collected using the Yahoo Finance API and is divided into individual time series stored in CSV format and contains data on a daily scale with trade volume, opening price, high, low along with closing and adjusted closing price which takes dividends into account. Thus when extracting the data from the Eikon Terminal instead, we mirrored this structure but only included the variables that were used in the final models: *Date*, *Volume* and *Closing Price*.

3.3 Data Cleaning, Labeling and Preprocessing

The textual component of FNSPID also had multiple issues, particularly related to labeling, as the labels were found to both miss relevant articles and be up to 28% inaccurate in some cases. The recall issues were most prominent when looking over a wider time span, as sometimes articles assigned to a certain company only existed for a year or two but others have articles covering decades. The amount of missed articles and overall label quality also varied from stock to stock, for instance, the original assignment for NVDA was both very accurate and covered a long time period.

The original pipeline labeled the articles by filling in the blank of this URL:

`https://www.nasdaq.com/market-activity/stocks/BLANK/news-headlines`

and then cycling through a predetermined list of stock symbols such as "A", "AA", "AAPL", and so on, saving the article titles in separate CSV files per stock. This method was effective in the sense that it delegated the labeling to the NASDAQ website, but it resulted in many articles being mislabeled, especially when the articles concerned multiple stocks. The information of which stock the article related to was stored in the column *stock_symbol* (henceforth referred to as *Original Assignment*) and upon minor investigation revealed glaring issues in temporal coverage. For instance, when examining the time period of articles relating to AAPL, only the years 2020-2023 were found. However when searching for string matches in article titles only using the word "Apple", titles between the period 2009-2024 were found. The original pipeline therefore successfully gathered over 15 million articles over the years of 1998 to 2024 but had issues with labeling, which this thesis intend to combat.

3.3.1 Labeling

The overall labeling strategy is illustrated in Figure 3.1. It considers the original assignment, two keyword-based labeling methods and one NER-based labeler as weak indicators of association, which are combined in an ensemble fashion to create a new labeled dataset by considering article titles. This strategy is further elaborated on and evaluated in this and the following subsections. This subsection describes the overall processes and explains which labeling methods were used.

The label methods used in addition to the original assignment of labels provided by the dataset, included a fuzzy labeler, a hard labeler and a labeler based on a NER transformer model. These will henceforth be referred to as fuzzy, hard and spaCy NER, or simply spaCy. The reasoning behind this approach is

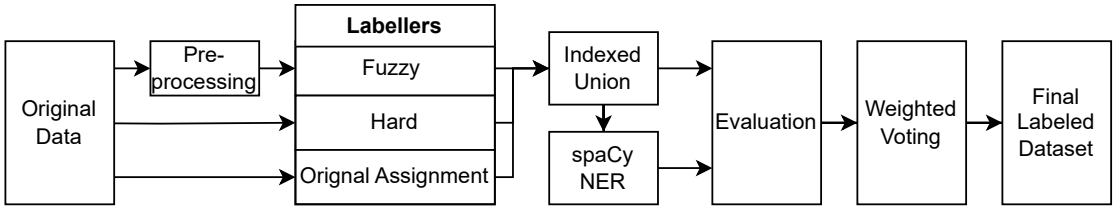

Figure 3.1: Labeling methodology

to combine the recall of more general label methods with the precision of more specific labeling methods. The fuzzy labeler was constructed using the `RapidFuzz` Python package (Bachmann, 2025), where string similarities between article titles and a predefined list of relevant keywords (see Table B.1) with a similarity score above 85[2] were considered a match by the *Partial Ratio Scorer* (PRS). The PRS attempts to find the optimal alignment between two strings and returns a similarity score as a float between 0 and 100. As the scorer is sensitive to casing and word order and can be affected by such things as punctuation, preprocessing was recommended by the package developers. Thus, before applying the function, the data was preprocessed by removing punctuations, forcing lowercase and removing common English stopwords, as described in Marcucci (2024, pp. 425–468). The other keyword-based labeler was a hard labeler which simply checks the article titles for exact matches with the predefined keyword list (Table B.2) which in turn is the same as the one for the fuzzy labeler but adapted to be able to capture differences in casing. The final labeler used is the transformer-based NER model as designed by Honnibal et al. (2020) via the spaCy library. spaCy is an open-source NLP library that offers a range of production-ready NLP tools. In terms of NER models, several options are available, but as discussed more thoroughly in Section 2.3, the transformer based model was used.

The innate nature of the label methods enabled us to save considerable time by allowing the spaCy NER model to only consider the articles that had already been flagged as relevant by either of our keyword-based labeling methods or the original assignment. The resulting dataset is referred to as the indexed union. The keyword-word based label methods thus served a dual purposed, beyond assigning associations, they also acted as a funnel with the fuzzy labeler maximize recall, the spaCy NER model to maximize precision and the hard labeler being somewhere in between. This behavior is apparent in Figures B.1 - B.7, which exhibited a hierarchical pattern among the labeling methods for all of the stocks. Another interesting observation shown in the figures was the limited time span covered by the original labels.

[2]A similarity score of 85 was empirically chosen through manual evaluation in order to cast a wide enough net without adding too much bulk to the datasets.

3.3.2 Evaluation and Ensemble Model

Evaluation was conducted using human intelligence. The purpose was to estimate the precision of our labeling methods by reviewing samples of label assignments [3]. More formally, our strategy was as follows: let D_s be the set of all articles that any of the label methods fuzzy, hard or the original assignment tagged as being about stock s and $N = |D_s|$ denote the population size for stock s representing the total number of articles for which at least one labeler considered the article relevant to stock s. We then drew a simple random sample $\{x_1, x_2, ..., x_n\} \subset D_s$ of size n, where $n = 50$ in our case. Each article x_i was assigned a binary value: $y_i = 1$ if the label was correct, or $y_i = 0$ if it was incorrect. We thus constructed the unbiased estimator $\hat{p}$ of the true accuracy p for each labeler as the sample mean, that is, the probability that the labeler assigned a correct label to a stock-relevant article. This also provided the sampling variance, where the second term corrected for the finite population.

$$\hat{p_L} = \frac{1}{n}\sum_{i=1}^{n} y_i \quad and \quad Var(\hat{p_L}) = \frac{\hat{p_L}(1-\hat{p_L})}{n} \cdot \left(\frac{N-n}{N-1}\right).$$

Furthermore, we calculated the confidence interval as

$$\hat{p_L} \pm z_{a/2} \cdot \sqrt{\frac{\hat{p_L}(1-\hat{p_L})}{n}}.$$

The raw result of the evaluation are presented in Table 3.2, and relevant evaluation statistics are shown in Table B.4. In general, however, we found that the accuracy of the original assignment varied significantly, from 72% for Amazon to 100% for NVIDIA which further emphasized the need for re-labeling. Aside from that, the labeling methods performed as expected, with the fuzzy labeler was overall the least accurate, but with showed greater variability across stocks (32% for NVIDIA and 72% for Alphabet), while the hard labeler achieved consistently high precision (84% for Microsoft to 98% for NVIDIA). The spaCy NER labeler was perfectly precise, achieving 50/50 accuracy on all evaluated stocks. It should be stressed at this point, that while spaCy assigned correct labels (true positives) in all of the considered cases, its restrictive nature increases its chance to miss relevant labels (false negatives). This recall was difficult to measure directly, but a rough estimation could be made by examining Figures B.1 to B.7.

We constructed our final dataset by designing an ensemble model based on the individual performance of each labeler on a specific stock. This was done by defining a weighted vote function such that for every ith article in $|D_s|$, where $s \in \{AAPL, MSFT, ...\}$, we calculated a score as

[3] For a sum total of 1400 article titles to be manually evaluated. See Section B.1 for some interesting examples of this process.

Table 3.2: Manual Evaluation of Labeling Accuracy (Correct Labels out of 50)

Stock	**Original Assignment**	**Fuzzy**	**Hard**	**spaCy**
AAPL	37	29	45	50
MSFT	45	17	42	50
AMZN	36	35	43	50
GOOGL	37	36	48	50
NVDA	50	16	49	50
META	37	25	47	50
TSLA	42	26	48	50

$$score_i^s = \sum_{k=1}^{K} w_k \cdot 1(y_i^{l_k} = s)$$

where $1(\cdot)$ denotes an indicator function (1 if true and 0 otherwise) and $score_i^s \in [0, \sum w_k]$ represented the confidence score for article i being about stock s. A decision was then made that if $score_i^s$ exceeded the threshold $\tau \in [0, \sum w_k]$ then the article i would be included in the final dataset. The weights for each stock/labeler combination were normalized such that $\sum w_k = 1$, and are individually shown in Table B.5. The threshold τ was set to 0.4. The choice of the threshold was ultimately heuristic, intended to balance between recall and precision. It allowed the spaCy model to dominate while still allowing the other models to push the score over the threshold if they were sufficiently accurate. A visual representation of the model's application is provided in Figures B.1 to B.7. These illustrate that, in scenarios where the keyword-based labeling methods were accurate, the model most frequently selected the hard labels. Conversely, in cases where they were not, the model defaulted to the labels chosen by spaCy. The overall size reduction of the dataset(s) at each step is illustrated in Figure 3.2. The figure shows that the union of the three naïve labelers (original assignment, fuzzy and hard) reduced the size drastically, with the ensemble labeler refining the size further.

3.3.3 Sentiment Scoring and Concatenation with Numerical Data

With the final datasets constructed, we obtained one dataframe per stock containing a unique article index and the date on which the article was published. This meant that an article can appear once per stock but is not exclusive to that stock. It also allowed flexibility in using article titles, summaries, or full articles as the basis for sentiment scoring. In our case, we chose to consider both the title and full

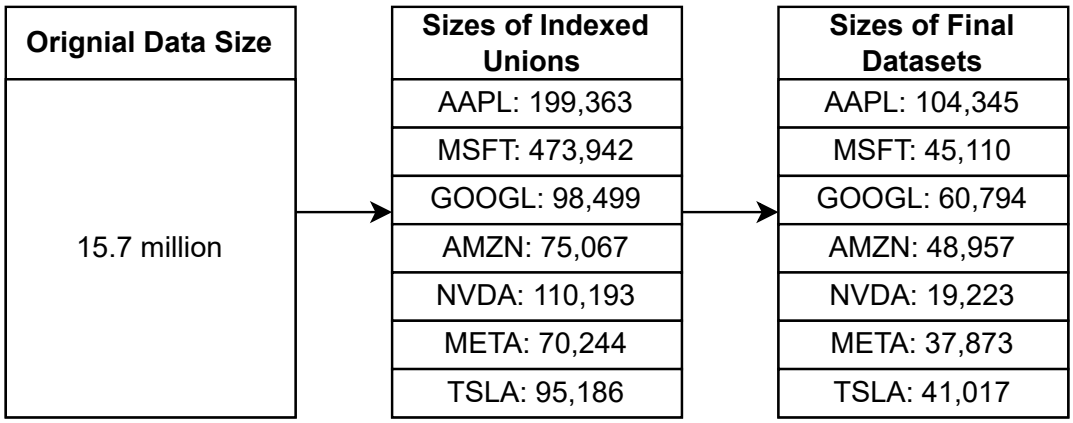

Figure 3.2: Size Reduction Visualized as Number of Articles in the Dataset(s) in Each Step of Our Labeling Methodology

text as a singular string in order to prevent cases of missing data if the article body was non existent. An example of this data structure is shown in Table 3.3.

Table 3.3: Penultimate Data Structure per Stock (NB: Example Data)

Article Index	**Date**	**String**	**Sentiment**
124205	2003-05-23	Title: (...) Article: (...)	4
245	2003-05-23	Title: (...) Article: (...)	3
53949	2003-05-25	Title: (...) Article: (...)	1
...	...	...	...

The numerical part of the data treated each day as an observation, with corresponding price and volume data. In order to successfully merge these two datasets, we assigned a sentiment score per day. For days where multiple articles affected the sentiment, the day was flagged as updated and assigned an averaged sentiment score. For days with no articles, the day was flagged as not updated, and the sentiment was decayed until the next day with an update. This structure is again illustrated with fictional data in Table 3.4. Note how the sentiment was averaged for $2003-05-23$ and then decayed until next sentiment update on $2003-05-25$.

Table 3.4: Final Data Structure per Stock (NB: Example Data)

Date	Volume	Closing Price	Sentiment Score	#Articles	Updated
2003-05-23	440502	213.00	2.5	23	1
2003-05-24	304000	211.00	2.121	0	0
2003-05-25	800023	201.00	1	203	1
...	...	...	...	...	...

3.4 Descriptive Statistics

In this section, we present our dataset in several tables of descriptive statistics related to the variables used in the thesis. Firstly, Table 3.5 shows the date-range of the stocks, where the availability of closing price data determined the start date, while the end date was artificially harmonized as the last available data point of 2023. The table also includes the mean, standard deviation, minimum and maximum values of the dependent variable closing price. Table 3.6 shows the traded volume of each stock. Table 3.7 presents the mean and standard deviation of the sentiment scores generated by our sentiment models; note that the minimum and maximum value for all sentiment scores were 0 and 5, respectively. Finally, Table 3.8 shows how many articles remained after the end-date harmonization (which explains the difference from Figure 3.2) and how many of the dates in the dataset corresponded to an updated sentiment score, along with descriptive statistics of article counts.

Table 3.5: Descriptive Statistics of Dates and Stock Prices

	Date Range		Closing Price			
Stock	Start	End	Mean	Std	Min	Max
AAPL	2006-01-03	2023-12-26	47.475	53.972	1.810	198.110
GOOGL	2005-05-02	2023-12-26	43.266	38.606	5.560	149.840
META	2012-05-18	2023-12-26	157.490	90.749	17.730	382.180
MSFT	2005-04-28	2023-12-26	92.849	95.967	15.150	382.700
TSLA	2010-06-29	2023-12-26	70.312	101.379	1.050	409.970
NVDA	2005-04-28	2023-12-26	5.546	9.824	0.150	50.410
AMZN	2005-04-28	2023-12-26	48.166	54.082	1.300	186.570

Table 3.6: Descriptive Statistics of Trading Volumes

	Trading Volume			
Stock	**Mean**	**Std**	**Min**	**Max**
AAPL	385858338.536	387643101.747	24048344	3373059549
GOOGL	64431068.913	62716598.959	9312760	823657780
META	30645940.421	26447841.865	5467488	580587742
MSFT	45123561.635	28405226.355	7425603	591078581
TSLA	96876509.389	80144525.730	1779212	914082284
NVDA	550569387.311	324669522.568	45645120	3692927840
AMZN	107450731.335	90032640.443	17626740	2088091780

Table 3.7: Descriptive Statistics of Sentiment Scores by Model and Stock

	FinBERT		RoBERTa		VADER	
Stock	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
AAPL	4.167	0.834	2.836	0.528	3.561	0.935
GOOGL	4.215	1.022	2.793	0.636	3.477	1.034
META	3.972	1.231	2.877	0.827	3.705	1.139
MSFT	4.074	1.096	2.770	0.628	3.727	1.080
TSLA	4.052	1.105	2.795	0.684	3.660	1.139
NVDA	3.377	1.291	2.724	0.680	3.417	1.274
AMZN	3.966	1.127	2.869	0.694	3.800	1.095

Table 3.8: Daily Article Count Statistics and Sentiment Update Proportion per Stock

	Article Count Statistics				Sentiment Updated		Total
Stock	**Mean**	**Std**	**Min**	**Max**	**True**	**False**	**Articles**
AAPL	21.509	27.612	0	816	0.950	0.050	97 348
GOOGL	12.075	26.553	0	779	0.921	0.079	56 703
META	9.029	18.723	0	369	0.921	0.079	26 364
MSFT	9.065	12.968	0	372	0.890	0.110	42 584
TSLA	10.618	26.203	0	598	0.925	0.075	36 069
NVDA	3.738	17.092	0	430	0.469	0.531	17 560
AMZN	9.536	21.684	0	596	0.858	0.142	44 798

Chapter 4

Methodology

In this chapter the methodology of the thesis is outlined. In Section 4.1, we present the technical aspects of the sentiment generating models, while Section 4.2 explains the predictive models in more formal terms. Section 4.3 describes the metrics that are considered when evaluating our models are described. Finally, the training process and a presentation of relevant hyperparameters are detailed in Section 4.4. In short, the new articles are transformed into sentiment scores by three different models, and combined with the numerical dataset before being passed through three different prediction model. The output is then evaluated by regression metrics and with pseudo-classification metrics based upon directionality.

4.1 Sentiment Analysis

In this section, we discuss the technical details of the sentiment scoring models, including FinBERT, presented in Section 2.1. We also introduce another common transformer-based sentiment model not trained on financial data namely RoBERTa, along with a traditional rule-based sentiment model called VADER. Furthermore, we address how we handled cases where multiple articles related to a stock on the same day, as well as cases where no articles were available.

4.1.1 Sentiment Generating Models

In order to make use of the news articles, we convert the text into a numeric value that indicates whether the article conveys a positive, neutral, or negative sentiment toward a stock. This is done by generating

what is called a sentiment score, typically ranging from [-1,1], where -1 is negative, 0 is neutral and 1 is positive. In our work, we rescale this range to [0,5] to enable sentiment decay, which is expanded upon below.

FinBERT

FinBERT is a BERT-based model architecture, but specifically constructed for financial NLP tasks, and the primary goal is polarity or sentiment analysis of documents, that is negative, neural or positive. FinBERT differs from BERT depending on task at hand: for a classification task, a dense layer is added on top of the final the hidden state, while regression tasks require a different loss function and use MSE is used instead of cross entropy. The main difference between FinBERT and BERT, however, lie in fine-tuning the training strategy to mitigate catastrophic forgetting by implementing techniques that adjusts the learning rate and layer influence. To adjust the learning rate slanted triangular learning rate and discriminative fine-tuning is used to shape the learning process to follow a slanted triangle while discriminative fine-tuning allows the learning rate to decrease when entering lower layers. To adjust the layer influence, gradual freezing was used which starts with all layers except for the final dense classification layer to be frozen and during training layers are then gradually unfreeze starting with the top layer. FinBERT is trained on three different financial corpora, namely *TRC2-financial*, *Financial PhraseBank* and *FiQA Sentiment* to ultimately promote the models vocabulary (Araci, 2019).

RoBERTa

Robustly Optimized BERT Pretraining Approach (RoBERTa) is developed as a replication aiming at generating a new pretrained language model based on the original BERT architecture. The quality and quantity of training data are key when developing a language model, as they significantly influence the model's vocabulary and the probability distributions that inform its prediction (Liu et al., 2019). To improve performance, RoBERTa introduced several modifications to the original BERT training procedure. Firstly, RoBERTa was trained on a significantly larger dataset containing a total of 160GB of uncompressed text originating from five different corpora, compared to BERT's 16GB of uncompressed text. It also introduces dynamic masking instead of static masking and thus generating new masking patterns during each training epoch. The objective of next sentence prediction was removed as it was deemed unnecessary for effective training. Lastly, the training procedure was conducted over longer sequences, with larger batch sizes and greater number of training steps compared to BERT.

