universität wien

# MASTERARBEIT | MASTER'S THESIS

Titel | Title

Predicting Daily Stock Price Movements Using GPT-3.5-Turbo


verfasst von | submitted by

Michael Knaus BA


angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2024

| | |
|---|---|
| Studienkennzahl lt. Studienblatt \| Degree programme code as it appears on the student record sheet: | UA 066 977 |
| Studienrichtung lt. Studienblatt \| Degree programme as it appears on the student record sheet: | Masterstudium Business Analytics |
| Betreut von \| Supervisor: | Dipl.-Inform. Dipl.-Volksw. Dr. Christian Westheide |

# Acknowledgements

I would like to thank all those who supported me during the preparation of this master's thesis, as well as during my previous years of study. I want to express my deepest gratitude to my partner Hannah. You stood by my side with interest and motivation throughout the entire writing process. Without your support, I would not have been able to complete this thesis and this master's degree so successfully and promptly. Additionally, I would like to especially thank my parents, who made my studies possible through their support. I am infinitely grateful to both of you for that. I also want to mention Prof. Westheide, who supervised and reviewed my master's thesis. I would like to sincerely thank you for the smooth communication and constructive suggestions during the preparation of this thesis. Finally, I would like to thank my friends and fellow students for their support throughout my studies.

# Abstract

In this thesis, the potential of GPT-3.5-Turbo in predicting stock returns using sentiment analysis of news headlines is investigated. The experiments of this study can be divided into two parts. In the first part, the results by Lopez-Lira and Tang (2023) are reproduced with a largely equivalent methodology, albeit with lower magnitude. It can be concluded that stock returns are predictable on a daily horizon with the help of GPT-3.5-Turbo. Apart from reproducing Lopez-Lira and Tang (2023)'s results, it is demonstrated that return predictability persists in 2023. Both limits-to-arbitrage and underreaction to news seem to play a role in explaining the cumulative returns of a self-financing long-short strategy, but the effect of limits-to-arbitrage is more pronounced. Predictability is larger for small caps and for short trades and only a small part of positive returns seem to persist after transaction costs. The study also finds strong evidence that a machine learning model trained on numerical data can complement large language models in the return prediction task of this study, if this model is used to filter news headlines before the sentiment classification task. In the second part of this study, three different optimization methods for large language models are tested: Prompt engineering, few-shot learning and fine-tuning. Prompt engineering is a cost effective and fairly straightforward approach to improve performance on the prediction task of this study. Few-shot learning results show that performance between the classification task, measured in accuracy, and trading strategy, measured in Sharpe ratio, can diverge. Fine-tuning shows mixed results, but no definitive conclusions on its merits for performance improvements should be drawn, as this study is severely limited by budget constraints. Finally, interesting directions for future research in the areas of prompt engineering, few-shot learning and fine-tuning are suggested.

# Kurzfassung

In dieser Arbeit wird das Potenzial von GPT-3.5-Turbo zur Vorhersage von Aktienrenditen mittels Sentiment-Analyse von Nachrichtenüberschriften untersucht. Die Experimente dieser Studie lassen sich in zwei Teile gliedern. Im ersten Teil können die Ergebnisse von Lopez-Lira and Tang (2023) mit einer weitgehend äquivalenten Methodologie reproduziert werden, wenn auch in geringerer Größe. Es kann gefolgert werden, dass Aktienrenditen mit Hilfe von GPT-3.5-Turbo über einen eintägigen Zeitraum vorhergesagt werden können. Abgesehen von der Reproduktion der Ergebnisse von Lopez-Lira and Tang (2023) wird gezeigt, dass die Vorhersagbarkeit der Renditen auch im Jahr 2023 anhält. Sowohl Limits-to-Arbitrage als auch Underreaction to News scheinen eine Rolle bei der Erklärung der kumulativen Renditen einer selbstfinanzierenden Long-Short-Strategie zu spielen, jedoch ist der Effekt von Limits-to-Arbitrage stärker ausgeprägt. Die Vorhersagbarkeit ist größer bei Small Caps und bei Short-Trades und nur ein kleiner Teil der positiven Renditen scheint nach Transaktionskosten bestehen zu bleiben. Die Studie findet auch starke Belege dafür, dass ein Machine Learning Modell, das mit numerischen Daten trainiert ist, Large Language Models in dem Return Prediction Task dieser Studie ergänzen kann, wenn dieses Modell verwendet wird, um Nachrichtenüberschriften vor der Sentiment-Klassifizierung zu filtern. Im zweiten Teil dieser Studie werden drei verschiedene Optimierungsmethoden für Large Language Models getestet: Prompt Engineering, Few-Shot Learning und Fine-Tuning. Prompt Engineering ist ein kostengünstiger und relativ einfacher Ansatz zur Leistungsverbesserung im Prediction Task dieser Studie. Die Ergebnisse des Few-Shot Learning zeigen, dass die Performance zwischen Classification Task, gemessen an der Accuracy, und der Handelsstrategie, gemessen an der Sharpe-Ratio, divergieren kann. Fine-Tuning zeigt gemischte Ergebnisse, aber es sollten keine finalen Schlussfolgerungen über dessen Nutzen zur Leistungsverbesserung gezogen werden, da diese Studie stark durch Budgetbeschränkungen eingeschränkt ist. Schließlich werden interessante Richtungen für zukünftige Forschung in den Bereichen Prompt Engineering, Few-Shot Learning und Fine-Tuning vorgeschlagen.

# Contents

# List of Tables

# List of Figures

# 1. Introduction

The development of large language models has made tremendous progress in recent years, leading to various new application areas of those models. One such application area is finance. While numerical data has long been used for prediction tasks in finance, the emerging analysis of textual data becomes even more powerful with the advent of large language models. It is well established that stock returns are predictable based on news sentiment (Tetlock, 2007; Tetlock et al., 2008; Garcia, 2013; Jiang et al., 2021). Lopez-Lira and Tang (2023) are among the first to show that large language models can predict stock returns on a daily horizon using sentiment analysis of news headlines. The authors use various language models for sentiment analysis of news headlines and calculate different trading strategies based on the output labels of the classification task. However, they do not investigate any optimization methods for large language models. The experiments in this study can be divided into two parts. In the first part, GPT-3.5-Turbo is used to replicate the classification task and trading strategies reported in Lopez-Lira and Tang (2023). The results are taken as a baseline for optimization. In the second part, prompt engineering, few-shot learning and fine-tuning is leveraged to optimize the baseline.

In short, the results reported by Lopez-Lira and Tang (2023) can be replicated, albeit with lower magnitude. In the period between October 2021 and December 2022, a self-financing long-short strategy yields a cumulative return of 147% and a Sharpe ratio of 2.83, without the application of any optimization methods. Return predictability persists in 2023. Both limits-to-arbitrage and underreaction to news seem to play a role in explaining the cumulative returns, but the effect of limits-to-arbitrage is more pronounced. This study finds strong evidence that a machine learning model trained on numerical data can complement large language models in the return prediction task of this study, if this model is used to filter news headlines before the sentiment classification task. Prompt engineering is a cost effective and fairly straightforward approach to improve Sharpe ratio and cumulative returns. On the test set, this optimization method produces an increase of 0.11 in terms of Sharpe ratio and 9% in terms of cumulative returns of a self-financing long-short strategy. Few-shot learning shows promising, but also mixed results. It improves accuracy on the classification task, but not Sharpe ratio and cumulative returns of a self-financing long-short strategy. Fine-tuning underperforms compared to the baseline on the validation set and slightly outperforms on the test set, but no definite conclusion can be drawn for this optimization method, as budget constraints severely limited this study.

This study proceeds as follows. The second chapter reviews literature about large language models. First, the term large language model is formally defined. Then, the architecture of GPT-3.5-Turbo is reconstructed from various publications. Lastly, different learning paradigms for large language models are discussed. The third chapter reviews literature about return predictability of stock prices. The reaction of stock prices to news and the use of natural

language processing methods for stock price prediction are discussed. The fourth chapter presents the methodology of the study. It provides details about the creation of the dataset, the setup of the sentiment classification task and the calculation of the trading strategy. The fifth chapter presents the results. It is divided into three subsections. The first subsection establishes baseline results by following the methodology presented in chapter 4. In the second subsection, optimization methods are applied to the baseline. The third subsection investigates the impact of transaction costs on trading strategy returns. The sixth chapter concludes the study and recommends directions for future research.

# 2. Large Language Models

This chapter first introduces the concept of word occurrence in a language as a probability distribution and gives a formal definition of the term autoregressive large language model. Then, important concepts from past language models are discussed. Next, the Generative Pretrained Transformer architecture is reconstructed by reviewing published code and articles by OpenAI. In particular, a sketch of GPT-3.5-Turbo's architecture is presented. Finally, relevant literature about in-context-learning and fine-tuning is reviewed.

## 2.1. Definition

In simple terms, a language model calculates the probabilities that any next word will occur given the known past words of a text. For instance, given the text `"I am"` it calculates a probability of the next word being `"Michael"`. Formally, text of a language consists of a vocabulary $V$ of tokens $w \in V$. A token is the smallest chosen unit of text and can be a character, sub-word unit, word or larger, while a text sequence is a sequence of tokens from the set of all text sequences $\Omega = \{(w_1, w_2, \dots)|w_i \in V\}$. It is assumed that the occurrence of text sequences in a language is characterized by a probability distribution $P$ over the sample space $\Omega$. The event that the first $n$ tokens of a text sequence are known is written as $w_1^n \subset \Omega$. A text sequence can be thought of as the result of a stochastic process of random variables $W_i$, where each $W_i$ describes the occurrence of a particular token at sequence position $i$. By using the definition of conditional probability recursively, a text sequence probability can be factorized into conditional probabilities of a next token occurring given the preceding known text sequence. The probability that a text sequence will occur in a language can thus be described by

$$\begin{aligned} P(w_1^n) &= P(W_1 = w_1 \cap \dots \cap W_n = w_n) \\ &= P(w_1^{n-1}) * P(w_n|w_1^{n-1}) \\ &= P(w_1) * \prod_{i=2}^{n} P(w_i|w_1^{i-1}) \end{aligned} \tag{2.1}$$

A language model is a function $\hat{P}$ that approximates the probability distribution $P$ of a language.