VADER

Valence Aware Dictionary and sEntiment Reasoner is a rule-based approach sentiment analysis tool that assigns sentiment scores based on syntactic and grammatical heuristics. Each feature in the model, such as punctuation, capitalization, adverbs, conjunctions, emojis, acronyms, and slang, is associated with a valence score ranging from $[-4, 4]$. These heuristics include common features such as punctuation, capitalization, adverbs, conjunctions but also including western-style emojis or emoticons, acronyms and slang. In total the dictionary consists of over 7,500 features that is scored within the previously mentioned range. When applied to a document, the text is tokenized and assigned a valence score if it exists in the dictionary. The negative, neutral and positive scores are then summed and provides a sentiment score.

4.1.2 Sentiment Averaging and Decay

As this thesis deals with both textual data in the shape of news and financial data in the shape of trading volume and price, it is important to acknowledge the possible difference in recorded frequency. If a direct concatenation were performed based on the frequency of news and their sentiment, it would likely result in null values; conversely, concatenation based on financial data would lead to mismatched dimensions or data loss. To avoid this issue, we chose to average the sentiment scores derived from financial news, thereby aligning a single day sentiment score with the corresponding financial data for that day.

Furthermore, when working with these daily average sentiment scores, we follow the reasoning of Simkin & Roychowdhury (2014) in asserting that sentiment unlikely to remain constant over time. That is, the mood reflected in news sentiment on day 1 is unlikely to fully represent the sentiment on day 5. Therefore a decaying factor of FACTOR was applied to the sentiment score exceeding $|2.5|$ to decay or increase towards 2.5 which is the neutral sentiment mood towards the underlying company and asset. The following function is applied:

$$Score_t = 2.5 + \left(\frac{Score_{(t-1)} - 2.5}{t^{1.5}} \right). \tag{4.1}$$

Table 4.1: Excerpt of dataset showing daily close prices and sentiment scores for AAPL in January 2006.

Date	Close	Volume	Symbol	VADER	FinBERT	RoBERTa	#Articles	Updated
2006-01-06	2.72	704,558,074	AAPL	2.5	2.5	2.5	0	False
2006-01-09	2.89	675,445,571	AAPL	2.5	2.5	2.5	0	False
2006-01-10	2.89	2,280,355,264	AAPL	2.5	5.0	2.5	1	True
2006-01-11	3.00	1,494,197,022	AAPL	2.5	5.0	2.5	0	False
2006-01-12	3.01	1,280,809,145	AAPL	2.5	3.38	2.5	0	False
2006-01-13	3.06	776,614,349	AAPL	2.5	2.98	2.5	0	False
2006-01-17	3.03	836,861,897	AAPL	2.5	2.81	2.5	0	False
2006-01-18	2.95	1,213,119,217	AAPL	2.5	2.72	2.5	0	False
2006-01-19	2.82	1,696,401,800	AAPL	2.5	2.67	2.5	0	False
2006-01-23	2.72	1,059,771,408	AAPL	2.5	5.0	2.5	1	True

4.2 Model Architecture

In this section, the models introduced in Section 2.2 are described in more formal terms. We begin with the original Transformer model, outlining mathematical foundations of the attention and self-attention mechanisms, followed by the multi-head attention, positional encoding, and the assembly into an encoder-decoder structure. Subsequently, we discuss the technical aspects of two Transformer variants, namely the Informer and PatchTST.

4.2.1 Vanilla Transformer

Transformers enable sequence modeling without relying on either convolutions or recurrent structures. This allows for effective modeling of long-range dependencies, as well as parallelization where the latter allowing for much faster computation with modern *Graphical Processing Units* (GPUs). The following description of their technical detail is based on Soydaner (2022) as well as Ghojogh & Ghodsi (2020). At the core of the transformer architecture lies the attention mechanism, first popularized by Bahdanau et al. (2014), which allows the context of an entire input sequence to be considered, but with strong focus on a particular part of the input. These attention functions can be split up like in Vaswani et al. (2017) to form multi-head attention which can capture different types of relationships and allow for increased parallelization. To preserve sequential information, each object or token in the sequence in question is assigned a positional encoding, typically based on sine and cosine wave functions. These components,

attention, positional encoding, feed-forward networks, and normalization layers are combined into encoder-decoder stacks, which allows for deep hierarchical representation.

Attention and Self-Attention Mechanisms

Attention can be understood as a weighted average over a sequence, allowing the model to prioritize the most relevant parts. These important components often reflect emphasis or context of a sentence, a classic example is whether or not the *model* is walking down a runway or is optimized using backpropagation. What these important parts are and how they are decided, is described below.

Most commonly, attention is calculated using a *Query-Key-Value* (QKV) setup, which mirrors how a database could function: A query is posed, if it matches with a key, the corresponding value is returned. In this simple database there is only one query-key-value combination, but in the case of attention we are more interested in how related or *similar* the key and query combinations are (local context). These are then normalized with the softmax function which can then be combined with the value (or global context) by calculating the dot product. This process can be formally written as

$$Attention(q, \{k\}_{i=1}^{n}, \{v\}_{i=1}^{n}) := \sum_{i=1}^{n} (a_i) v_i \tag{4.2}$$

where $a_i := softmax(s_i) = \frac{exp(s_i)}{\sum_{k=1}^{n} exp(s_k)}$, and $s_i = similarity(\mathrm{q},\mathrm{k}_i)$. Or more neatly expressed in matrix from as:

$$Attention(Q, K, V) = softmax\big(similarity(Q, K)\big)V. \tag{4.3}$$

Some similarity measure is thus needed to determine how related key and query are. One example is the scaled inner product as used in Vaswani et al. (2017), which they named the *Scaled Dot-Product Attention* and is there formulated as

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4.4}$$

where d_k denotes the dimensionality of the queries and keys. The reason for scaling the dot products by $\frac{1}{\sqrt{d_k}}$ is to counteract the gradient vanishing issue, which can occur when dot products grow excessively large in magnitude. So far the function has considered the local query-key context, but as the input

sequence is scaled with the softmax function, each similarity score $\in [0, 1]$ is multiplied with the global value matrix it enables the model to attend to long-range dependencies as well. Note that the form of attention described here is commonly referred to as *Self-Attention*, as it describes a scenario where the queries, keys and values originates from the same words in the sequence, which is in contrast to cross-attention that compares two different sequences.

Multi-Head Attention

The attention function can be split up into some number of heads, commonly $h = 8$, and computed in parallel which creates what is known as the *Multi-Head Attention*. The primary motivation for this design is to allow a particular sequence to be evaluated in multiple ways, as each head has separate projections of Q,K,V capturing more semantic detail by considering different perspectives. Formally, multi-head attention are a concatenation of a single attention function linearly projected h different times, or

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O)$$

where each attention head is computed as: $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$ and thus get a single W^O, or the weight describing the linear transformation of the resulting concatenated head. The process is illustrated in Figure 4.1, where the sequence is fed into h linear transformations of Q, K and V, considered separately and then combined.

Another benefit of splitting up the attention calculations into smaller parts is the boon to computational efficiency that can be gained. Firstly by considering a smaller dimensional space as $d_k = \frac{d_{model}}{h}$. Secondly by enabling further parallelization which increases performance in particular when utilizing modern GPUs.

Positional Encoding

Since the transformer model discards both recurrence and convolution, it requires an alternative mechanism to incorporate positional information about elements in the input sequence. The most common approach of doing this is utilizing sine and cosine functions such that

$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{2i/d}})$$

and

$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{2i/d}})$$

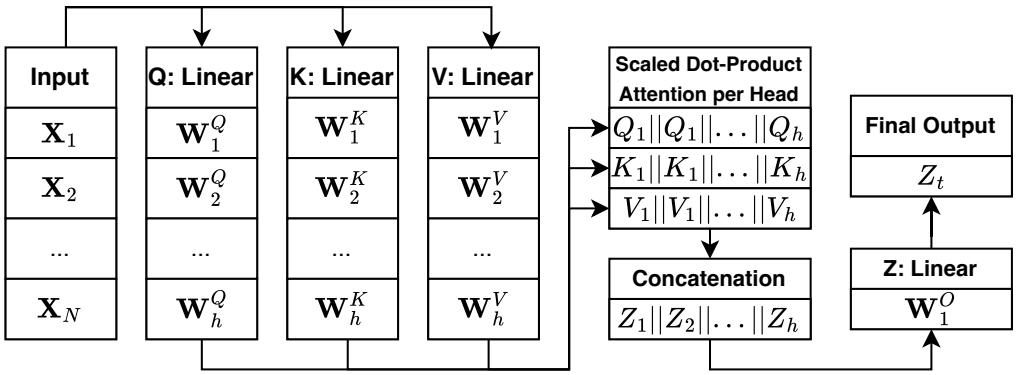

Figure 4.1: Schematic representation of the *Multi-Head Attention* mechanism based on Ghojogh & Ghodsi (2020)

where *pos* denotes the position in the sequence, i is the dimension index, and d is the model dimensionality. These functions produce position-specific encodings that can be added to the input embeddings, allowing the model to infer order and relative distance. This formulation enables the model to retain positional information in a way that is both continuous and generalizable to unseen sequence lengths. Moreover, since sine and cosine functions are periodic, they naturally encode relative positions, which is useful for sequence modeling. The encodings can either be precomputed using fixed functions (as shown above), or learned during training. In either case, the positional information is added to the input embeddings at the base of the network and remains constant throughout the forward pass, ensuring that positional context is preserved even in the absence of recurrence:

$$x_i \leftarrow x_i + PE(i).$$

This approach to positional encoding was introduced by Vaswani et al. (2017).

Encoder-Decoder Structure

At its core, a transformer model can be described as a sequence-to-sequence model and share many of their properties. An input sequence is encoded into some abstract space and is then reconstructed by the decoder. An example of the structure is shown in Figure 4.2, which follows the input's path through the encoder and decoder blocks before being passed into the output layer.

The input sequence first undergoes positional encoding, as described above, and the result is a combined

representation that preserves both content and position. This input sequence is then fed into the encoder, which consists of a stack of identical blocks. Each encoder block begins with a multi-head attention mechanism, allowing the model to attend to different parts of the sequence simultaneously. Following attention, the encoder applies a residual connection (where the input bypasses blocks and is added with the layer output) and layer normalization (which normalizes inputs across features), which helps gradient flow and training stability. The subsequent feedforward network applies two linear transformations with a ReLU activation in between, further transforming the representation, followed again by residual and normalization layers.

On the decoder side, the input is the shifted target sequence, which also undergoes positional encoding. The decoder block begins with a masked version of the multi-head attention layer, which ensures the model can only attend to previous positions, enforcing the autoregressive property required for generation. Next, the decoder then incorporates information from the encoder through a second multi-head attention layer, where the queries come from the decoder and the keys and values come from the encoder's output in order to align the input and output sequences. As with the encoder, residual and normalization steps follow each block. The final decoder output is passed through a linear projection layer followed by a softmax activation function, which transforms the output into a probability distribution over the output space. The result is a predicted output token or value.

The architecture shown here is a slight variation of Vaswani et al. (2017) for a numerical context but the architecture can be adapted for both discrete classification and continuous-valued forecasting tasks, such as predicting stock prices or movement direction.

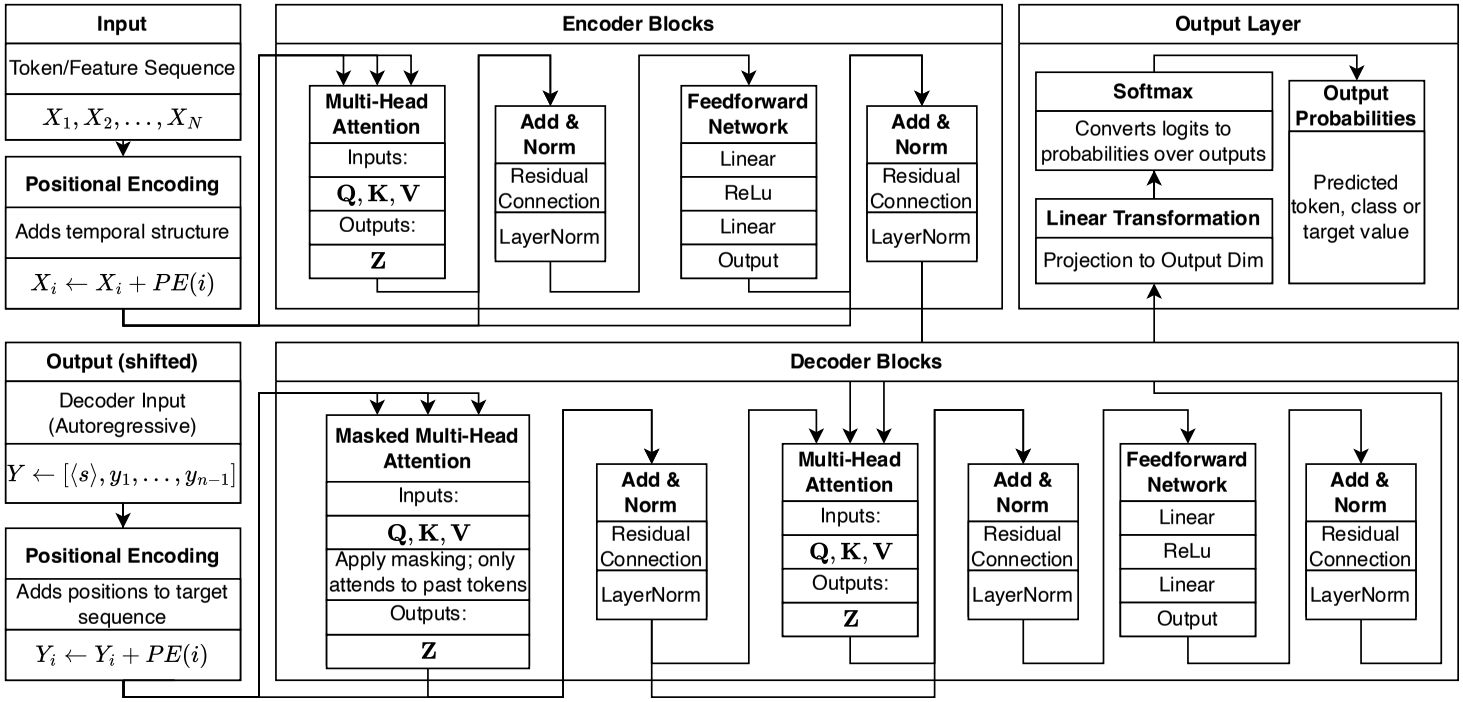

Figure 4.2: Schematic Representation of the Encoder-Decoder Structure Based on Vaswani et al. (2017).

4.2.2 Informer

Extending on the work made by Vaswani et al. (2017) was as mentioned in Section 2.2 done by Zhou et al. (2021) by addressing the issues of space and time complexity by implementing a *ProbSparse* self-attention and a distillation of it, but also a generative style decoder.

ProbSparse Self-Attention

The *ProbSparse* self-attention mechanism seeks to reduce the quadratic time complexity of traditional attention by focusing on computation only for the most informative query-key pair. Instead of computing dense attention across all query-key pairs, they allow each key $\mathbf{k}_i$ to only attend to the top-u most dominant queries, which significantly reduces computational overhead. The self-attention is defined as

$$A(Q, K, V) = softmax\left(\frac{\overline{Q}K^\top}{\sqrt{d}}\right)V.$$

Here, $\overline{Q}$ is a sparse matrix of the same shape as query q and it contains only the dominant queries of u-queries and is based on a sparsity measurement $M(\mathbf{q}, \mathbf{K})$. There is a notational difference used by Zhou et al. (2021) where d simply substitutes d_k which is used in the previously described *Transformer*-model. The number of selected dominant queries u is determined by $u = c \cdot \ln L_Q$, where c is a constant sampling factor and L_Q is the length of the query sequence. This ultimately reduces both the time and space complexity of attention computation $\mathcal{O}(L^2) \longrightarrow \mathcal{O}(L \log L)$. To select the top-$u$ queries without computing all dot-product pairs, Zhou et al. introduce a sparsity measurement via Lemma 1 (add to appendix). Based on this, the sparsity score is defined using the *max-mean* operator:

$$M(q_i, K) = \max_j \left\{\frac{q_i k_j^\top}{\sqrt{d}}\right\} - \frac{1}{L_K}\sum_{j=1}^{L_K}\left(\frac{q_i k_j^\top}{\sqrt{d}}\right).$$

This formulation avoids full pairwise dot-product computation by approximating query importance relative to its average attention, making it both computationally efficient and numerically stable. Due to the long-tail distribution of attention scores, only a small subset of queries contributes significantly, and ProbSparse leverages this by computing attention selectively. This design makes it especially effective for long-sequence modeling tasks.

Self-Distillation Mechanism

Following the ProbSparse self-attention mechanism, the encoder's feature map may still contain redundant combinations of the value matrix V. To mitigate this, a distilling operation is applied to prioritize dominant features and produce a more focused self-attention map for the subsequent layer. This operation is formally described as:

$$X_{j+1}^t = MaxPool\left(ELU\left(Conv1d\left([X_j^t]_{AB}\right)\right)\right).$$

$[\mathbf{X}_j^t]_{AB}$ denotes the output of the attention block, which includes multi-head ProbSparse self-attention along with a convolutional filtering. The Conv1d operation applies a one-dimensional convolution across the time dimension using a kernel of width 3, followed by an *Exponential Linear Unit* (ELU) activation. A max-pooling layer with stride 2 is then applied to downsample the feature map, effectively halving its temporal length. This reduces memory and computation, achieving a complexity of $\mathcal{O}((2-\epsilon)LlogL)$, where ϵ is a small constant. To improve the robustness of the distilling process, multiple encoder stacks are constructed with halved inputs, each having one fewer distillation layer than the previous. This results in a pyramid-like structure where all output dimensions are aligned. Finally, the outputs from all stacks are concatenated to form the encoder's final hidden representation.

Generative Decoder and Forecasting

Zhou et al. (2021) adopt a standard decoder structure originating in the Transformer model (Vaswani et al., 2017), which consists of two identical multi-head attention layers stacked. Their generative inference mitigates the speed plunge that can occur in long-run prediction. The input vectors which enters the decoder are formally described as

$$X_{de}^t = Concat(X_{token}^t, X_0^t) \in \mathbb{R}^{(L_{token}+L_y)*d_{model}}. \tag{4.5}$$

Here, X_{token}^t is a start token, X_0^t serves as a dummy vector that occupies the decoder input space prior to generation. To enforce the autoregressive property and prevent future positions from being attended to, a masked multi-head attention is added to the self-attention mechanism by setting the masked dot products to $-\infty$. Finally, a fully connected layer receives the output and the size of it, d_y, depends on the feature setting being univariate or multivariate. Instead of using a fixed start token for decoding, the Informer model employs a generative approach by sampling a past segment of the input sequence as a start token

which typically is a known sequence from earlier in time (5 days before the prediction window). This segment, along with a timestamp vector representing the target period, forms the decoder input. Unlike traditional dynamic decoding that predicts step-by-step, Informer performs prediction in a single forward pass, making it faster and more efficient for long-term forecasting.

4.2.3 PatchTST

In the previously mentioned work by Nie et al. (2022) in Section 2.2 they introduced two new applications to transformers to improve on predictive accuracy and reduction of time and space complexity in a time-series setting. This improvements include a technique called patching and, in a multivariate setting, the introduction of channel independence.

Patches

Transformers have a quadratic time and space complexity with respect to the number of input tokens N, i.e., $\mathcal{O}(N^2)$. For time series forecasting, the input sequence length L typically determines N, since each time step is treated as a token. This makes the model computationally expensive for long sequences. Nie et al. (2022) proposed a patching strategy which groups consecutive time steps into patches of length P, and extracting them with a given stride S, which defines the step size between patches. This results in a reduced number of tokens by $N = \left\lfloor \frac{L-P}{S} \right\rfloor + 2$. This reduces the complexity to $\mathcal{O}\left(\left(\frac{L}{S}\right)^2\right)$ while still allowing the model to encode a long look-back window.

Using the Microsoft stock as an example, the original input sequence had a length of $L = 4\ 637$, corresponding to $N = L$ tokens in the basic transformer setting. This would results in a baseline complexity of $\mathcal{O}(N^2) = \mathcal{O}(4\ 637^2) = 21\ 501\ 769$. By applying patching with patch length $P = 16$ and stride $S = 8$, the number of input tokens becomes approximately $N = \frac{4\ 637-16}{8} + 2 \approx 580$, and the resulting complexity is thus $\mathcal{O}(N^2) = \mathcal{O}(580^2) = 336\ 400$. Which represents a relative complexity reduction of $\frac{336\ 400}{21\ 501\ 769} \approx \frac{1}{63}$. This confirms a reduction in complexity resulting in a substantial improvement in computational efficiency, without sacrificing the model's capacity to learn from long-term temporal patterns.

Channel Independence

Another addition to the research of transformers performed by Nie et al. (2022) was to introduce channel-independence in transformer-models. In a time-series task, involving several variables means dealing with a multi-channel signal, where each input token can either mix channels or treat them independently. With channel-mixing each input token is represented by a row vector, which allows the model to use information from all channels at a certain time step. With channel-independence however, every channel is treated as a univariate time series. Each univariate time series $x^{(i)} \in \mathbb{R}^{1\times L}$, where $i = 1, \ldots, M$, is then transformed into a patch sequence $X_p^{(i)} \in \mathbb{R}^{N\times P}$, which is then independently processed by the model.

Table 4.2: Example of channel independence

Time-step	**Channel-Mixing Token**	**Channel-Independence Tokens**
$t = 1$	$[\, x_1^{(1)}, x_1^{(2)}, \ldots, x_1^{(M)} \,]$	$x_1^{(1)},\ x_1^{(2)},\ \ldots,\ x_1^{(M)}$
$t = 2$	$[\, x_2^{(1)}, x_2^{(2)}, \ldots, x_2^{(M)} \,]$	$x_2^{(1)},\ x_2^{(2)},\ \ldots,\ x_2^{(M)}$
$\vdots$	$\vdots$	$\vdots$
$t = L$	$[\, x_L^{(1)}, x_L^{(2)}, \ldots, x_L^{(M)} \,]$	$x_L^{(1)},\ x_L^{(2)},\ \ldots,\ x_L^{(M)}$

Channel-independence allows each time series to be processed in its native temporal resolution without requiring interpolation or imputation, preserving the integrity of the data. After independent processing, each channel produces a representation summarizing its temporal patterns. These representations are then fused, for example through concatenation, to form a comprehensive joint representation used for the final prediction.

4.3 Evaluation Metrics

In this section, we describe the relevant metrics that we base the evaluation of our models on. First, some common methods for evaluating regression models are introduced, both based on mean errors, but also corresponding metrics based upon median errors. Second, directional accuracy is introduced, both as an independent metric (in the form of *Mean Directional Accuracy* (MDA)) but also as a way of discretizing the continuous output giving us access to confusion matrices and related metrics. Finally an adaptation of the *Diebold-Mariano* (DM) test is presented as a tool to compare the different stock-model combinations.

At this point, it is worth noting that we want to evaluate how the sentiment models affect the forecasting

models performance which is applied on seven different stocks, which in turn creates 42 unique stock model combinations, as $2 \cdot 3 \cdot 7 = 42$. We handle this by averaging over the stocks to reduce dimensionality to $2 \cdot 3 \cdot 1 = 6$ models. The metrics in the following subsection are easily adapted to averages using their geometric means, such as average F_1 score, this is also the case for directional accuracy which will be expanded upon below. However, the DM test will require some additional adaptation in order to be suitable for our purposes.

4.3.1 Regression Metrics

As previously mentioned the nature of the variable of interest, the first difference of the closing price, is continuous. This means that regression based metrics are a natural choice for optimization and evaluation.

Mean Metrics

In evaluating regression model performance, mean-based error metrics are among the most widely used (Botchkarev, 2018). In particular, the MAE and the MSE provide intuitive measures of average deviation between predicted and actual values. MAE captures the average absolute magnitude of prediction errors, while MSE penalizes larger errors more heavily due to the squaring term, making it more sensitive to outliers. These metrics are especially relevant when overall model accuracy is of interest, and they align well with standard optimization objectives in deep learning (Plevris et al., 2022).

Median Metrics

To complement the mean-based metrics, we also report median-based alternatives, namely, the *Median Absolute Error* (MedAE) and the *Median Squared Error* (MedSE). These metrics are less sensitive to extreme values and offer a more robust assessment of model performance in the presence of outliers or non-normal error distributions (Lee et al., 2003). In the context of financial time-series forecasting, where occasional large deviations may occur due to market shocks or news events, median metrics could potentially provide a more representative summary of typical model behavior.

4.3.2 Pseudo Classification With Directional Accuracy

Prediction in the machine learning context is typically divided into two main tasks (Goodfellow et al., 2016, pp. 98–99): *regression*, where the output is continuous so that $y \in \mathbb{R}$ (as is the case when predicting actual stock prices), and *classification*, where the output is assigned to a discrete class label.

In standard binary classification settings, the model often maps real-valued outputs to a probability space $[0, 1]$ using a function such as sigmoid or softmax, followed by a threshold τ

$$\hat{y} = \begin{cases} 1 & \text{if } d > \tau \\ 0 & \text{otherwise,} \end{cases}$$

where d is the model output and τ is the classification threshold (commonly $\tau = 0.5$). However, in our case, the models are trained to predict continuous-valued stock prices. To enable classification-style evaluation, we transform these predictions into directions (Greer, 2003). Specifically, we define the predicted class $\hat{y}_t$ based on the direction of the predicted change which can be formulated as

$$\hat{y}_t^{\text{class}} = \begin{cases} 1 & \text{if } \hat{y}_t - \hat{y}_{t-1} \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad \textit{and} \quad y_t^{\text{class}} = \begin{cases} 1 & \text{if } y_t - y_{t-1} \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

such that the true direction is defined by the sign of the actual price change. Since our target variable is the first difference of the closing price such that $y_t = \nabla ClosePrice_t = ClosePrice_t - ClosePrice_{t-1}$, we are effectively calculating the second difference of the closing price, capturing changes in direction.

This formulation, pioneered by Cumby & Modest (1987), allows us to evaluate the regression models using classification-oriented metrics such as precision, recall, F1-score, and MDA, by comparing predicted and actual movement directions. For simplicity, price changes of zero are grouped with the positive class.

Mean Directional Accuracy

MDA is a commonly used metric in financial forecasting (Blaskowitz & Herwartz, 2011; Schumaker & Chen, 2009), because it gives a natural scenario for classification. This is because it may be more interesting to look at price movement rather than exact prices, we can thus classify a positive price movement to the positive class and vice versa. Mathematically it is formalized as

$$MDA_{s,m} = \frac{1}{T-1}\sum_{t=2}^{T} 1\left[sign(y_t^{(s)} - y_{t-1}^{(s)}) = sign(\hat{y}_t^{(s,m)} - \hat{y}_{t-1}^{(s,m)})\right],$$

where s is the individual stock and m is each model combination while y_t is actual values and $\hat{y}_t$ is predicted values.

In order to reduce the number of models, we compute the average for each stock so that

$$\overline{MDA}_m = \frac{1}{S}\sum_{s=1}^{S} MDA_{s,m},$$

where S=7 in our case. Facilitating for direct comparison between the 6 models in a single-value metric.

Confusion Matrices and Related Metrics

Although our models are trained for regression, the directional framing introduced above enables extensive and intuitive evaluation using classification-oriented metrics. At the most basic level, we can compare the predicted direction of price movement with the actual direction, resulting in four possible outcomes: true positives, false positives, true negatives, and false negatives. These outcomes can be counted and arranged into a confusion matrix, as shown in Table 4.3, which provides an overview of the performance of a theoretical model.

Table 4.3: Theoretical Confusion Matrix

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

From this point, we can extract key evaluation metrics. *Precision* measures the proportion of predicted positive movements that were correct, while *recall* quantifies the proportion of actual positive movements that the model correctly identified. These are defined mathematically as

$$Precision = \frac{TP}{TP+FP} \quad and \quad Recall = \frac{TP}{TP+FN}.$$

These can be combined into a single measure, the F_1 score, which is the harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}.$$

4.3.3 Forecasting Evaluation: The Diebold-Mariano Test

The DM test has become a classic method of comparing different forecasting models since its introduction by Diebold & Mariano (1995). It is mainly concerned with testing if the difference between the cost functions of two time-series is significantly different from zero. Formally, let $\hat{y_t}^{(1)}$, $\hat{y_t}^{(2)}$ be two competing time-series targeting series y_t, and $g(\cdot)$ be some loss function. With the difference at some point time t between the series $d_t = g(e_t^{(1)}) - g(e_t^{(2)})$, where e_t is the forecasting error, which gives us the test statistic

$$DM = \frac{\overline{d}}{\sqrt{V\hat{A}R_{NW}(\overline{d})/T}} \xrightarrow{d} \mathcal{N}(0,1) \quad \text{as } T \to \infty,$$

where $\bar{d} = \frac{1}{T}\sum_{t=1}^{T} d_t$. $V\hat{A}R_{NW}$ is the long run HAC-estimator as proposed by Newey & West (1987), which adjust for potential serial correlation and heteroskedasticity in the loss series. In our setting, we adapt the test to aggregates across series, so that $d_t^s = g(e_{1,t}^{(s)}) - g(e_{2,t}^{(s)})$ and $\bar{d}_t = \frac{1}{T}\sum_{s=1}^{S} d_t^{(s)}$ which then can be utilized in the regular DM framework in order to perform pairwise tests between the final 6 models. The hypotheses are thus H_0: $\bar{d}_t = 0$ and H_1: $\bar{d}_t \neq 0$, or, whether or not the difference between the forecasting models are zero. Since we are testing three different Informer and PatchTST models, which creates $\binom{6}{2} = 15$ pairwise comparisons, we also decided to implement the *Benjamini-Hochberg* (BH) procedure (Benjamini & Hochberg, 1995) in order to control the rate of false discovery, as well as the *Bonferroni* correction (Bonferroni, 1936) to control for family-wise error rate.

Adaptations of the DM test for multiple time-series is not a novel concept, for instance Borup et al. (2019) used a variant for panel-based which accounts for cross-sectional dependence and long-range temporal correlation. While our setup is simpler, this provides a theoretical precedent for aggregating forecast performance over multiple assets, such as averaging loss differentials across stocks prior to DM testing.

4.4 Hyperparameters and Model Training

In this section, we begin by introducing the optimization method employed when training our models, Adam, which is the default choice in the repositories for our transformer models. After this we present the strategy of tuning hyperparameters: Bayesian optimization for continuous and grid search for categorical

variables using the `Optuna`-package. Finally the selection of hyperparameters and a description of the training procedure for the transformer based models is shown.

4.4.1 Optimization method: Adam

All models trained in this thesis have been optimized using the Adam algorithm, which was developed by Kingma & Ba (2014), and thoroughly described in Goodfellow et al. (2016, p. 305). Adam is a first order optimization algorithm which like its name suggests is an adaptive learning rate algorithm which means that it is able to push through local minima using momentum. This momentum is incorporated in the algorithm by a biased estimate of the first-order moment along with an (also biased) estimate of the second-order moment, before successively correcting for this bias.

Mathematically, Adam updates parameter matrix θ_t at time step t by first getting the gradients with respect to the objective function at timestep t, $g_t = \nabla_\theta f_t(\theta_{t-1})$ and then updating the biased first moment estimate $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$, and the biased second moment estimate $v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2$. The algorithm then corrects this bias for both moment estimates by computing $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$ and $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$. Finally the parameters are updated as

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

where g_t is the gradient at time t, α is the learning rate, and β_1, β_2 are decay rates for the moment estimates, and are by default set to 0.9 and 0.999 respectively. ϵ is a small constant, typically 10^-8.

Adam has seen empirical success, as discussed in Kingma & Ba (2017), for parametric models such as logistic regression with neat convex loss functions. It has also been proven to be effective when training deep learning models with non-convex loss functions such as CNNs and DNNs which demonstrates its applicability in our work as well.

4.4.2 Hyperparameter Tuning: Optuna

The hyperparameters for all models were tuned using the `Optuna` package in Python made by Akiba et al. (2019), which is framework for applying different kinds of optimization to model training in a define-by-run fashion. Usually, hyperparameter optimization frameworks are developed in the define-and-run style which means that the operator first constructs a search space and then explores the space in two

separate phases. In contrast, Optuna is built in the define-by-run style which is more reminiscent on how deep learning frameworks such as `PyTorch` is designed, which allows the user to program the parameter search space directly. Support for the define-and-run style is also available and is used in this work as it is easier to present the potential values ranges of the hyperparameter, while still enjoying the benefits of define-by-run as the search space is not actually statically defined in advance.

Practically, the algorithm tries to find the best hyperparameters by finding the model configuration which minimizes validation error (MSE in our case), it does this by doing several trials consisting of multiple epochs where different values of the hyperparameter are tested. In the Optuna framework, the number of trials are determined which decides how many tests the optimizer has to establish the optimal hyperparameter setting, in our case this number was set to 10. This translates the number of trained models to be 420 different trained models all with different combinations of hyperparameters.

In our work, we configured two transformer models which were tuned both by some common hyperparameters and tied to their specific architectures. These are shown in Table 4.4 which shows which hyperparameters that are tuned and in which value range. The tuned parameters were *Dropout* as a form of regularization in order to prevent overfitting, *Learning Rate* as an initial value passed to Adam and adjusted by the default learning rate adjustment schedule (*type 1*, which halves the learning rate every two epochs) and finally *Batch Size* which can affect training stability and convergence speed. For the Informer, we also tuned *Label Length*, which is an Informer specific hyperparameter which determine how many observations the model is allowed to peek ahead. The PatchTST models were tuned over *Patch Length* which determines how many time steps which are grouped into a single patch token, where small patches are more granular but leads to longer sequences, while large patches can capture more temporal context but reduce the sequence length. *Stride* determines if and how much the patches overlap by controlling how much to shift between patches.

Table 4.4: Transformer Hyperparameter Search Space

	Parameter	**Data Type**	**Value range**
Both	Dropout	Float	[0.05, 0.5]
	Learning Rate	Float	$[1 \times 10^{-5}, 1 \times 10^{-3}]$(log = True)
	Batch Size	Categorical	[32, 64, 128]
Informer	Label Length	Categorical	[15, 30, 45]
PatchTST	Patch Length	Categorical	[16, 32, 48]
	Stride	Categorical	[8, 10, 12]

We used Bayesian optimization for the continuous values via the default *Tree-structured Parzen Estimator* (TPE) in Optuna. This is done by supplying a prior distribution (uniform for dropout and log-uniform for learning rate, hence the log=True in Table 4.4) and instead of searching the whole parameter space, Bayesian optimization allows the estimator to continually update the posterior distribution based upon evidence from earlier epochs. For the categorical parameters however, a grid-like random search was instead employed which randomly selects which values are tried in different trials.

Both the Informer and PatchTST allows for a large variety of tunable parameters, all of which can have an impact on model performance. The full model specifications can be found in Tables C.1 and C.2, which gives an overview of parameters, some of which were left as default (such as optimizer and learning rate adjustment schedule), others were specified based on our context (such as sequence and prediction length) and finally the ones that were tuned. The tuning was performed for each stock which gives a different set of tuned parameter per stock-model combination (42 different models), which is why Table 4.5 only report the mode optimal value of each prediction model and hyperparameter.

Table 4.5: Mode of Final Hyperparameter Values

	Parameter	**Informer**	**PatchTST**
Both	Dropout	0.302	0.478
	Learning Rate	1.40×10^{-3}	3.60×10^{-5}
	Batch Size	256	128
Informer	Label Length	15	–
PatchTST	Patch Length	–	16
	Stride	–	8

4.4.3 Models

The models that were used in this work was created with the `Time-Series Library` which is an open-source library for research in deep learning, with a focus on time series analysis (Wang et al., 2024; Wu et al., 2023). The two types of transformer-based architectures, Informer and PatchTST, was trained in a setting as equal as possible to allow for fair comparison. The variables were first scaled by `StandardScaler` (not including binary variables) from the `Scikit-Learn` library. The second step of the training process was to split the complete data set into three separate data sets for training, validation and testing, and this split was made as 70/10/20 respectively. All training was conducted using a single NVIDIA A100 GPU with 40GB VRAM through Google Colab cloud computing. For reproducibility, all models described

below was trained using the same seed, with full architectures and final model specifications, with the exception of those tunable, can be found in Section 4.2. Each model was trained with a fixed setting of 30 training epochs and included early stopping with a patience of three in order to avoid the risk of overfitting. Early stopping with a patience of 3 and a maximum of 30 epochs was applied, using a learning rate scheduler of *type 1*. The learning rate scheduler refers to a separate component (in addition to Adam) which adjusts the learning rate over time, and is in this case over epochs. In all visualizations below, losses have been averaged by the number of companies at that specific epoch.

Informer

Training the Informer was made in accordance with the optimal hyper parameters described in the previous section, thus resulting in a total of 21 different models by including seven different companies and three different sentiment models. Each Informer model was trained using a consistent configuration file via the "run.py" script provided by the original repository on GitHub. The input data was processed with a sequence length of 60 and a prediction horizon of one. Key architectural parameters such as encoder/decoder layers, hidden dimensions, and attention heads were fixed across runs and are documented in Table C.1. Hyperparameters marked as tunable were optimized for each sentiment source and company individually, and their selected values are reported in Table C.3.