The above idea that a language can be described by a sequence of states and that the probability of this state sequence can be modelled by using the recursion $P(w_1^n) = P(w_1^{n-1}) * P(w^n|w_1^{n-1})$ was already developed in the 1970s (Bahl et al., 1975). However, in this early

formulation each state was not only described by tokens, but also by part of speech tags. The modern definition provided above can at least be traced back to Bahl et al. (1983) and Brown et al. (1992).

Note that by the given definition, a language model is characterized by sequential unidirectional processing of text that goes left-to-right. In practice, previously predicted tokens are used as inputs for new predictions at model inference, a paradigm commonly called autoregressive language modelling (Brown et al., 2020). As Blitzstein and Hwang (2019, p.53) point out, the factorization order of tokens in equation (2.1) can be chosen arbitrarily. While this fact is leveraged in some models such as XLNet (Yang et al., 2019), such permutation language modelling is rather uncommon. Another modelling approach by Devlin et al. (2019) introduced BERT, which uses a learning objective that does not calculate next token probabilities given preceding text sequences as context. Even though such models are commonly called masked language models, they do not fit the formal definition of a language model, since they do not straightforwardly assign text sequences to probabilities.

There is no agreed upon threshold on when a language model is large. The term large is rooted in the fact that neural networks (NNs), which currently achieve state-of-the-art language modelling results, require the processing of large amounts of data compared to contemporary standards. This fact was already discussed during the advent of NNs for language modelling (Bengio et al., 2003). However, since around 2020 the trend emerged to call any contemporary state-of-the-art language model a large language model (Radford et al., 2019b).

The experiments in this thesis are conducted with GPT-3.5-Turbo. This NN is a large autoregressive language models (Radford et al., 2018b, 2019b; Brown et al., 2020; Ouyang et al., 2022).

## 2.2. Important Concepts from Past Language Models

Despite the elegantly sounding analogy of biological and artificial NNs, it was initially not obvious that such a model would be suitable for a language model. In fact, the first popular models were $N$-gram models. In such models, it is assumed that for a text sequence $w_1^n$, the approximation $P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-c}^{n-1})$ holds (Bahl et al., 1983). Formally, this approximation can be seen as a $c$-order Markov assumption. Since for any contemporary language model there are coherent texts too long to be processed as a whole, the size of $c$ plays still a role for current models and is commonly referred to as context length or context window.

For a vocabulary $V$ with $|V|$ tokens and a training set with $T$ tokens, a simple $1$-gram model has $V - 1$ model parameters (i.e. all single token probabilities) that are estimated on a count basis. If $C(w_i)$ is the number of times $w_i$ occurs in the training set, then $\hat{P}(w_i; T) = \frac{C(w_i)}{T}$. At inference, the model is a discrete lookup table of those learned probabilities. The extension to $N$-gram models with larger $N$ entails the problems of exponential parameter growth and sparse data, since in general $V^N - 1$ parameters have to be learned, but most of the possible $N$-grams occur only zero or one time during training. Those shortcomings can be mitigated through interpolated estimation or class-based models (Brown et al., 1992). However, those two problems could not be overcome and led to the rise of NNs.

In its most basic form, NNs are characterized by layers of linear transformations with possibly non-linear activation functions. How those transformations, activations, and other design choices are combined within a NN is commonly referred to in articles as the architecture of the NN (Bengio et al., 2003; Vaswani et al., 2017; Radford et al., 2018b).

Bengio et al. (2003) present the first feedforward architecture that learns token embeddings and next token prediction simultaneously, where a token embedding is a real-valued vector representation of a token. It is useful to examine this architecture in detail because it is the first to combine several important concepts that are still used in state-of-the-art language models. Those concepts are a lookup layer with token embeddings, gradient-based learning with a the maximum log-likelihood learning objective, perplexity as an evaluation metric, and probability outputs over the entire vocabulary for each prediction using softmax.