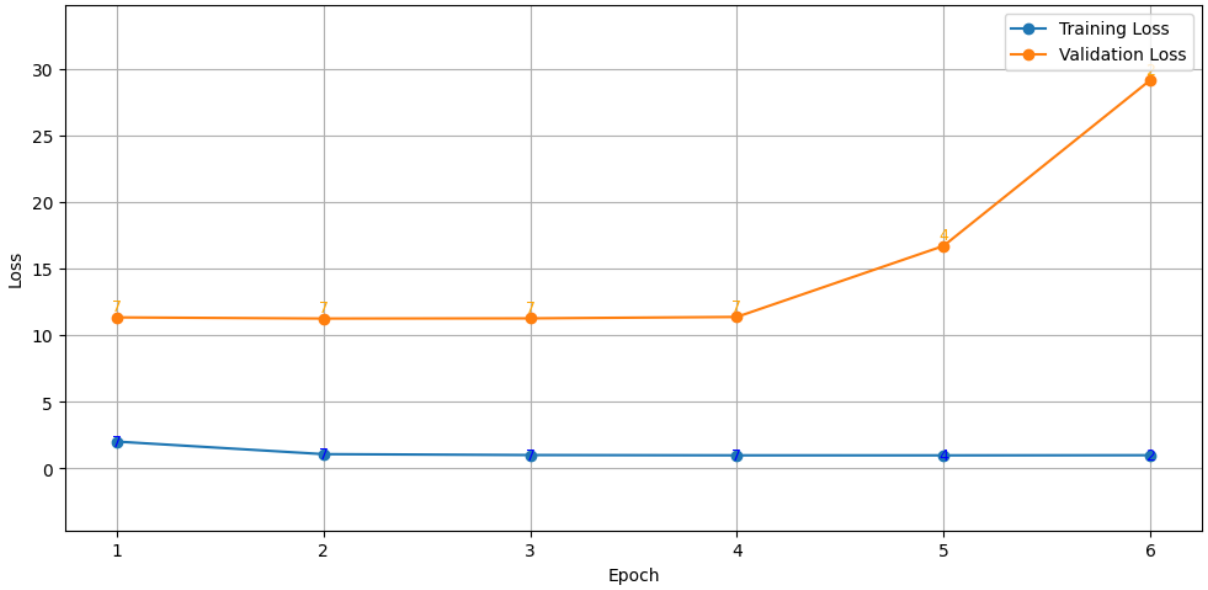

Figure 4.3: Average training and validation loss per epoch for Informer-FinBERT

The first model which was trained was the Informer-FinBERT model and in Figure 4.3 plot we can view

the training process. This model trained for a minimum of four epochs and a maximum of six epochs, with a mean runtime of 4.857 epochs before early stopping shut down the training process. It is evident from the plot that the validation and training loss does not converge, but rather diverged passing from each epoch. We can also state that after four epochs, three companies dropped and only two companies was able to run for six epochs and then validation loss rapidly increases thus indicating signs of overfitting.

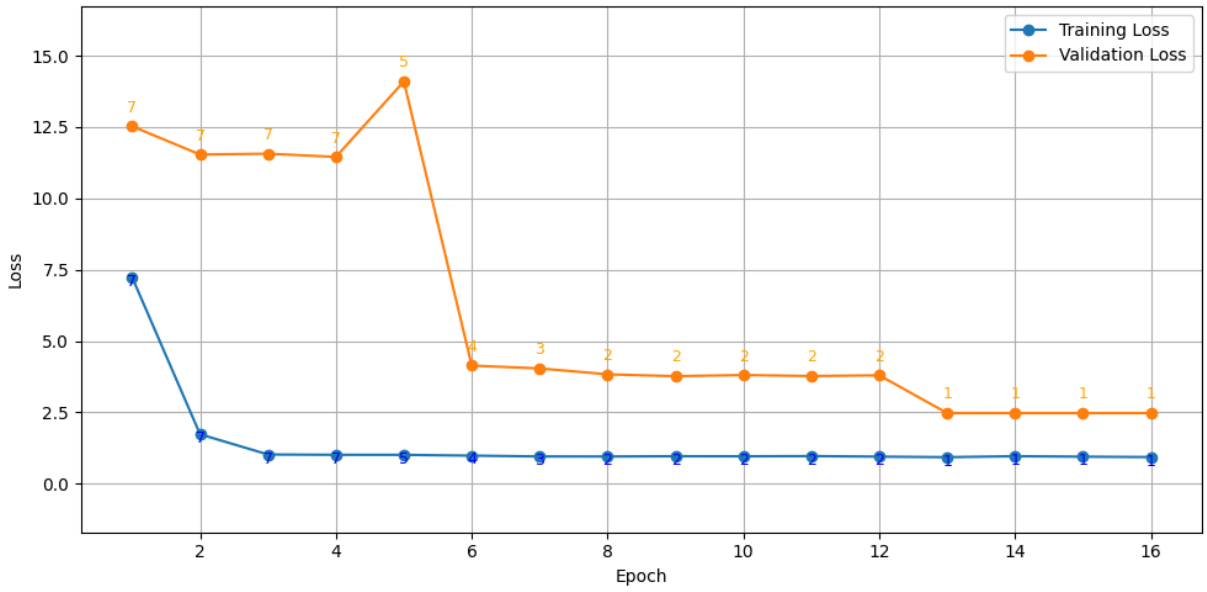

Figure 4.4: Average training and validation loss per epoch for RoBERTa

Moving further to the training of RoBERTa we find in Figure 4.4 that the training and validation loss starts to converge as passing through the fifth epoch and improves the further it reach. In the RoBERTa training process, a minimum of four epochs was run, and maximum of 16 with a mean of 7.714.

The final training process of the complete set of Informer models included the VADER sentiment label and can be seen in Figure 4.5. Compared to the FinBERT model we see a clearer convergence between validation loss and training loss, however the convergence is not a clear compared to the Informer-RoBERTa model. The model completed a minimum of four, maximum of 12 and a mean of 6 epochs were completed.

The total time complexity of the Informer model was reported to be 9 minutes and 24 seconds for the entirety of the training to be completed. With a generally low number of epochs completed, the company specific models that triggered early stopping at a sooner stage was AAPL, MSFT and TSLA. A full description of number of epochs trained can be found in Figure C.1 in the appendix.

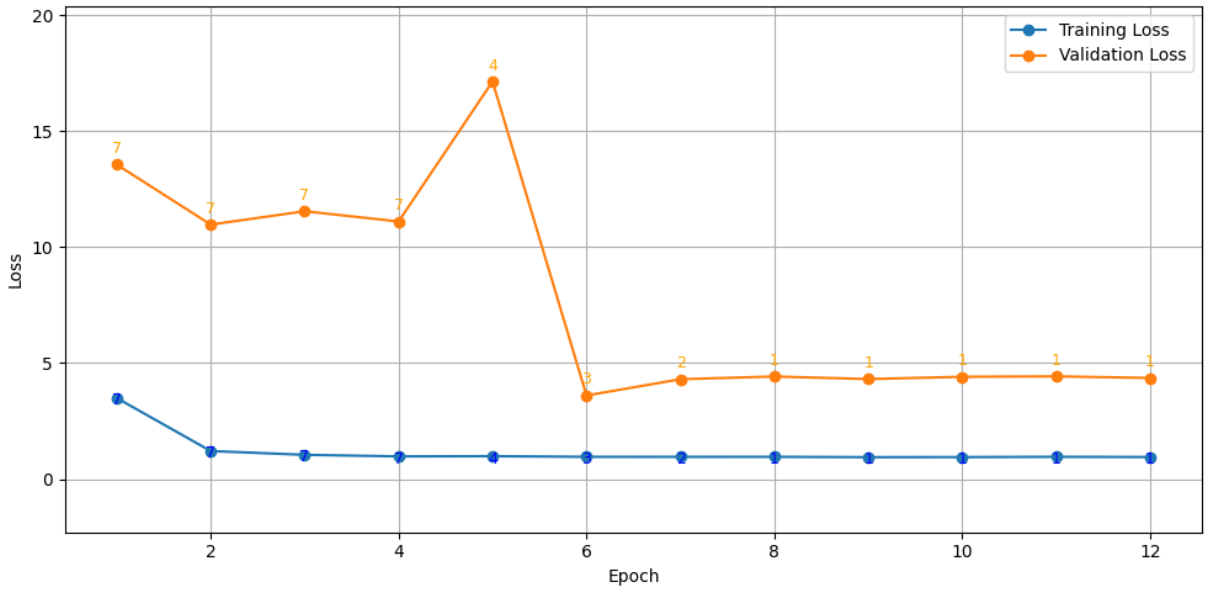

Figure 4.5: Average training and validation loss per epoch for VADER.

PatchTST

Proceeding to the training of PatchTST was done by utilizing the "run.py" file from the GitHub repository and by choosing the hyper parameters described in previous section. As in the case of the Informer architecture, a total number of 21 models were trained and the scores below are averaged by the number of sentiment labels which are three. Tuned hyper parameters differ from those described in preceding section, as PatchTST includes an approach to decrease dimensions of the data and those are patch length and stride length. The full descriptions of the model configuration and hyper parameter settings can be found in Tables C.2 and C.4.

In Figure 4.6 we find that the training loss and validation loss consistently decreases from epoch to epoch, however a rapid decrease occurs in the sixth epoch while one company model shut down training at this point due to the early stopping function. We see that the validation loss and training loss seem to converge at the later epochs, although being stopped by early could indicate a local minima or saddle point at which ADAM failed. The PatchTST-FinBERT combination ran for a minimum of six, maximum of 15 and with a mean of 9.286 epochs.

Similarly to the training of PatchTST-FinBERT, we find in Figure 4.7 that the training of PatchTST-RoBERTa early on drops companies at the fifth and seventh epoch, to the result of one company likely to be noisy and that PatchTST and ADAM was not able to grasp the underlying patterns. The PatchTST-

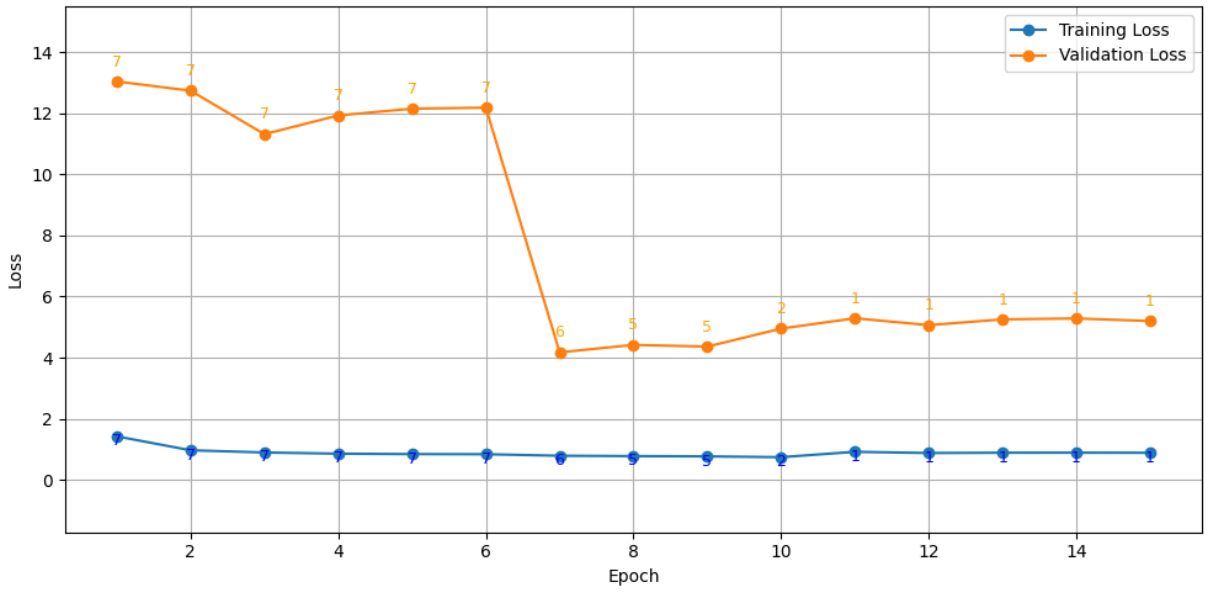

Figure 4.6: Average training and validation loss per epoch for FinBERT

RoBERTa models ran for a minimum of four, maximum of 15 and mean of 8.857 epochs. Similarly, the training and validation loss seem to converge.

The final model, PatchTST-VADER, can be seen in Figure 4.8 and reveals a similar training process as for the FinBERT and RoBERTa sentiment labels. During the first four epochs there is a decline, followed by the sixth epoch were a greater reduction in validation loss occurs and only four companies ran for seven epochs. These PatchTST-VADER models ran for a minimum of five, maximum of 16 and with a mean of 7.857 epochs.

The total time complexity of the PatchTST models was reported to be 8 minutes and 29 seconds for the entirety of the training to be completed. There is a clear pattern across the PatchTST models as all depict a steep decrease in validation loss at the seventh epoch, and the cause of this is possibly due to one or two company datasets which can occur as too noisy for the model and ADAM to perform. The three companies who dropped at the earlier epochs are NVIDIA, Tesla and Google, to view the full description and number of epochs trained in each model can be found in the figure C.2.

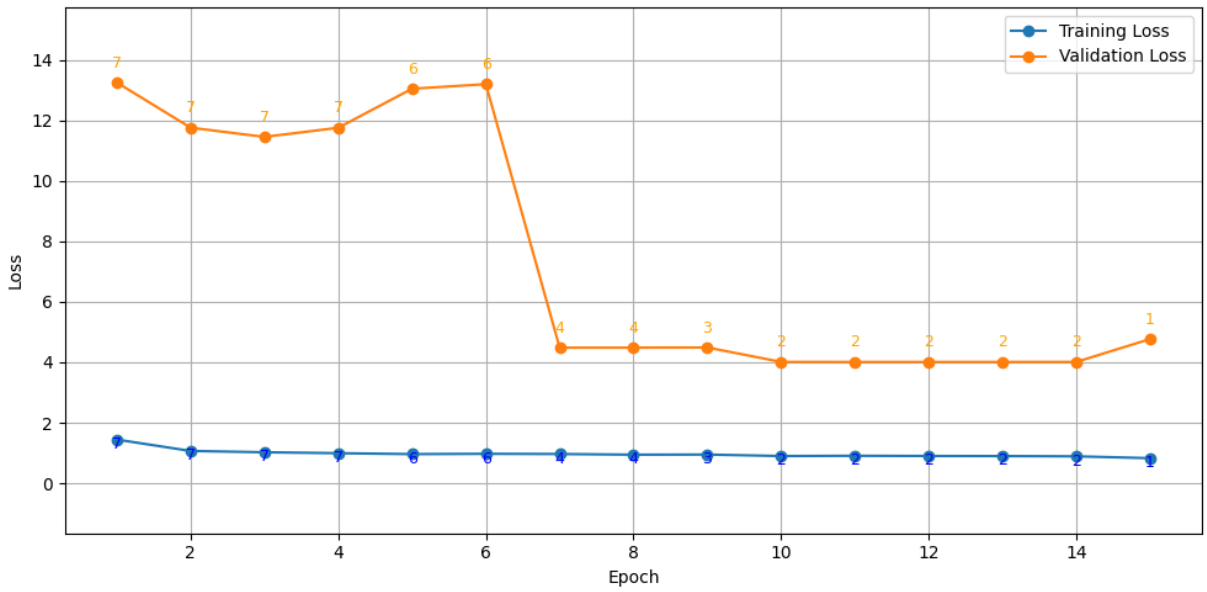

Figure 4.7: Average training and validation loss per epoch for RoBERTa

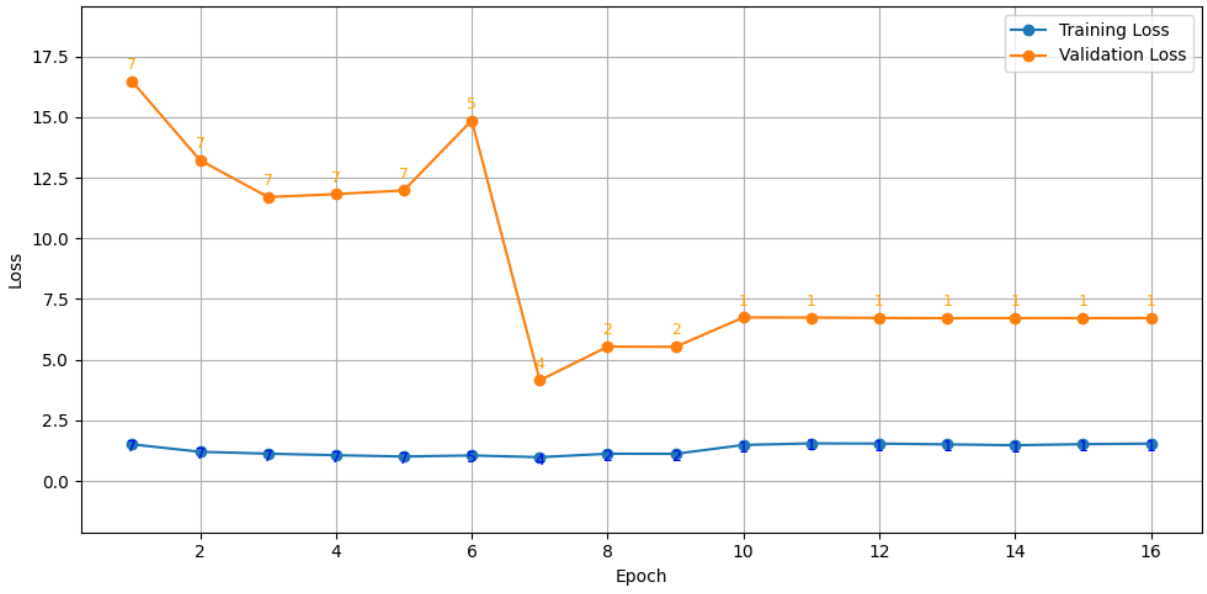

Figure 4.8: Average training and validation loss per epoch for VADER.

Chapter 5

Analysis

In this chapter, the results of our work is presented, analyzed and discussed. First in Section 5.1, the regression and classification metrics are shown, together with the adapted DM-tests. Secondly, in Section 5.2 the results are discussed in relation to previous literature. In Section 5.3 we draw conclusions based upon our results and discussion. Limitations and suggestions of future research is given in Section 5.4.

5.1 Findings

In Table 5.1 we find the average metrics across both prediction models and sentiment models. In the top three rows we find the different combinations of Informer and the three sentiment models, and we find that Informer-FinBERT receive the lowest scores across all metrics, that is $\overline{MAE}$, $\overline{MSE}$, $\overline{MedAE}$ and $\overline{MedSE}$. We can also see difference between FinBERT and the other sentiment models with scores nearly ≈ 0.3 points lower than the second best performer RoBERTa. The most poor performance was Informer-VADER with a $\overline{MAE}$, $\overline{MSE}$, $\overline{MedAE}$ and $\overline{MedSE}$ scores approximately two to four times higher than those of Informer-FinBERT. In the bottom three rows we find the results for all combinations of PatchTST and sentiment models, and at a first glance we find that all scores are significantly lower than those provided by the Informer models. The best performance from the PatchTST models were PatchTST-VADER with the lowest $\overline{MAE}$, $\overline{MedAE}$ and $\overline{MedSE}$ while PatchTST-RoBERTa achieved the lowest $\overline{MSE}$ score. However, the differences in scores do not vary as much of those found for the Informer models which could be seen as an early indication of PatchTST being more robust with weaker signals. Overall, the PatchTST models outperformed the Informer models in terms of regression.

Table 5.1: Average Regression Metrics

Prediction	**Sentiment**	$\overline{\textbf{MAE}}$	$\overline{\textbf{MSE}}$	$\overline{\textbf{MedAE}}$	$\overline{\textbf{MedSE}}$
Informer	FinBERT	**0.487**	**0.395**	**0.422**	**0.209**
	RoBERTa	0.759	0.750	0.748	0.618
	VADER	0.877	0.983	0.871	0.823
PatchTST	FinBERT	0.240	0.156	0.184	0.044
	RoBERTa	0.222	**0.139**†	0.164	0.033
	VADER	**0.221**†	0.143	**0.154**†	**0.028**†

Note: **Bold**: best per prediction model, †: best overall

In Table 5.2 we find the pseudo classification scores from the different model combinations. In the top three rows we find the classification scores for the different Informer models and we find that the Informer-FinBERT model performed the best out of the three models across all scores. Although, the scores are relatively similar between models with a maximum difference of ≈ 0.01 for the recall score signals a relatively robust performance of the Informer across various sentiment signals. Looking further at the bottom three rows, we see the PatchTST scores for the different sentiment models and we find that PatchTST-VADER received the highest $\overline{MDA}$ and $\overline{Precision}$, while PatchTST-FinBERT received the highest $\overline{Recall}$ and $\overline{F_1 - score}$. The best overall performing model was Informer-FinBERT. However, although PatchTST performed generally worse than Informer in terms of pseudo classification metrics the differences are not as evident as those previously described in Table 5.1. The best performing models in terms of classification of directions were the Informer models which is in contrast to previously mentioned PatchTST models for regression.

Table 5.2: Average Directional Classification Metrics

Prediction	**Sentiment**	$\overline{\textbf{MDA}}$	$\overline{\textbf{Precision}}$	$\overline{\textbf{Recall}}$	$\overline{F_1\textbf{-score}}$
Informer	FinBERT	**0.504**†	**0.470**†	**0.508**†	**0.488**†
	RoBERTa	0.497	0.463	0.498	0.480
	VADER	0.500	0.466	0.502	0.483
PatchTST	FinBERT	0.432	0.391	**0.401**	**0.396**
	RoBERTa	0.430	0.384	0.376	0.380
	VADER	**0.435**	**0.392**	0.391	0.392

Note: **Bold**: best per prediction model, †: best overall

The result of the DM tests are shown in Table 5.3. Overall we found that there was a significant difference between the PatchTST models and the Informer models, with the latter being consistently outperformed. There was also significant differences within the Informer models, where the transformer-based sentiment models were clearly superior to the rule based variant. The results of the tests between the PatchTST models were less clear however, with some significant differences but with statistics of lesser magnitude, and also a case where we could not reject the null of zero difference: VADER vs RoBERTa.

Table 5.3: Diebold-Mariano Tests

	Model 1	**Model 2**	**Test Statistic**
Between Models	Informer-FinBERT	PatchTST-FinBERT	**-13.241*****
	Informer-RoBERTa	PatchTST-RoBERTa	**-29.872*****
	Informer-VADER	PatchTST-VADER	**-32.265*****
Within Informer	Informer-FinBERT	Informer-VADER	**24.413*****
	Informer-FinBERT	Informer-RoBERTa	**22.253*****
	Informer-VADER	Informer-RoBERTa	**-10.018*****
Within PatchTST	PatchTST-FinBERT	PatchTST-VADER	**-4.069*****
	PatchTST-FinBERT	PatchTST-RoBERTa	**-7.423*****
	PatchTST-VADER	PatchTST-RoBERTa	-0.730
Note:	**Bold**: significant at $p < 0.05$ after both BH and Bonferroni correction; *p<0.1; **p<0.05; ***p<0.01 (uncorrected)		

Figures A.1 to A.7 show the actual predicted and true series that the above measures were calculated from and gives an intuitive picture of how the prediction models performed using the signals from the sentiment models. It is apparent to see that the PatchTST models consequently managed to find the pattern of the test data and remain closer to the true values as compared to the Informer, with an exception of the predictions of META where they performed quite similar across all sentiments models. The Informer models generally made more jagged and erratic predictions that while being further from the predicted series still captured directional movement. The Informer models also sometimes lost signal as like RoBERTa in Figure A.3 and VADER in Figure A.7, and reverted to predicting the mean of the series.

5.2 Discussion

Referring to the previous subsection consisting of our findings, we find that the results vary between the type of metrics that the analysis is based on. Several interesting findings have emerged from our analysis, both regarding the predictive performance of the two transformer models and the effect of the sentiment signals. These insights span both regression-based distance metrics and pseudo classification scores.

In terms of distance and regression metrics we find that the PatchTST demonstrate superior performance across all combinations, with consistently lower distances between the predicted and actual values. Beginning with predictive performance and the effect of sentiment signals for regression, we find that the PatchTST does provide more consistent and robust predictive ability regardless of the sentiment method. In contrast, the Informer's performance was severely impacted by the type of sentiment score provided, which can indicate that possibly weak or noisy signals are difficult for the attention mechanism to adapt and learn from. This difference becomes evident in the prediction curve which can be found in Figures A.1 to A.7. Another interesting aspect to consider, in the case of Informer, is that the signal provided by FinBERT was drastically better than those of RoBERTa and VADER where PatchTST preferred the opposite. Table 3.7 reveals that RoBERTa is the most pessimistic sentiment model (but still with a mean above neutral), followed by VADER, with FinBERT being the most optimistic. This contrast may suggest that Informer favors either strongly filtered or very permissive sentiment inputs, whereas PatchTST might benefit from stronger and less subtle sentiment signals.

On the other hand, when evaluating metrics in terms of direction (that is, the ability to correctly classify the price movement), the differences between models are less clear than those in regression. Here, we find that the different combinations of Informer do outperform the PatchTST with consistently higher pseudo classification scores (see Table 5.2), especially when combined with FinBERT. The Informer appears robust to different sentiment methods in this classification task. PatchTST, however, is somewhat more sensitive to the sentiment input, though the differences between combinations are relatively smaller. The cause of this difference is nebulous however, one possible reason is the models predictive nature: Whereas PatchTST is capable of capturing longer trends, this might explain its smoother and more stable predictions across time. The Informer instead reacts more erratically to new information, producing sharper local deviations that align with sudden directional changes in the features.

It is also important to address the differences in scale and temporal coverage across the individual companies, as both the number of news articles and corresponding date ranges vary significantly. As shown in Table 3.8, there are substantial differences between companies, for instance, Apple was mentioned in a total of 97,348 articles, whereas NVIDIA appeared in only 17,560 articles. Interestingly, we do not

observe any clear differences in model performance, which raises an important point regarding the data scale. Although common belief in machine learning is that "more data is better", our finding suggests that this relationship may not always be straightforward.

Both of the prediction models studied in this work are adaptations of the basic transformer with the intention of modeling timeseries data more efficiently [1]. The manner of adaptation vary between them; while PatchTST reduce computational load via patches and channel independence, it still retains the self-attention of the original Transformer. The Informer in contrast reduce computational intensity by adapting the attention mechanism in a probabilistic way. The performance loss could thus be due to the ProbSparse attention mechanism in the Informer struggling to adapt to noisier inputs or weaker signals.

The results from this study align with previous research in several areas, particularly especially within the field of NLP, where the inclusion of news sentiment in financial forecasting settings has been shown to affect prediction accuracy as in Bollen et al. (2011). Moreover, the development and improvements made in the transformer domain have enabled more efficient use of these powerful architectures, reducing time complexities for sequential data and predictive forecasting. This, in turn, allows for more exploratory approaches 2.2. Broadly however, the performance of our transformers were not dramatically different than Schumaker & Chen (2009) for instance. Additionally, our findings show that the implementation and integration of FinBERT yields consistent results in terms of both regression and pseudo-classification metrics, in line with the findings of Gu et al. (2024), Loughran & McDonald (2011) and Araci (2019), although not always being far superior.

5.3 Conclusion

Depending on the task at hand and under the specific conditions, the decision of both sentiment method and model should be carefully taken into consideration. Referring to the three research questions (**RQs**) stated in Section 1, we draw the following conclusions:

[1] It should also be noted that in terms of time complexity, the training duration is not to be considered an issue in our setting as it merely took 20 minutes to train all 21 models, although this heavily relied on the computing power provided by the NVIDIA A100 GPU, as proposed by Goodfellow et al. (2016).

RQ1: How do different sentiment generating models compare on providing sentiment signals for financial forecasting?

There is a clear and significant difference between sentiment models in terms of their restrictiveness and their impact on predictive performance, as measured by both regression and classification metrics. This effect is particularly pronounced in the Informer, which was more sensitive to the choice of sentiment input with FinBERT consistently improving performance, a finding also supported by the DM test results. In contrast, the PatchTST models demonstrated greater robustness to variation in sentiment input. According to the DM tests, FinBERT performed significantly worse than the other sentiment models in this architecture, although the magnitude of the difference was smaller. In terms of raw performance metrics, VADER produced the best results in most cases.

RQ2: How do Informer and PatchTST compare in terms of forecasting performance in financial time series in relation to both continuous and discrete outputs?

The effect of model architecture depends on the type of evaluation metric. PatchTST consistently outperforms Informer in all regression metrics, indicating stronger predictive accuracy. This is supported by the DM test results, that found significant differences in forecasting performance. However, Informer shows better at pseudo classification tasks, especially in $\overline{MDA}$, although the performance is less substantial than those achieved by PatchTST in the regression setting. This can indicate a task-specific trade-off between the two transformer-based architectures.

RQ3: To what extent do transformer-based models extract meaningful patterns from financial time series when sentiment signals are weak or noisy?

Our results indicate that transformer-based models can extract meaningful patterns from financial time series even when sentiment signals are weak or noisy. The PatchTST model consistently performs well across all sentiment inputs, suggesting that it is relatively insensitive to sentiment variation and exhibits strong robustness in such conditions. In contrast, the Informer model shows a higher dependency on the quality of the sentiment signal. It achieves its best performance when using FinBERT, but performs significantly worse with RoBERTa and VADER. In some cases, Informer appears to lose predictive signal and reverts to behavior resembling a mean model, depending on the sentiment input. This suggests that Informer is more sensitive to noise and reliant on strong external features to maintain performance.

5.4 Limitations, Challenges and Potential Future Research

In this paper there are several limitations and possible biases that can have an impact on the results from the study. First, the decision to only include 'Magnificent Seven' were arbitrarily selected but with the knowledge of that these corporations are today some of the largest companies on the financial market but also that they have long been listed on the NASDAQ stock exchange. Their performance as constantly growing companies can of course bias the results both in terms of the sentiment in news but also in price. However, as this thesis focus on revealing the effect of sentiment from news articles on the price movements on equity prices we deem it wise to use some of the more common companies as number of news that these companies has been made subject of is vast. On the other hand, some of these companies have not always been the most commonly mentioned corporations such as NVIDIA which early on was only a company directed to the manufacturing of GPUs to gaming computers, but has in later years become one of the core companies for the AI revolution.

Future research could for instance explore the methodological choices made in this thesis. Model selection was limited to Informer and PatchTST, though many other transformer-based architectures for time-series forecasting exist. While classical models like LSTMs and GRUs are already well-studied, incorporating a baseline model could provide a useful point of comparison. The approach to sentiment averaging and decay also warrants further theoretical development, as the current implementation may risk information loss. Additionally, it could be interesting to examine how sentiment and price relate to each other over different predictions lengths or lags. Finally, further inference could examine how the volume and temporal coverage of related articles affect predictive performance. Transformers are also famously very data-hungry, and increasing the data scale by incorporating more stocks or more variables is a natural continuation of our work.

Recent developments in deep learning and AI present significant ethical and environmental challenges that must be addressed, and this thesis also touches on several of these concerns. Key considerations include fairness, the potential impact on individuals, and the environmental cost associated with the substantial energy requirements of training such models. In terms of fairness and individual impact, the application of deep learning in financial prediction can lead to distorted market behavior or the creation of unfair advantages, often influenced by disparities in access to computational resources. While financial prediction is inherently complex due to the unpredictable nature of markets, promoting transparency can help mitigate some of these risks. Moreover, the use of advanced models frequently demands high computational power, leading to increased energy consumption and negative environmental consequences. Therefore, the development of more efficient architectures such as PatchTST and Informer, which aim to

reduce time complexity and, consequently, computational and energy demands, is of critical interest and should be further explored.

Bibliography

Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. (2019). "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Araci, D. (2019). Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*.

Bachmann, M. (2025). *RapidFuzz (Version 3.13.0)*. `https://pypi.org/project/rapidfuzz/`. Python package for rapid fuzzy string matching.

Bahdanau, D., Cho, K. & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Benjamini, Y. & Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)* vol. 57, no. 1 pp.289–300.

Blaskowitz, O. & Herwartz, H. (2011). On economic evaluation of directional forecasts. *International journal of forecasting* vol. 27, no. 4 pp.1058–1065.

Bollen, J., Mao, H. & Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of computational science* vol. 2, no. 1 pp.1–8.

Bonferroni, C. (1936). Teoria statistica delle classi e calcolo delle probabilita. *Pubblicazioni del R istituto superiore di scienze economiche e commericiali di firenze* vol. 8 pp.3–62.

Borup, D., Christensen, B. J. & Ergemen, Y. E. (2019). Assessing predictive accuracy in panel data models with long-range dependence.

Botchkarev, A. (2018). Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology. *arXiv preprint arXiv:1809.03006*.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems* vol. 33 pp.1877–1901.

Cumby, R. E. & Modest, D. M. (1987). Testing for market timing ability: A framework for forecast evaluation. *Journal of Financial Economics* vol. 19, no. 1 pp.169–189. ISSN: 0304-405X. https://doi.org/https://doi.org/10.1016/0304-405X(87)90033-X. https://www.sciencedirect.com/science/article/pii/0304405X8790033X.

Diebold, F. X. & Mariano, R. S. (1995). Com paring predictive accu racy. *Journal of Business and Economic Statistics* vol. 13, no. 3 pp.253–263.

Dong, Z., Fan, X. & Peng, Z. (2024). *FNSPID: A Comprehensive Financial News Dataset in Time Series*. arXiv: 2402.06698 [q-fin.ST]. https://arxiv.org/abs/2402.06698.

Fama, E. F. (1970). Efficient capital markets. *Journal of finance* vol. 25, no. 2 pp.383–417.

Fredriksson, T., Mattos, D. I., Bosch, J. & Olsson, H. H. (2020). "Data labeling: An empirical investigation into industrial challenges and mitigation strategies". In: *International conference on product-focused software process improvement*. Springerpp. 202–216.

Ghojogh, B. & Ghodsi, A. (Dec. 2020). Attention Mechanism, Transformers, BERT, and GPT: Tutorial and Survey. https://doi.org/10.31219/osf.io/m6gcn.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Greer, M. (2003). Directional accuracy tests of long-term interest rate forecasts. *International Journal of Forecasting* vol. 19, no. 2 pp.291–298. ISSN: 0169-2070. https://doi.org/https://doi.org/10.1016/S0169-2070(01)00141-8. https://www.sciencedirect.com/science/article/pii/S0169207001001418.

Gu, W., Zhong, Y., Li, S., Wei, C., Dong, L., Wang, Z. & Yan, C. (2024). "Predicting stock prices with finbert-lstm: Integrating news sentiment analysis". In: *Proceedings of the 2024 8th International Conference on Cloud and Big Data Computing*pp. 67–72.

Honnibal, M., Montani, I., Landeghem, S. V. & Boyd, A. (2020). spaCy: Industrial-strength Natural Language Processing in Python. https://doi.org/10.5281/zenodo.1212303. https://doi.org/10.5281/zenodo.1212303.

Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P. & Ba, J. (2017). *Adam: A Method for Stochastic Optimization.* arXiv: 1412.6980 [cs.LG]. https://arxiv.org/abs/1412.6980.

Lee, A. H., Fung, W. K. & Fu, B. (May 2003). Analyzing Hospital Length of Stay: Mean or Median Regression? *Medical Care* vol. 41, no. 5 pp.681–686. https://doi.org/10.1097/01.MLR.0000062550.23101.6F.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692.*

Loughran, T. & McDonald, B. (2011). When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks. *The Journal of finance* vol. 66, no. 1 pp.35–65.

Marcucci, J. (2024). "Macroeconomic forecasting with text-based data". In: *Handbook of Research Methods and Applications in Macroeconomic Forecasting.* Ed. by M. P. Clements & A. B. GalvÃ£o. Chapters. Edward Elgar Publishing. Chap. 16pp. 425–468. https://ideas.repec.org/h/elg/eechap/22222_16.html.

Newey, W. K. & West, K. D. (1987). A Simple, Positive Semi-Definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix. *Econometrica* vol. 55, no. 3 pp.703–708. ISSN: 00129682, 14680262. http://www.jstor.org/stable/1913610(**visited** 20 May 2025).

Nie, Y., Nguyen, N. H., Sinthong, P. & Kalagnanam, J. (2022). A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730.*

Plevris, V., Solorzano, G., Bakas, N. P. & Ben Seghier, M. E. A. (2022). Investigation of performance metrics in regression analysis and machine learning-based prediction models.

Refinitiv (2025). *Eikon Database Terminal.* https://www.refinitiv.com/en/products/eikon-trading-software. Accessed via Refinitiv Eikon terminal, April 2025.

Schumaker, R. P. & Chen, H. (2009). Textual analysis of stock market prediction using breaking financial news: The AZFin text system. *ACM Transactions on Information Systems (TOIS)* vol. 27, no. 2 pp.1–19.

Shelar, H., Kaur, G., Heda, N. & Agrawal, P. (2020). Named entity recognition approaches and their comparison for custom ner model. *Science & Technology Libraries* vol. 39, no. 3 pp.324–337.

Simkin, M. V. & Roychowdhury, V. P. (Dec. 2014). Why does attention to web articles fall with Time? *Journal of the Association for Information Science and Technology* vol. 66, no. 9 pp.1847–1856. ISSN: 2330-1643. https://doi.org/10.1002/asi.23289. http://dx.doi.org/10.1002/asi.23289.

Soydaner, D. (2022). Attention mechanism in neural networks: where it comes and where it goes. *Neural Computing and Applications* vol. 34, no. 16 pp.13371–13385.

Stryker, C. & Holdsworth, J. (2025). *What is NLP?* Accessed: 2025-03-25. IBM. https://www.ibm.com/think/topics/natural-language-processing.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. (2017). Attention Is All You Need. *CoRR* vol. abs/1706.03762. arXiv: 1706.03762. http://arxiv.org/abs/1706.03762.