The NN consists of three layers: The first layer is a lookup layer that takes a text sequence $w_{i-c}, \ldots, w_{i-1}$ of context length $c$ and looks up the $c$ corresponding $m$-dimensional token embeddings from an embedding matrix $E \in \mathbb{R}^{m \times |V|}$. Conceptually, one can think of the lookup operation as $w_i$ being an one-hot encoded vector such that the chosen embedding is $Ew_i$. The chosen embeddings are concatenated to a vector $x \in \mathbb{R}^{c*m}$ and serve as input to the hidden layer. The $h$-dimensional hidden layer uses biases $a$, weights $A$, and $\tanh$ as activation. The output layer uses biases $b$, weights $B$, a residual connection $C$ to $x$, and a softmax activation function. In total, the model has learnable parameters $\Theta = (E, a, A, b, B, C)$. The probability that a token $w_i$ with vocabulary index $k$ is the next token is then calculated as

$$\begin{aligned} x &= (Ew_{i-c}, \ldots, Ew_{i-1}) \\ y &= Cx + b + B \tanh(Ax + a)B \in \mathbb{R}^{|V|} \\ \hat{P}(w_i | w_{i-c}^{i-1}; \Theta) &= \frac{\exp(y_k)}{\sum_j \exp(y_j)} \end{aligned} \tag{2.2}$$

For $T$ tokens in the training set, a learning rate $\epsilon \in \mathbb{R}$ and disregarding the regularization function $R$ mentioned in the paper, the parameters $\Theta$ are optimized for maximum average next token log-likelihood via stochastic gradient ascent

$$\begin{aligned} L(w_1^T) &= \frac{1}{T} \sum_{i=1}^{T} \log \hat{P}(w_i | w_{i-c}^{i-1}; \Theta) \\ \Theta &\leftarrow \Theta + \epsilon \frac{\mathrm{d}L}{\mathrm{d}\Theta} \end{aligned} \tag{2.3}$$

Note that for every next token prediction, the probabilities of all tokens $w \in V$ are calculated, but only the gradient of $\hat{P}(w_i | w_{i-c}^{i-1}; \Theta)$ for the correct next token $w_i$ is used for weight updates. Additionally, in practice target labels might be smoothed and gradient updates implemented in batches smaller than $T$.

The described language model yielded state-of-the-art results with about 20% lower perplexities on the test set than the previously best $N$-gram models. In simple terms, the inverse of perplexity describes an average probability that a language model assigns to each next word in a text. Formally, perplexity of a language model is the exponential of its negative average

next token log-likelihood $\text{PPL}(\hat{P}) = 2^{-L(w_1^T)}$. If $-L(w_1^T)$ is rearranged such that the negative fraction and the sum are inputs to $\log$, then one gets $\text{PPL}(\hat{P})^{-1} = \sqrt[T]{\prod_i^T \hat{P}(w_i|w_{i-c}^{i-1}; \Theta)}$, motivating the simple interpretation of perplexity. The base in the definition of perplexity can be chosen arbitrarily, while the exponent can also be seen as the empirical entropy of the model. In that sense, the maximization objective is simultaneously minimizing the model entropy (Brown et al., 1992).

Bengio et al. (2003) point out that their feedforward architecture achieves state-of-the-art results by mitigating the sparsity and exponentiality problems of $N$-gram models. The real-valued embeddings capture a notion of similarity between tokens. Hinton (1986) showed early that feedforward NNs can learn embeddings that capture semantic meaning such as nationality, gender or age. In a later study, Mikolov et al. (2013) find that embedding spaces can capture semantic meaning to such a remarkable extent that vector operations can be used for analogy mining. For instance, let $x_w$ be the embedding for token $w$. Then $x_{\texttt{queen}} \approx x_{\texttt{king}} + x_{\texttt{woman}} - x_{\texttt{man}}$ turns out to be a good approximation because the vector difference $x_{\texttt{woman}} - x_{\texttt{man}}$ captures the semantic concept of this gender difference well. The sparsity problem is mitigated because the model can use its representation of gender and other semantic concepts implicit in its embeddings to assign a reasonable probability to cases such as $P(w_{\texttt{crowned}}|w_{\texttt{the}}, w_{\texttt{queen}}, w_{\texttt{is}})$, even though it might only have seen `the king is crowned` during training. The exponentiality problem is mitigated because the number of model parameters only scales linearly with $V$ and $c$ (Bengio et al., 2003).

A problem in Bengio et al. (2003)'s architecture is the fact that text is arguably inherently sequential, but at the time there was no mechanism that allowed feedforward NNs to process sequential data well. Additionally, the context length $c$ has to be defined prior to training and might not semantically fit the input sequence length, leading to a suboptimal concatenation of $x$ (Mikolov et al., 2010). For instance, if the context crosses sentence or paragraph boundaries of a text, part of the context might not lead to useful information for next token prediction. This is the case in the field of machine translation, which has co-evolved closely with standard language modelling.

As a consequence, recurrent NNs (RNNs) became a popular architecture between 2010 and 2017. RNNs process input text sequences token-by-token in sequential time steps. To predict the next token at time step $t$, the context tokens between $t - c$ and $t - 1$ are compressed into a single vector representation $h_{t-1} = \tanh(Wx_{t-1} + Vh_{t-2})$ and the next token prediction is simply the softmax of a linear transformation of $h_{t-1}$. Mikolov et al. (2010) report contemporary state-of-the-art results in terms of perplexity by using a RNN for speech recognition, but they do not use a lookup layer with token embeddings. Mikolov et al. (2013) show that their RNN trains token embeddings that capture remarkable syntactic and semantic meaning.

In principle, RNNs can learn predictions for arbitrary context length $c$, but in practice they are subject to a training problem called vanishing gradient. The partial derivatives of weights far away from the final output layer are a sum of long products, where most factors are close to zero since weight matrices are initialized with small values and the derivative of $\tanh$ maps to $(0, 1]$. Hochreiter and Schmidhuber (1997) propose a RNN architecture called the LSTM, which mitigates the vanishing gradient problem by introducing gating mechanisms. Since the main focus of this thesis lies on GPTs, a detailed explanation of the LSTM architecture is omitted.

Sundermeyer et al. (2012) are the first to use an LSTM architecture in the context of language modelling. They report lower perplexity of about 8% when using LSTMs instead of standard RNNs.

Sutskever et al. (2014) use an LSTM architecture in the context of machine translation by leveraging an encoder-decoder architecture. Conceptually, this architecture consists of a separate LSTM for each language. Assume that a German sentence $x_1^n$ must be translated into the English sentence $y_1^m$, where each $x_i$ and $y_j$ are token embeddings. The first model, the encoder, maps $x_1^n$ to the last hidden state $h_n$. The second model, the decoder, is an autoregressive language model, where the first hidden state is $s_0 = h_n$. The German sentence is then represented by the fixed-length vector $s_0$ and used as context for each translation step $y_i$. Bahdanau et al. (2016) point out that this architecture creates a bottle neck at $s_0$. They argue that the generation of each $y_i$ should get its most important information from different parts of $x_1^n$, meaning at each step $s_0$ should be different. For instance, if the two sentences are `Ich heiße Michael` and `My name is Michael`, then $y_4$ should be mostly influenced by $x_3$, while $y_2$ should mostly attend to a combination of $x_1$ and $x_2$. The authors propose the attention mechanism to implement this idea. In simple terms, for each English token $y_i$ and German token $x_j$ attention calculates the probability that $y_i$ depends on $x_j$. In the example, attention gives a probability that `name` depends on `heiße`. In technical terms, the attention output for a translation step $y_i$ is the expected value of the encoding of $x_1^n$ at this step. Since `name` should depend mostly on the tokens `ich` and `heiße`, the attention output at translation step $y_2$ will be a linear combination of vector representations of each German word, where the probability weights will be highest for `ich` and `heiße`.

Vaswani et al. (2017) propose a machine translation encoder-decoder architecture that only uses attention and completely dispenses with RNNs. They call this architecture transformer. Most importantly for the purpose of this thesis, Vaswani et al. (2017) introduce a variant of attention called masked self-attention, which can be directly used for language modelling. Note that they use a different notation, where a sequence $x_1^n$ of $d$-dimensional token embeddings is encoded row-wise in a matrix $X \in \mathbb{R}^{n \times d}$. This interprets the matrix $X$ more as a table of instances than as column vectors that are linearly transformed by the learned weights, but effectively it makes no difference as $(XW)^T = W^T X^T$. Since the same row-wise notation is used in the articles that introduce GPTs, it will also be used from now on in this thesis.

The transformer uses a variant of attention, where the similarity function within softmax is the scaled dot-product. Suppose $X$ and $Y$ encode row-wise $d$-dimensional token embeddings of sequence lengths $n$ and $m$, $\mathtt{sm}$ is the softmax function and $\alpha_{i-1} \in \mathbb{R}^{n \times 1}$ is the $(i-1)$-th row of a probability weight matrix $\alpha$. Query, key and value matrices are calculated as $Q = YW^q = (q_1, \ldots, q_m)^T$, $K = XW^k = (k_1, \ldots, k_n)^T$ and $V = XW^v = (v_1, \ldots, v_n)^T$. Attention output for a single attention layer at translation step $y_i$ is then defined as

$$\begin{aligned} \alpha_{i-1,j} &= \mathtt{sm}(\frac{1}{\sqrt{d}} q_{i-1}^T k_j) \\ \mathtt{attn}(y_{i-1}, X) &= \alpha_{i-1}^T V = \sum_j^n \alpha_{i-1,j} v_j \end{aligned} \tag{2.4}$$

As in the original paper by Bahdanau et al. (2016), the attention output is the expected encoding of $X$ for translation step $i$ of $y_i$. Note that apart from $X$, the prediction of the next token $y_i$ in the translation only depends on $y_{i-1}$. This is the case because $q_{i-1}$ only depends on $y_{i-1}$, as $q_{i-1}^T = y_{i-1}^T W^q$. In the architecture of the actual transformer decoder, a masked self-attention mechanism precedes the attention described in equation (2.4) such that $y_{i-1}$ is already a vector representation of the entire text sequence $y_1^{i-1}$ up to translation step $i-1$. Suppose that for the earlier example the next translation step is $y_3 =$ `is` and $y_2$ is the vector representation of the already translated text sequence `My name`, then the attention output $\texttt{attn}(y_2, X)$ is a vector representation of `Ich heiße Michael. My name`. This vector representation is then used to calculate the probability that the next token $y_3$ equals `is`.