Wang, I. (2025). *The 'Magnificent Seven' are back in the stock market's driver's seat — but are they still a buy?* Published May 10, 2025. Accessed May 10, 2025. MarketWatch. https://www.marketwatch.com/.

Wang, Y., Wu, H., Dong, J., Liu, Y., Long, M. & Wang, J. (2024). Deep Time Series Models: A Comprehensive Survey and Benchmark.

Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J. & Long, M. (2023). "TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis". In: *International Conference on Learning Representations*.

Zerveas, G., Jayaraman, S., Patel, D., Bhamidipaty, A. & Eickhoff, C. (2020). *A Transformer-based Framework for Multivariate Time Series Representation Learning*. arXiv: 2010.02803 [cs.LG]. https://arxiv.org/abs/2010.02803.

Zhang, Y. & Teng, Z. (2021). "Introduction". In: *Natural Language Processing: A Machine Learning Perspective*. Cambridge University Presspp. 3–24.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H. & Zhang, W. (2021). *Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting*. arXiv: 2012.07436 [cs.LG]. https://arxiv.org/abs/2012.07436.

Appendix A

Results

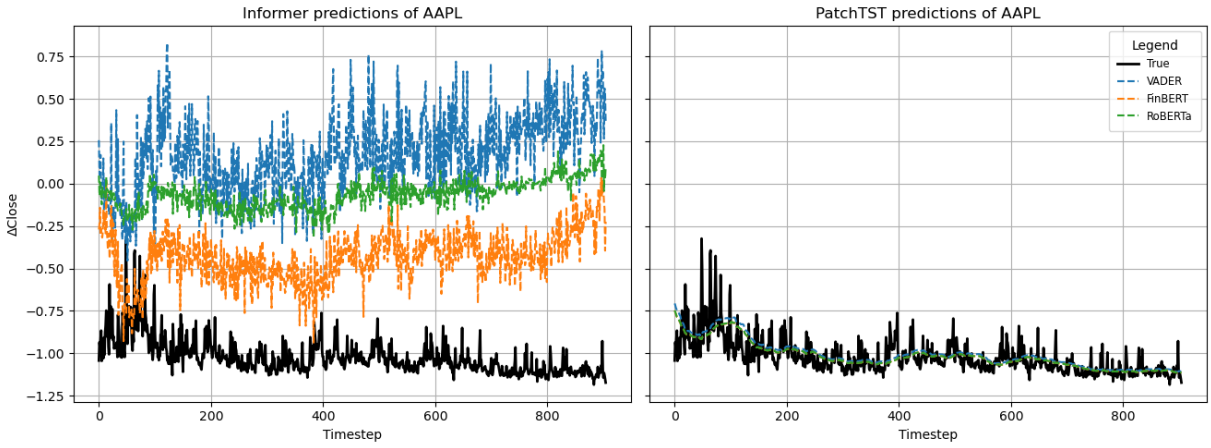

Figure A.1: Informer and PatchTST model predictions of AAPL closing price using different sentiment features.

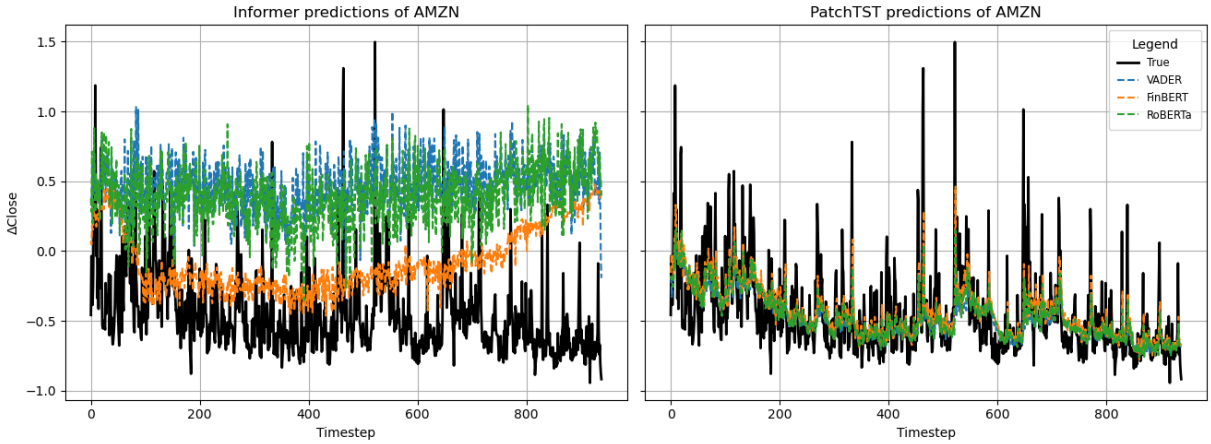

Figure A.2: Informer and PatchTST model predictions of AMZN closing price using different sentiment features.

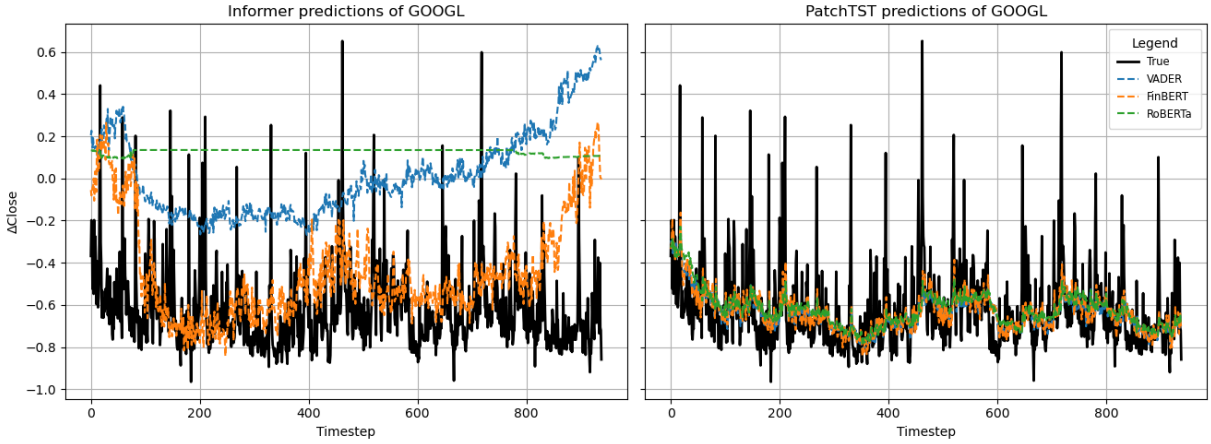

Figure A.3: Informer and PatchTST model predictions of GOOGL closing price using different sentiment features.

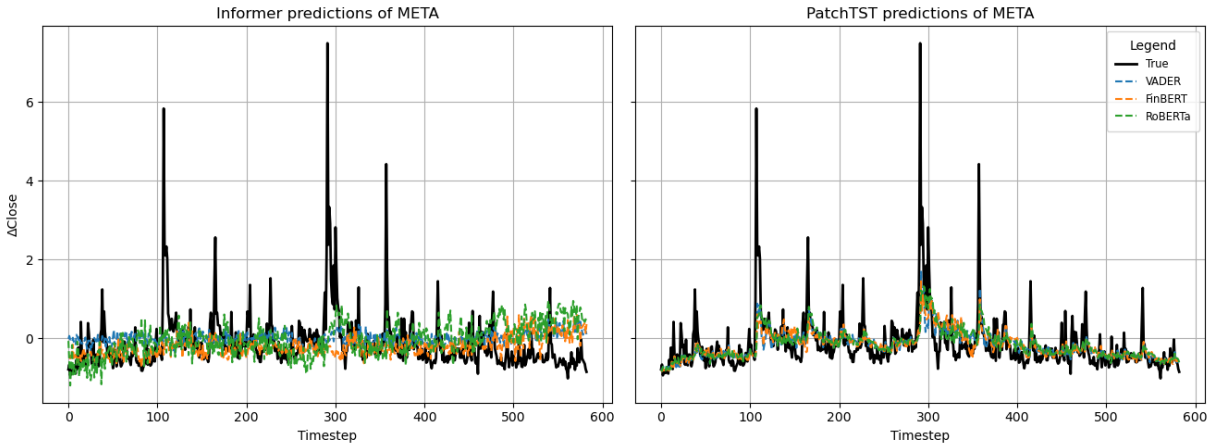

Figure A.4: Informer and PatchTST model predictions of META closing price using different sentiment features.

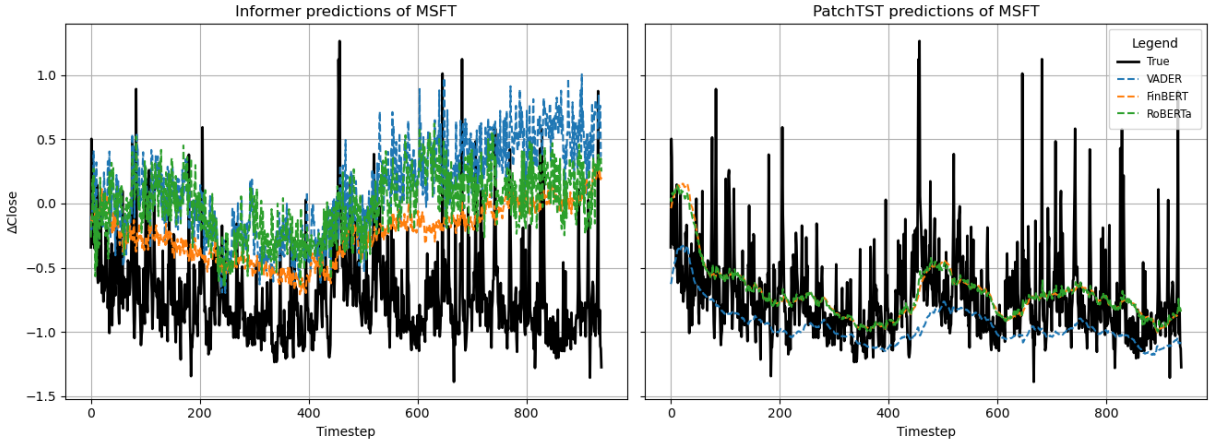

Figure A.5: Informer and PatchTST model predictions of MSFT closing price using different sentiment features.

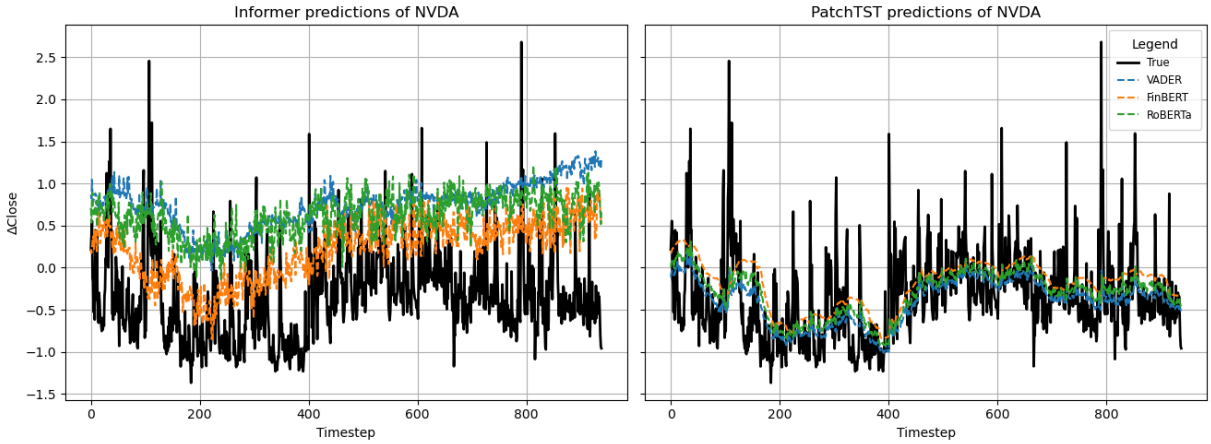

Figure A.6: Informer and PatchTST model predictions of NVDA closing price using different sentiment features.

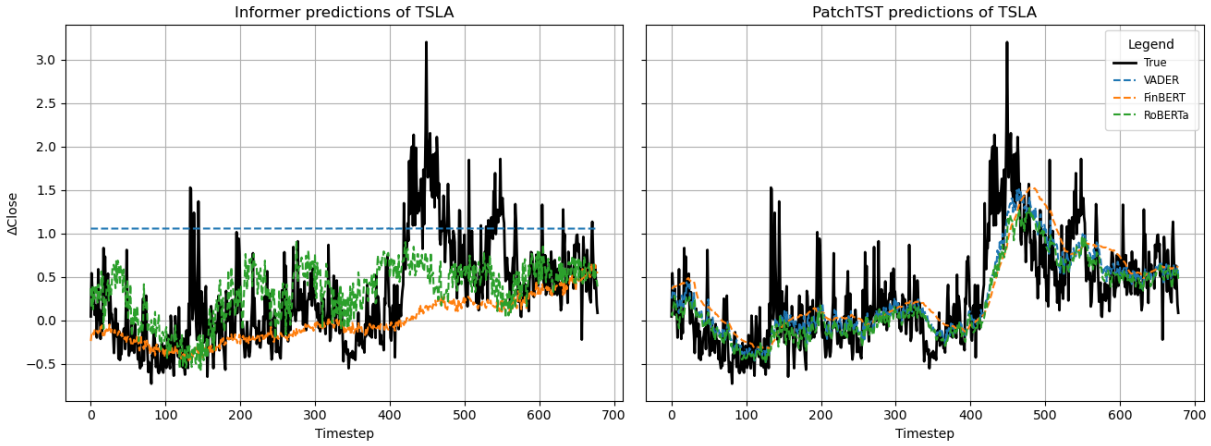

Figure A.7: Informer and PatchTST model predictions of TSLA closing price using different sentiment features.

Table A.1: Full Regression Metrics by Informer Predictions

Stock	**Sentiment**	**MAE**	**MSE**	**MedAE**	**MedSE**
AAPL	FinBERT	**0.592**	**0.386**	**0.604**	**0.365**
	RoBERTa	0.958	0.943	0.966	0.933
	VADER	1.195	1.505	1.193	1.424
AMZN	FinBERT	**0.450**	**0.294**	**0.405**	**0.164**
	RoBERTa	0.853	0.837	0.877	0.769
	VADER	0.935	0.970	0.970	0.941
GOOGL	FinBERT	**0.259**†	**0.118**†	**0.187**†	**0.035**†
	RoBERTa	0.754	0.600	0.793	0.629
	VADER	0.647	0.485	0.627	0.393
META	FinBERT	**0.456**	0.597	**0.297**	**0.088**
	RoBERTa	0.577	0.715	0.432	0.186
	VADER	0.515	**0.577**	0.430	0.185
MSFT	FinBERT	**0.558**	**0.404**	**0.526**	**0.277**
	RoBERTa	0.747	0.684	0.754	0.568
	VADER	0.864	0.950	0.828	0.686
NVDA	FinBERT	**0.690**	**0.617**	**0.690**	**0.476**
	RoBERTa	0.996	1.164	1.054	1.110
	VADER	1.125	1.460	1.168	1.364
TSLA	FinBERT	**0.402**	0.349	**0.244**	**0.060**
	RoBERTa	0.429	**0.308**	0.361	0.130
	VADER	0.861	0.933	0.878	0.771

Note: **Bold**: best per stock, †: best overall

Table A.2: Full Regression Metrics by PatchTST Predictions

Stock	**Sentiment**	**MAE**	**MSE**	**MedAE**	**MedSE**
AAPL	FinBERT	**0.056**†	**0.007**†	**0.039**†	**0.002**†
	RoBERTa	**0.056**†	**0.007**†	**0.039**†	**0.002**†
	VADER	0.060	**0.007**†	0.047	**0.002**†
AMZN	FinBERT	0.172	**0.061**	0.124	0.015
	RoBERTa	**0.167**	0.062	**0.116**	**0.013**
	VADER	0.171	0.067	0.118	0.014
GOOGL	FinBERT	**0.119**	**0.032**	0.084	**0.007**
	RoBERTa	0.124	0.034	0.093	0.009
	VADER	**0.119**	0.033	**0.083**	**0.007**
META	FinBERT	0.327	0.387	0.203	0.041
	RoBERTa	0.327	0.375	0.208	0.043
	VADER	**0.323**	**0.374**	**0.197**	**0.039**
MSFT	FinBERT	0.254	0.125	0.197	0.039
	RoBERTa	**0.245**	**0.116**	**0.190**	**0.036**
	VADER	0.273	0.164	0.174	0.030
NVDA	FinBERT	0.404	0.263	0.352	0.124
	RoBERTa	0.347	0.215	0.285	0.081
	VADER	**0.320**	**0.205**	**0.242**	**0.059**
TSLA	FinBERT	0.351	0.215	0.286	0.082
	RoBERTa	0.289	0.163	**0.214**	**0.046**
	VADER	**0.283**	**0.149**	0.220	0.048

Note: **Bold**: best per stock, †: best overall

Table A.3: Directional Classification Metrics for Informer

Stock	Sentiment	MDA	TP	FP	TN	FN	Precision	Recall	F_1
AAPL	FinBERT	**0.514**	222	238	243	201	**0.483**	**0.525**	**0.503**
	RoBERTa	0.487	203	244	237	220	0.454	0.480	0.467
	VADER	0.482	200	245	236	223	0.449	0.473	0.461
AMZN	FinBERT	0.507	224	248	252	214	0.475	0.511	0.492
	RoBERTa	**0.525**	233	241	259	205	**0.492**	**0.532**	**0.511**
	VADER	0.501	216	246	254	222	0.468	0.493	0.480
GOOGL	FinBERT	**0.488**	210	262	248	218	**0.445**	0.491	**0.467**
	RoBERTa	0.481	210	269	241	218	0.438	**0.491**	0.463
	VADER	0.488	206	258	252	222	0.444	0.481	0.462
META	FinBERT	0.474	131	164	145	142	0.444	0.480	0.461
	RoBERTa	0.509	140	153	156	133	0.478	0.513	0.495
	VADER	**0.521**	145	151	158	128	**0.490**	**0.531**	**0.510**
MSFT	FinBERT	**0.509**	221	240	256	221	**0.479**	0.500	**0.489**
	RoBERTa	0.503	219	243	253	223	0.474	**0.495**	0.485
	VADER	0.517	224	235	261	218	0.488	0.507	0.497
NVDA	FinBERT	**0.541**†	230	232	277	199	**0.498**†	**0.536**†	**0.516**†
	RoBERTa	0.494	206	252	257	223	0.450	0.480	0.464
	VADER	0.504	216	252	257	213	0.462	0.503	0.482
TSLA	FinBERT	**0.496**	163	186	173	156	0.467	0.511	0.488
	RoBERTa	0.484	159	190	169	160	0.456	0.498	0.476
	VADER	0.485	167	197	162	152	**0.459**	**0.524**	**0.489**

Note: **Bold**: best per stock, †: best overall

Table A.4: Directional Classification Metrics for PatchTST

Stock	**Sentiment**	**MDA**	**TP**	**FP**	**TN**	**FN**	**Precision**	**Recall**	$\mathbf{F_1}$
AAPL	FinBERT	0.390	130	258	223	293	0.335	**0.307**	**0.321**
	RoBERTa	0.390	130	258	223	293	0.335	**0.307**	**0.321**
	VADER	**0.393**	130	256	225	293	**0.337**	**0.307**	**0.321**
AMZN	FinBERT	**0.457**	189	260	240	249	**0.421**	**0.432**	**0.426**
	RoBERTa	0.439	171	259	241	267	0.398	0.390	0.394
	VADER	0.425	174	275	225	264	0.388	0.397	0.392
GOOGL	FinBERT	0.441	183	279	231	245	0.396	**0.428**	**0.411**
	RoBERTa	0.433	164	268	242	264	0.380	0.383	0.381
	VADER	**0.455**	177	260	250	251	**0.405**	0.414	0.409
META	FinBERT	**0.483**†	127	155	154	146	**0.450**†	**0.465**†	**0.458**†
	RoBERTa	0.464	115	154	155	158	0.428	0.421	0.424
	VADER	0.469	119	155	154	154	0.434	0.436	0.435
MSFT	FinBERT	0.396	157	282	214	285	0.358	0.355	0.356
	RoBERTa	0.415	155	262	234	287	0.372	0.351	0.361
	VADER	**0.434**	175	264	232	267	**0.399**	**0.396**	**0.397**
NVDA	FinBERT	0.414	170	291	218	259	**0.369**	**0.396**	**0.382**
	RoBERTa	**0.418**	164	281	228	265	**0.369**	0.382	0.375
	VADER	0.410	148	272	237	281	0.352	0.345	0.349
TSLA	FinBERT	0.441	136	196	163	183	0.410	0.426	0.418
	RoBERTa	0.448	127	182	177	192	0.411	0.398	0.404
	VADER	**0.460**	142	189	170	177	**0.429**	**0.445**	**0.437**

Note: **Bold**: best per stock, †: best overall

Appendix B

Data

B.1 Evaluation by Human Intelligence

In general we found that the original assignment contained both abstract connections as well as click-bait titles. An example of a click-bait title is "Warren Buffet Owns a Lot of Stocks – Here's the One I'm Most Excited About", which was referring to AAPL, but which, quite understandably, no other labeler found. Additionally, there were vague articles such as "10 Technology Stocks Moving In Thursday's Pre-Market Session" supposedly relating to NVDA, and "Today's Pickup: Car Hauling's Challenging Times; Amazon's Foray Into Freestanding Logistics" relating to TSLA, which consistently were missed by the other labelers.