In GPT-3.5-Turbo, only the masked self-attention mechanism is used for language modelling. In a language modelling task, the target sequence equals the input sequence, so $Y = X$. Additionally, the target tokens are offset to the right by one position. As described in equation (2.4), the $(i-1)$-th row of the attention weight matrix $\alpha$ is used for the prediction of token $i$. However, during training the model should not be able to generate the token $i$ by learning to put all attention on this token, essentially setting $\alpha_{i-1,i} = 1$. To prevent the model from cheating by looking into the future, the attention probabilities of tokens after the context are masked out, meaning they are set to $\alpha_{ij} = 0$ for $j > i$. Note that in the transformer architecture, the attention function input does not depend on sequential calculation of hidden states, making parallelization of attention within one layer possible. Thus, the calculation of masked self-attention can be more compactly written as

$$\texttt{self\_attn}(X; W^q, W^k, W^v) = \texttt{sm}(\frac{1}{\sqrt{d}} Q K^T) V \tag{2.5}$$

Vaswani et al. (2017) find that each masked self-attention output captures different syntactic and semantic meaning, if several outputs are used in the same layer. They call each attention function an attention head. For $h$ heads per layer and an adapted scaling factor of $\sqrt{\frac{d}{h}}^{-1}$, each head has its own parameters $\Theta_i = (W_i^q, W_i^k, W_i^v)$ with respective shapes of $d \times (\frac{d}{h})$. The individual heads are horizontally concatenated and joined by a linear transformation in what is called a multi-head attention (MHA) layer

$$\texttt{mha}(X; \Theta_1, \dots \Theta_h, W^O) = \texttt{Concat}(\texttt{self\_attn}(X; \Theta_1), \dots, \texttt{self\_attn}(X; \Theta_h)) W^O \tag{2.6}$$

As already mentioned, the $(i-1)$-th row of an attention head calculates a vector representation of the sequence $w_{i-c}^{i-1}$ for the next token prediction $w_i$. $W^O$ compresses the sequence representations of all heads into a single representation. In a transformer decoder, several layers of MHA are stacked. This means that the $(i-1)$-th row of each head in the second layer calculates a single sequence representation of the $c$ compressed sequence representations from the first layer. Those $h$ representation get then compressed into a single refined sequence representation of $w_{i-c}^{i-1}$. For every MHA layer, this stacking should continually refine the sequence representation of the initial $c$ context embeddings, as different attention heads have been shown to capture different syntactical and semantic meaning (Clark et al.,

2019; Vig, 2019). For instance, Vig (2019) suggests that attention of GPT-2 captures more abstract concepts like gender bias. When prompted with `The doctor asked the nurse a question. She`, there are heads, where the token `She` attends mostly to `nurse`. If the model input is the same except that `She` is exchanged for `He`, then the token `He` attends mostly to `doctor`. This means that the next word generations are either conditioned on the nurse being female or the doctor being male. This hypothesis is supported by the model completions which are `...She said "I'm not sure what you're talking about."` and `...He asked her if she ever had a heart attack`. While this bias is unfortunate, it suggests that based on the textual training data, the MHA mechanism learns a world representation that captures more abstract concepts than simple language syntax.

MHA lies at the heart of the initial transformer's learning mechanism. While MHA is the main contribution of Vaswani et al. (2017), the initial transformer also uses other architectural concepts relevant to GPTs. These include positional embeddings, residual connections, layer normalization, and feed-forward networks. Positional embeddings are needed for the model to incorporate positional information because a feed-forward transformer could otherwise not encode token order (Vaswani et al., 2017). Residual connections and layer normalization are mainly used to help with model convergence (He et al., 2016; Ba et al., 2016). There is no justification in the paper why the feed-forward network is used and an answer is not obvious. Geva et al. (2020) suggest that especially the feed-forward networks close to the final output layer capture useful semantics by mimicking the key-value mechanism in attention, with the exception that relu is used instead of softmax. Since Vaswani et al. (2017) point out on several occassions in the paper that design choices were often made to simply facilitate parallelization, an easier explanation might be that two layer feed-forward networks are efficient to implement and the transformer as a whole showed good results.

The original transformer as well as the presented concepts that the transformer builds upon are the foundation to the introduction of GPTs.

## 2.3. Generative Pre-trained Transformers

To date of this writing, the best available GPT is the GPT-4o model. Even though one paper for each of GPT-1 to GPT-4 was published (Radford et al., 2018b, 2019b; Brown et al., 2020; OpenAI et al., 2024), the details about training set, training regime and architecture within each of them decreased steadily. While the training code for GPT-1 was directly released (Radford et al., 2018a), for GPT-2 only the model architecture code for inference has been made available by OpenAI (Radford et al., 2019a). Starting from GPT-3, neither the model code nor training set have been published and details about training regime became increasingly vague. From that point on, models are only available via the OpenAI API.

Based on the available information published by OpenAI, this section highlights selected details about the architecture, training set, and training regime of past GPTs. Eventually, a sketch of GPT-3.5-Turbo's architecture and training regime is presented, as GPT-3.5-Turbo is used in the experiments of this thesis. While the sketch and other details might be inaccurate in some aspects and most of the architecture remains directly inaccessible via API anyway, a conceptual architectural understanding is beneficial for handling the model.

GPT-1 was released in June 2018 and used a semi-supervised training paradigm (Radford et al., 2018b). The authors describe this paradigm as a two step approach. In the first step, called pre-training, a model is trained on a large unlabelled dataset. In the second step, called fine-tuning, the pre-trained model is further trained on a smaller labelled dataset and the learning objective is usually different from pre-training. This setup is motivated by the suspicion that the model can transfer useful knowledge from the pre-trainig objective directly to the fine-tuning objective.

Architecturally, GPT-1 is a decoder-only transformer inspired by Vaswani et al. (2017). Minimal adaptions to the original transformer decoder include learned positional embeddings and a `gelu` activation function in the feed-forward network (Hendrycks and Gimpel, 2016) instead of `relu`. A model input is first transformed into the sum of its token embeddings and positional embeddings and serves as input to a stack of several transformer blocks, where each block consists of layers of normalization, MHA and a feed-forward network. For pretraining, a standard maximum next token log-likelihood objective as described in equation (2.3) is used for learning. As in the original transformer, the final block output is linearly transformed by the token embedding matrix and used as softmax input for the final token probability calculation. During fine-tuning, the last block output is linearly transformed by an objective specific weight matrix. In this setup, GPT-1 can directly take input as contiguous text and needs no further architectural adaptions for down-stream tasks such as sentiment classification, entailment, paraphrase detection and multiple choice question answering. Radford et al. (2018b) show that a language modelling pre-training objective results in useful transfer to those natural language understanding tasks. Especially relevant for this thesis is the demonstrated transfer to sentiment classification.

GPT-1 leverages a sub-word level vocabulary using the byte-pair encoding algorithm introduced by Sennrich et al. (2015). In this machine learning algorithm, a training text, a base vocabulary and a number for the desired count of merges is given to the model. The model is often referred to as tokenizer. In principle, any set of tokens can be defined as base vocabulary, but usually it is chosen in a way that most text can be processed. An example could be the set of all individual alphabet characters. During training, the training text is first split into the base vocabulary tokens. Then, those tokens are merged based on bi-gram frequency for the predefined number of times. Each merged token is added to the base vocabulary as a vocabulary token. During inference, the tokenizer first splits an input text into base vocabulary tokens and then merges based on the learned tokens until convergence. The resulting vocabulary can achieve effective compression by leveraging the fact that words are often composites of smaller units. For instance, at the date of this writing the word `sweetish` is composed of the two tokens `sweet` and `ish` in the GPT vocabulary (OpenAI, 2024e).

Conceptually, the authors note that GPT-1's architecture was chosen to progress towards language processing systems that can be used for various down-stream tasks with minimal architectural adaptions. In this vein, Radford et al. (2019b) introduce GPT-2 as a scaled version of GPT-1 that totally dispenses with the objective specific top layer. Fine-tuning is abandoned altogether in this setting and tasks can be presented as next token prediction given context tokens. A new training set comprised of unpublished books is used for pre-training to ensure that training data has long-range dependencies.

Surprisingly to the authors, GPT-2 can still achieve state-of-the-art results in down-stream tasks, just through transfer from the pre-training language modelling objective. The authors hypothesize that by learning to model next token prediction, GPT-2 learns tasks that are implicit in the training set. For instance, approximately 0.025 percent of the training set is in French, while the vast majority is in English. From learning next token prediction in this dataset, the model learns English-French translation on a very basic level (Radford et al., 2019b).

GPT-2, and presumably also GPT-3.5-Turbo, uses byte-pair encoding on UTF-8 byte level basis instead of a Unicode code point basis. This means that with a base vocabulary of only 256 one-byte variations all possible Unicode characters and consequently any language input can be processed by GTP-2 (Radford et al., 2019b). However, as Brown et al. (2020) later point out, this comes at the big caveat that byte merges are based on a training set mostly comprised of English words. For instance, the word `Austria` is encoded as one token, whereas its German translation `Österreich` is encoded as the two tokens `Ö` and `sterreich` (OpenAI, 2024e). Since it is hardly imaginable that the token embedding for `Ö` will as efficiently encode semantics about the whole word `Österreich` as `sweet` does for `sweetish`, this shows that one should take into consideration tokenization when using a GPT for a down-stream task in a non-English language.

Architectural adaptions from GPT-1 to GPT-2 are minimal. They include the moving of layer normalization to the input of each transformer sub-block, additional normalization after the final block, and slightly different weight initialization schemes. The training set is enlarged through a curated version of web-scraped outgoing Reddit links (Radford et al., 2019b).