We also found direct evidence of spaCys recall issues, for instance "Daily Dividend Report: AAPL,BKR,ABBV, ROK,WBA" was correctly assigned by all other labelers as AAPL. Another example was "Alphabet Inc (GOOGL) Stock Will Succeed Where Microsoft Was Stopped" which was missed by spaCy. (In a seed not used for the final version, the spaCy labeler also did one precision-error where it assigned "Little-known Indian tribe spotted in Peru´s Amazon" to the company Amazon).

The keyword-based labelers had their quirks as well. Their naive approach led them to capture irrelevant articles, but this varied quite a lot depending on the company and how distinct their keywords were. For instance, the fuzzy labeler was continuously flummoxed by the word "Metals" which it assigned to Meta. Sometimes abstract connections were found by lucky chance, for instance "Global Macro Outlook - China: Quality Over Quantity", where "Mac" was found and attributed to AAPL, which could be defended in this case.

Table B.1: Fuzzy keyword dictionary for MAG7 stocks

Stock	**Fuzzy Keywords**
AAPL	aapl, apple, apple inc, steve jobs, tim cook, ipad, iphone, mac, ios, macintosh, airpods, apple watch, apple tv, apple card, apple pay, icloud, app store, apple music, wozniak, steve wozniak, magnificent 7, magnificent seven, mag7, faang
MSFT	msft, microsoft, microsoft office, windows, azure, xbox, bing, linkedin, visual studio, microsoft teams, microsoft 365, microsoft dynamics, skype, onedrive, github, sharepoint, microsoft viva, viva engage, satya nadella, bill gates, paul allen, magnificent 7, magnificent seven, mag7
GOOGL	googl, goog, google, alphabet, youtube, gmail, android, chrome, google maps, google cloud, google drive, abc.xyz, larry page, sergey brin, sundar pichai, ruth porat, hennessy, ashkenazi, magnificent 7, magnificent seven, mag7, faang
AMZN	amzn, amazon, amazon.com, aws, alexa, kindle, amazon echo, amazon prime, ec2, prime video, twitch, audible, metro goldwyn mayer, mgm studios, fire tablet, jeff bezos, bezos, magnificent 7, magnificent seven, mag7, faang
NVDA	nvda, nvidia, geforce, geforce now, cuda, nvidia rtx, gtc, blackwell, nvidia drive, nvidia jetson, nvidia isaac, tegra, quantum computing, jensen huang, bill dally, magnificent 7, magnificent seven, mag7
META	meta, meta platforms, facebook, instagram, whatsapp, threads, messenger, zuckerberg, mark zuckerberg, meta quest, metaverse, the facebook inc, magnificent 7, magnificent seven, mag7, faang
TSLA	tsla, tesla, elon musk, musk, model 3, model s, model x, cybertruck, powerwall, megapack, solar city, tesla semi, supercharger, roadster, solarcity, electric vehicle, gigafactory, magnificent 7, magnificent seven, mag7

Note: Keywords based on occurrences on Yahoo Finance, Wikipedia and the companies' websites.

Table B.2: Hard keyword dictionary for MAG7 stocks

Stock	**Hard Keywords**
AAPL	AAPL, Apple, Apple Inc, Steve Jobs, Tim Cook, iPad, iPhone, Mac, iOS, Macintosh, AirPods, Apple Watch, Apple TV, Apple Card, Apple Pay, iCloud, App Store, Apple Music, Wozniak, Steve Wozniak, Magnificent 7, Magnificent Seven, MAG7, FAANG
MSFT	MSFT, Microsoft, Microsoft Office, Windows, Azure, Xbox, Bing, LinkedIn, Visual Studio, Microsoft Teams, Microsoft 365, Microsoft Dynamics, Skype, OneDrive, GitHub, SharePoint, Microsoft Viva, Viva Engage, Satya Nadella, Bill Gates, Paul Allen, Magnificent 7, Magnificent Seven, MAG7
GOOGL	GOOGL, GOOG, Google, Alphabet, YouTube, Gmail, Android, Chrome, Google Maps, Google Cloud, Google Drive, abc.xyz, Larry Page, Sergey Brin, Sundar Pichai, Ruth Porat, Hennessy, Ashkenazi, Magnificent 7, Magnificent Seven, MAG7, FAANG
AMZN	AMZN, Amazon, Amazon.com, AWS, Alexa, Kindle, Amazon Echo, Amazon Prime, EC2, Prime Video, Twitch, Audible, Metro Goldwyn Mayer, MGM Studios, Fire Tablet, Jeff Bezos, Bezos, Magnificent 7, Magnificent Seven, MAG7, FAANG
NVDA	NVDA, NVIDIA, Nvidia, GeForce, GeForce NOW, CUDA, NVIDIA RTX, GTC, Blackwell, NVIDIA DRIVE, NVIDIA Jetson, NVIDIA Isaac, Tegra, Quantum Computing, Jensen Huang, Bill Dally, Magnificent 7, Magnificent Seven, MAG7
META	META, Meta, Meta Platforms, Facebook, Instagram, WhatsApp, Threads, Messenger, Zuckerberg, Mark Zuckerberg, Meta Quest, Metaverse, The Facebook Inc, The Facebook, facebook, Magnificent 7, Magnificent Seven, MAG7, FAANG
TSLA	TSLA, Tesla, Elon Musk, Musk, Model 3, Model S, Model X, Cybertruck, Powerwall, Megapack, Solar City, Tesla Semi, Supercharger, Roadster, SolarCity, EV, Electric Vehicle, GigaFactory, Magnificent 7, Magnificent Seven, MAG7

Note: Keywords based on occurrences on Yahoo Finance, Wikipedia and the companies' websites.

Table B.3: spaCy NER keyword dictionary for MAG7 companies

Ticker	**spaCy NER Keywords (ORG, PERSON, PRODUCT)**
AAPL	AAPL, Apple, Apple Inc, Steve Jobs, Tim Cook, Steve Wozniak, iPhone, iPad, Mac, Apple Watch, Apple Music, Apple TV
MSFT	MSFT, Microsoft, Satya Nadella, Bill Gates, Paul Allen, Xbox, LinkedIn, Visual Studio, OneDrive, Skype, GitHub
GOOGL	GOOGL, GOOG, Google, Alphabet, Larry Page, Sergey Brin, Sundar Pichai, YouTube, Gmail, Android, Chrome, Google Maps, Google Drive
AMZN	AMZN, Amazon, Amazon.com, Jeff Bezos, Bezos, Alexa, Kindle, Twitch, Prime Video, Audible
NVDA	NVDA, Nvidia, Jensen Huang, Bill Dally, GeForce, CUDA, RTX, NVIDIA Drive, NVIDIA Jetson
META	META, Meta, Meta Platforms, Facebook, Mark Zuckerberg, Zuckerberg, Instagram, Messenger, WhatsApp, Threads, Meta Quest
TSLA	TSLA, Tesla, Elon Musk, Musk, Model 3, Model S, Model X, Cybertruck, Tesla Semi, Roadster

Note: Keywords based on occurrences on Yahoo Finance, Wikipedia and the companies' websites.

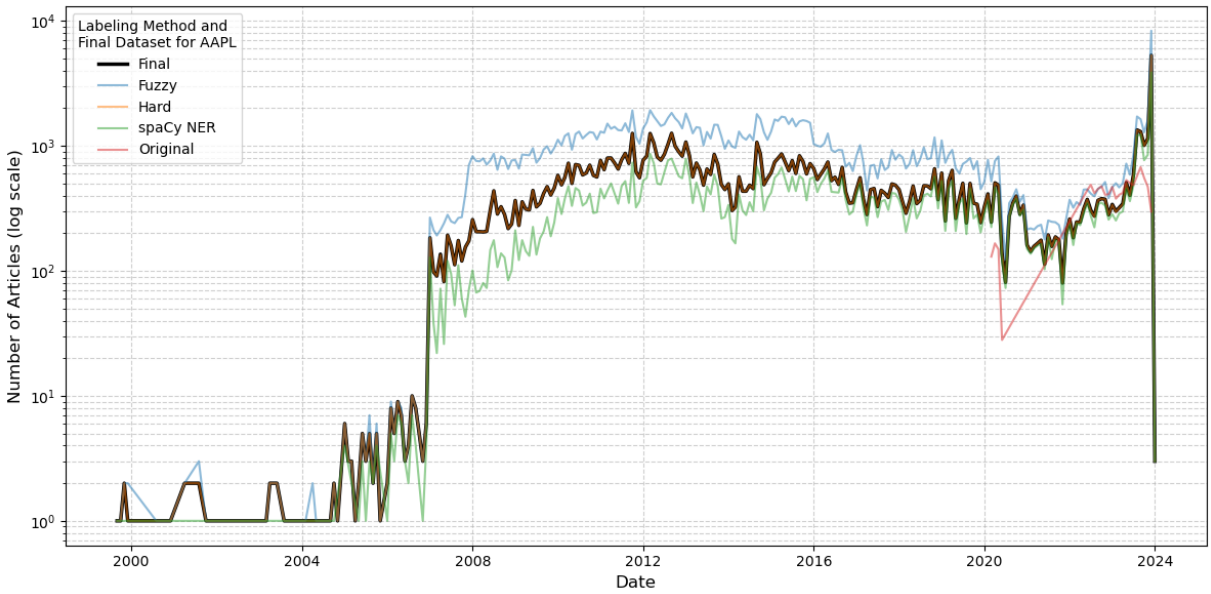

Figure B.1: Labeled AAPL Articles Over Time by Method (Log Scale)

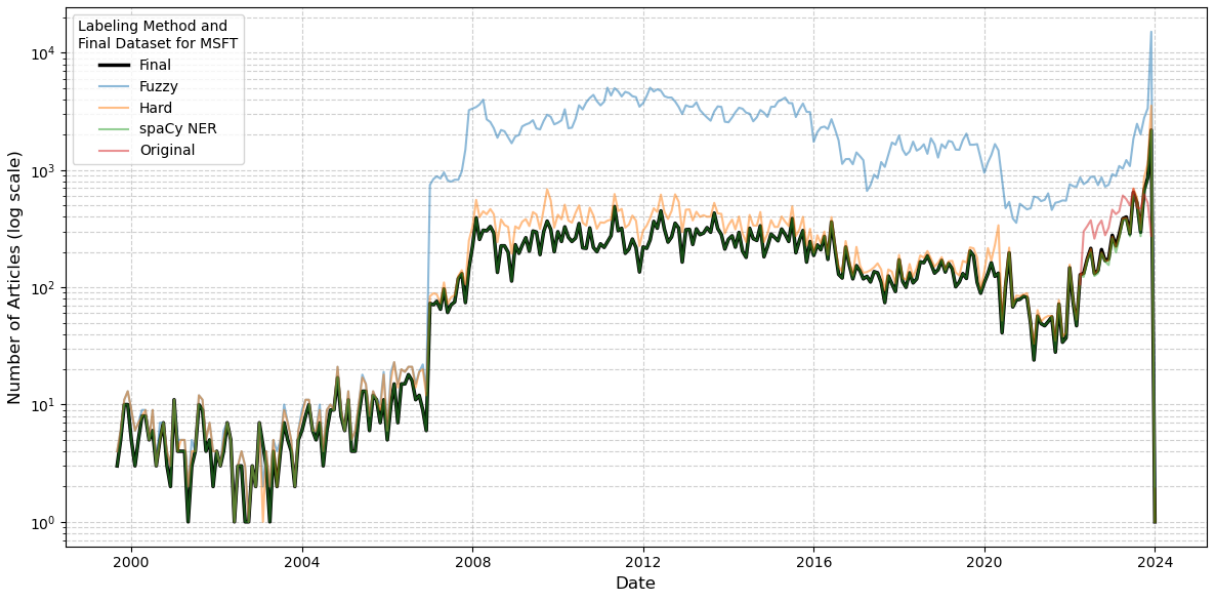

Figure B.2: Labeled MSFT Articles Over Time by Method (Log Scale)

Table B.4: Labeler Precision Metrics

Stock	**Labeler**	$\hat{\mathbf{p}}$	$\mathbf{Var}(\hat{\mathbf{p}})$	**CI Lower**	**CI Upper**
AAPL	Original	0.74	0.003847	0.618434	0.861566
	Fuzzy_85	0.58	0.004871	0.443212	0.717678
	Hard	0.90	0.001800	0.816856	0.983144
	spaCy	1.00	0.000000	1.000000	1.000000
MSFT	Original	0.90	0.001800	0.816850	0.983150
	Fuzzy_85	0.34	0.004488	0.208704	0.471296
	Hard	0.84	0.002688	0.738389	0.941611
	spaCy	1.00	0.000000	1.000000	1.000000
AMZN	Original	0.72	0.004029	0.595587	0.844413
	Fuzzy_85	0.70	0.004197	0.573021	0.826979
	Hard	0.86	0.002406	0.763853	0.956147
	spaCy	1.00	0.000000	1.000000	1.000000
GOOGL	Original	0.74	0.003846	0.618449	0.861551
	Fuzzy_85	0.72	0.004030	0.595577	0.844423
	Hard	0.96	0.000768	0.905697	1.014303
	spaCy	1.00	0.000000	1.000000	1.000000
NVDA	Original	1.00	0.000000	1.000000	1.000000
	Fuzzy_85	0.32	0.003450	0.190703	0.449269
	Hard	0.98	0.000392	0.941203	1.018797
	spaCy	1.00	0.000000	1.000000	1.000000
META	Original	0.74	0.003845	0.618461	0.861539
	Fuzzy_85	0.50	0.004997	0.361458	0.638542
	Hard	0.94	0.001127	0.874196	1.005804
	spaCy	1.00	0.000000	1.000000	1.000000
TSLA	Original	0.84	0.002667	0.738410	0.941590
	Fuzzy_85	0.52	0.004989	0.381556	0.658444
	Hard	0.96	0.000768	0.905698	1.014302
	spaCy	1.00	0.000000	1.000000	1.000000

Table B.5: Normalized Weights Assigned to Each Labeling Method by Stock

Stock	**Original**	**Fuzzy_85**	**Hard**	**spaCy**
AAPL	0.2298	0.1801	0.2795	0.3106
MSFT	0.2922	0.1104	0.2727	0.3247
AMZN	0.2209	0.2147	0.2638	0.3006
GOOGL	0.2164	0.2105	0.2807	0.2924
NVDA	0.3030	0.0970	0.2970	0.3030
META	0.2327	0.1572	0.2960	0.3145
TSLA	0.2530	0.1567	0.2892	0.3012

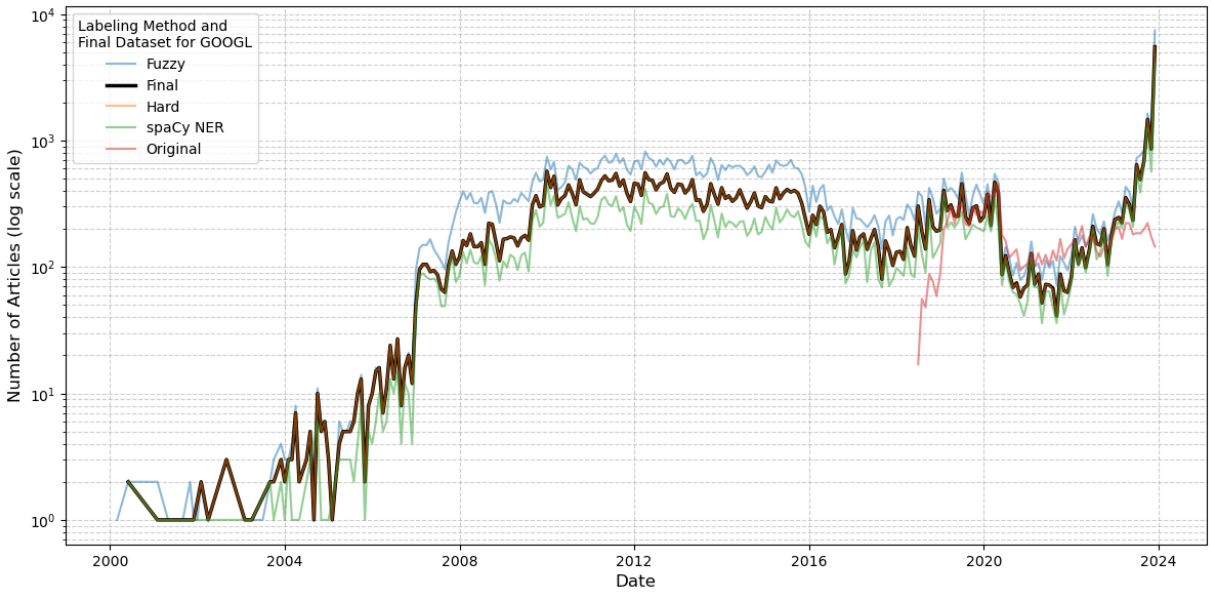

Figure B.3: Labeled GOOGL Articles Over Time by Method (Log Scale)

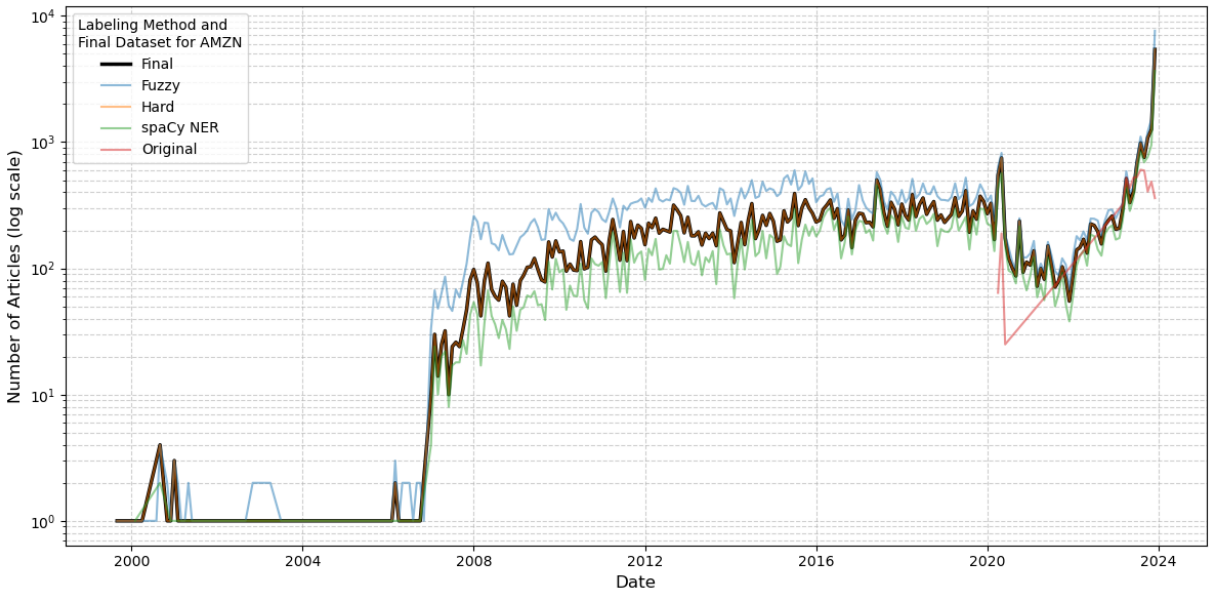

Figure B.4: Labeled AMZN Articles Over Time by Method (Log Scale)

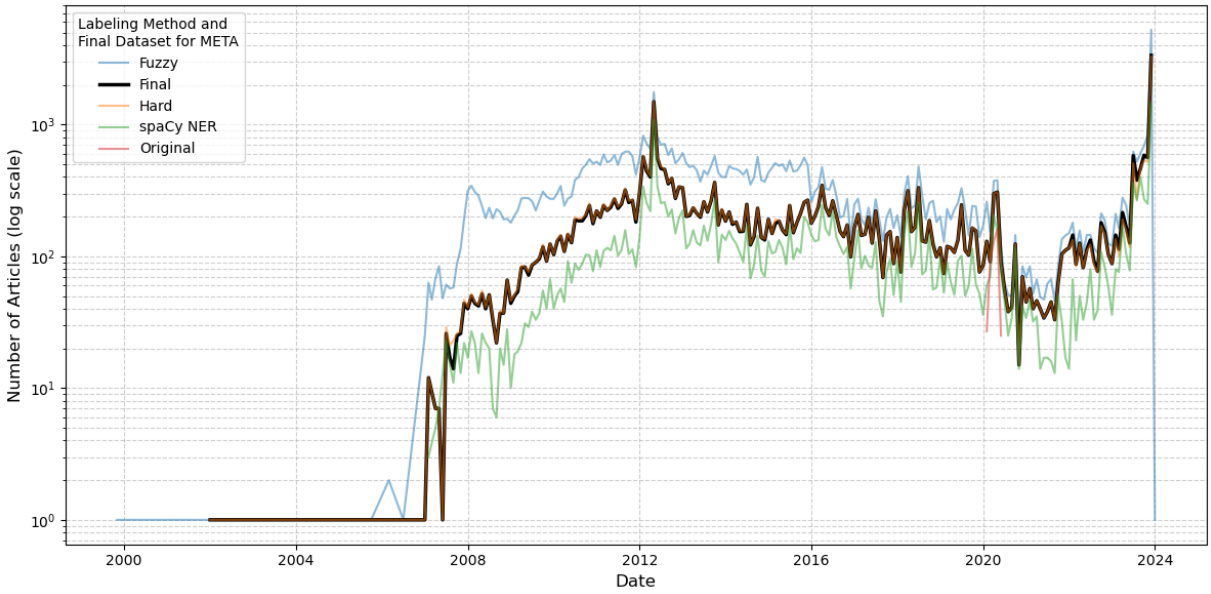

Figure B.5: Labeled META Articles Over Time by Method (Log Scale)

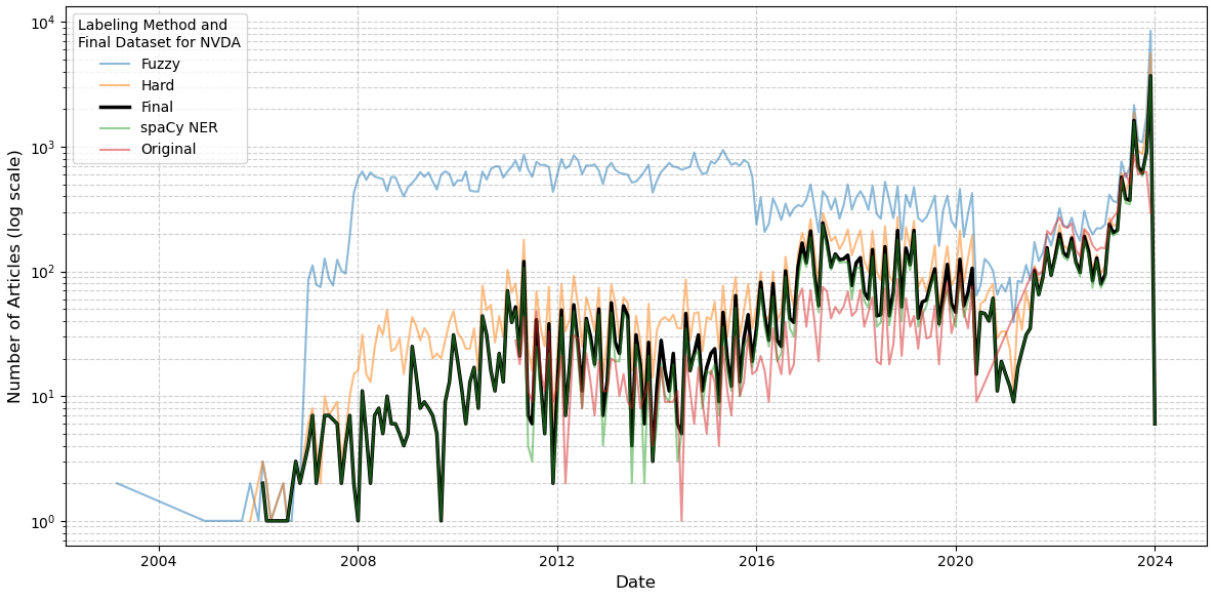

Figure B.6: Labeled NVDA Articles Over Time by Method (Log Scale)

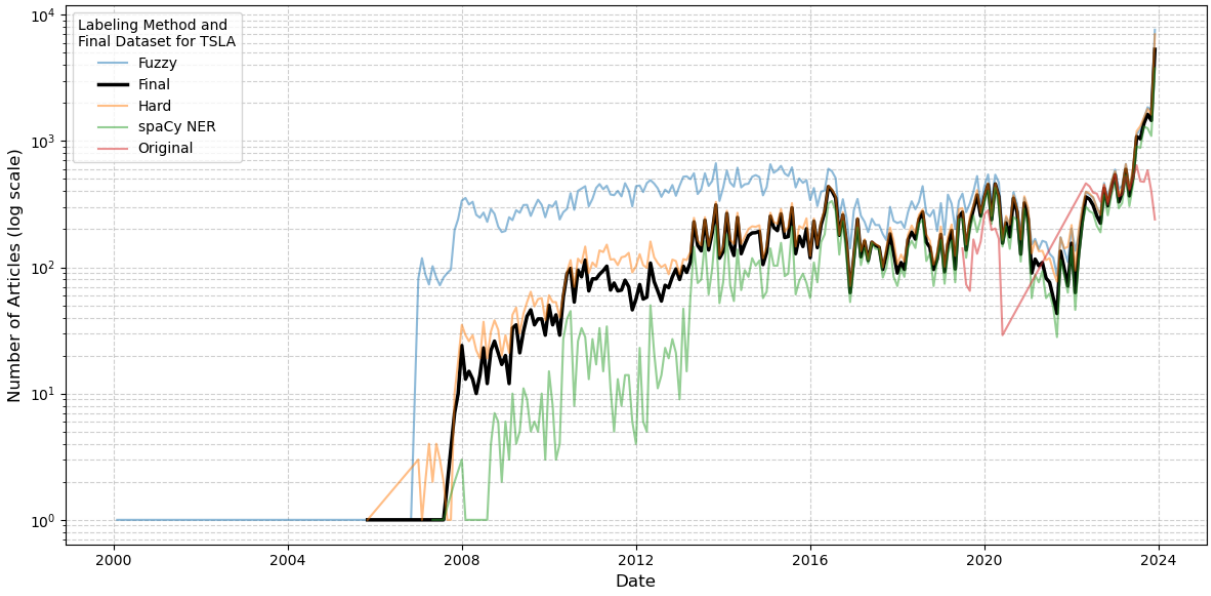

Figure B.7: Labeled TSLA Articles Over Time by Method (Log Scale)

Appendix C

Models

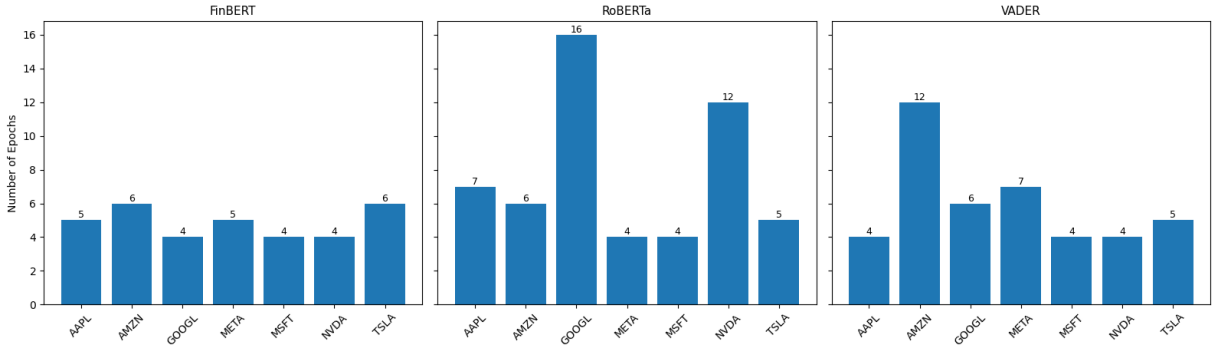

Figure C.1: Informer: Frequency of completed epochs across company and sentiment.

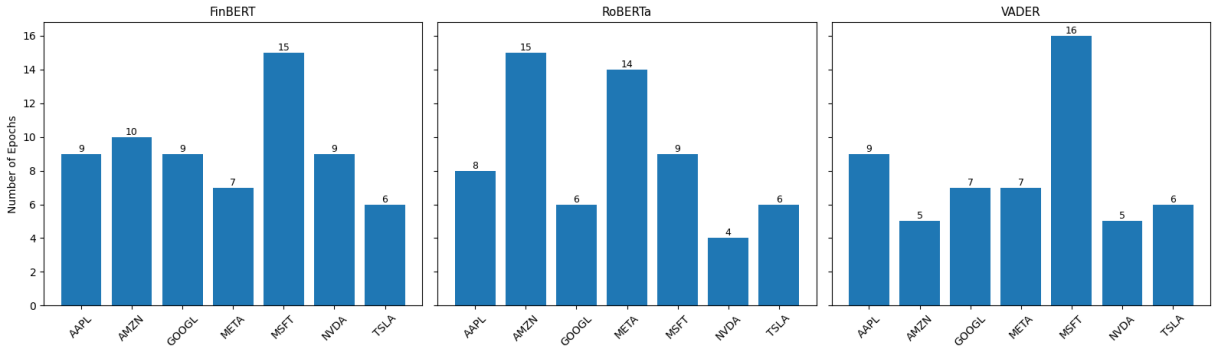

Figure C.2: PatchTST: Frequency of completed epochs across company and sentiment.

Table C.1: Informer Model Configuration

Parameter	**Value**
Input	
Target Variable	close_diff
Input Features	['volume', 'num_articles', 'sentiment_updated', 'sentiment_finbert', 'close_diff']
Feature Format	M (multivariate)
Temporal Frequency	Daily (d)
Sequence Length (seq_len)	60
Label Length ($label\_len$)	Tunable
Prediction Length ($pred\_len$)	1
Model Architecture	
Model	Informer
Encoder Input Size	5
Decoder Input Size	5
Output Size (c_out)	1
Model Dimension (d_model)	512
Feedforward Dimension (d_ff)	2048
Number of Attention Heads	8
Encoder Layers (e_layers)	3
Decoder Layers (d_layers)	2
Stacked Encoder Configuration	Not used
Attention Type	ProbSparse
Embedding Type	timeF
Activation Function	ELU
Distillation	Enabled
Dropout	Tunable
Training Settings	
Training Epochs	30
Batch Size	Tunable
Early Stopping Patience	3
Learning Rate	Tunable
Learning Rate Adjustment	type1
Loss Function	Mean Squared Error (MSE)
Random Seed	1337

Table C.2: PatchTST Model Configuration

Parameter	**Value**
Input	
Target Variable	close_diff
Input Features	['volume', 'num_articles', 'sentiment_updated', 'sentiment_finbert', 'close_diff']
Feature Format	M (multivariate)
Temporal Frequency	Daily (d)
Sequence Length (*seq_len*)	60
Prediction Length (*pred_len*)	1
Patch Length (*patch_len*)	Tunable
Stride	Tunable
Model Architecture	
Model	PatchTST
Encoder Layers (*e_layers*)	3
Decoder Layers (*d_layers*)	2
Encoder Input Size (*enc_in*)	5
Decoder Input Size (*dec_in*)	5
Output Size (*c_out*)	1
Dropout	Tunable
Activation Function	ELU
Training Settings	
Training Epochs	30
Batch Size	Tunable
Early Stopping Patience	3
Learning Rate	Tunable
Learning Rate Adjustment	type1
Loss Function	Mean Squared Error (MSE)
Random Seed	1337
GPU Acceleration	Enabled

Stock	Model	Dropout	Batch Size	Learning Rate	Label Length
AAPL	FinBERT	0.3016	128	0.00004	45
	RoBERTa	0.2259	256	0.00008	45
	VADER	0.3968	128	0.00003	30
AMZN	FinBERT	0.3888	512	0.00032	30
	RoBERTa	0.4152	512	0.00003	15
	VADER	0.1709	256	0.00002	15
GOOGL	FinBERT	0.4265	128	0.00036	30
	RoBERTa	0.1205	512	0.00365	15
	VADER	0.4875	512	0.00025	45
META	FinBERT	0.1230	512	0.00003	30
	RoBERTa	0.4947	256	0.00005	15
	VADER	0.2181	256	0.00007	15
MSFT	FinBERT	0.3691	128	0.00022	30
	RoBERTa	0.1162	256	0.00016	15
	VADER	0.2765	256	0.00006	15
NVDA	FinBERT	0.2433	512	0.00023	15
	RoBERTa	0.2081	256	0.00009	15
	VADER	0.4484	128	0.00013	45
TSLA	FinBERT	0.4835	128	0.00027	45
	RoBERTa	0.3325	512	0.00017	30
	VADER	0.4574	256	0.00381	45

Table C.3: Optimal Hyperparameters for Informer per Stock and Sentiment Model

Stock	Model	Dropout	Batch Size	Learning Rate	Patch Length	Stride
AAPL	FinBERT	0.4776	128	0.00140	16	10
	RoBERTa	0.3299	512	0.00329	32	10
	VADER	0.0526	128	0.00140	48	8
AMZN	FinBERT	0.0526	128	0.00007	16	8
	RoBERTa	0.2358	512	0.00011	32	12
	VADER	0.2708	128	0.00032	32	12
GOOGL	FinBERT	0.1073	512	0.00021	48	10
	RoBERTa	0.3871	128	0.00037	32	12
	VADER	0.3480	128	0.00032	16	10
META	FinBERT	0.1079	512	0.00021	48	10
	RoBERTa	0.2341	512	0.00006	48	8
	VADER	0.0871	128	0.00012	48	10
MSFT	FinBERT	0.1342	256	0.00107	16	8
	RoBERTa	0.4967	512	0.00074	16	12
	VADER	0.3100	512	0.00985	16	8
NVDA	FinBERT	0.3944	256	0.00112	16	8
	RoBERTa	0.3567	128	0.00050	32	12
	VADER	0.2868	128	0.00050	16	8
TSLA	FinBERT	0.4037	128	0.00242	32	12
	RoBERTa	0.4841	128	0.00018	32	8
	VADER	0.4673	128	0.00002	48	8

Table C.4: Optimal Hyperparameters for PatchTST per Stock and Sentiment Model

Appendix D

AI Statement

Tools used:

1. ChatGPT
2. Google Gemini

Generative AI has been used to support technical aspects of the thesis, particularly to debug code and to find solutions for coding problems, such as orienting the various Python libraries. Generative AI was also used to automate the creation and formatting of LaTeX tables in Chapters 3 through 5, and to adapt BibLaTeX to comply with the LUSEM Harvard referencing style. Additionally, AI tools were employed to improve language quality by providing feedback on word choice, structure, and academic tone throughout the thesis. Support was also used to enhance clarity and ensure correctness in the mathematical notation and explanations. All topic selection, research questions, methodological choices, analysis, interpretations, and conclusions were entirely carried out by the authors.