In GPT-2, results are achieved without fine-tuning, just by providing context via a contiguous input string. This contiguous input string is often referred to as prompt. With the introduction of GPT-3, different types of prompts are more stringently defined. Brown et al. (2020) contrast fine-tuning and in-context-learning (ICL). While fine-tuning is characterized by gradient updates, ICL is a paradigm in which the model only learns the task through information in the prompt. ICL can be further divided into zero-shot, one-shot and few-shot learning. In zero-shot learning, only the task is described in the prompt. In one-shot learning, the prompt contains a task description and exactly one demonstration of a desired input-output pair. In few-shot learning, the prompt contains the task description and more than one input-output demonstration.

Brown et al. (2020) claim that GPT-3 is a scaled version of GPT-2 with no architectural difference except that sparse attention patterns are used throughout the model. The training set is a mix of almost 500 billion byte-level tokens, which means that a substantial part of all available intelligible English text on the internet is used. At publication, the model achieved state-of-the-art results on language modelling benchmarks and respectable results on other natural language understanding tasks when used with ICL. Due to the scale of training data, the authors investigate how much of the training data is included in the test set of the investigated benchmark. During the empirical part of this thesis, it will be important to ensure that the dataset was not part of GPT-3.5-Turbos's pretraining. In practice, this will mean that instances in the dataset cannot be older than October 2021, since training data is used up to September 2021 (OpenAI, 2024c).

Note that for every generation step, GPT-3 returns the conditional probabilities over the entire vocabulary. Picking the next token becomes a non-obvious decoding problem. A greedy

approach often leads to sub-optimal results if longer sequences should be generated. In Brown et al. (2020) the authors experiment with temperature, beam search and nucleus sampling. Especially temperature and nucleus sampling are relevant today, as those hyperparameters are accessible via the OpenAI API. The term temperature is inspired by the fact that pre-softmax logits of a language model can be interpreted as energy of a Boltzmann distribution. Consequently, for pre-softmax logits $l$ and temperature $t$, the inverse of $t$ rescales the logits such that the next token is sampled from $\texttt{sm}(\frac{1}{t}l)$. Temperature close to zero pushes more probability mass towards tokens that were most likely in the initial distribution, while large positive temperature will flatten the initial distribution, making rare tokens more likely (Ackley et al., 1985). Nucleus sampling rescales the probability mass for all tokens whose probabilities sum up over a pre-defined parameter $p$. If $V^p$ is the smallest possible vocabulary subset such that $p' = \sum_{w \in V^p} \hat{P}(w|w_{i-c}^{i-1}) \geq p$, then for every token $w$ the intial probability distribution is rescaled either to $P(w|w_{i-c}^{i-1}) * \frac{1}{p'}$ or zero, depending if $w \in V^p$ (Holtzman et al., 2019).

Even though sampling can greatly improve next token generation for an open-ended text generation task, GPT-3 still lacks the dialog capabilities of GPT-3.5-Turbo. As Ouyang et al. (2022) point out, GPT-3 also often fails to follow human instructions. To mitigate this problem, the authors propose a three step paradigm to fine-tune a pre-trained GPT-3 model on human preferences. This approach falls under reinforcement learning from human feedback (RLHF). In the first step, called supervised fine-tuning, a pre-trained GPT-3 model is trained on human written (`prompt`, `completion`) pairs with a standard next token prediction objective. In the second step, a reward model is trained, which will serve as part of the objective function in the third step. The reward model is initialized as a smaller GPT-3 model with the unembedding layer substituted for a transformation that returns a scalar value. The training dataset consists of prompts with several possible completions. For each prompt, all completions are ranked relative to each other by human evaluators. The reward model takes a concatenated (`prompt`, `completion`) string as input and returns the scalar reward as output. For learning, the model objective is to maximize the scalar reward difference between a higher rated and lower rated completion, where both reward scalars can be interpreted as logits in a binary classification task. In the third step, the fine-tuned model from step one is further fine-tuned by the reward model. For each iteration, the fine-tuned model takes a sample prompt and creates a completion. The (`prompt`, `completion`) pair is the input to the reward model. The calculated scalar reward is then used as input for the objective function of step 3.

Approximately 3.5% of samples for step 2 of RLHF contain a classification task. The string format of those samples can be seen in the appendix of Ouyang et al. (2022). Since GPTs are fine-tuned on certain string formats for classification, it would be beneficial to know this format, since this would most likely lead to better results.

OpenAI's API was initially launched in June 2020 by supporting text completions for GPT-3 (OpenAI, 2020). In a text completion, the language model is provided with a context string and autoregressively generates new tokens until an end of sequence token is generated. In the aftermath of Ouyang et al. (2022)'s RLHF approach, OpenAI put increased focus on model alignment to enable multi-turn conversations. As a hallmark in this direction, OpenAI released ChatGPT in November 2022 on the basis of a RLHF aligned model that was referred to as GPT-3.5 (OpenAI, 2022), however API access was still restricted to text completion models. In

March 2023, access to chat completion models was extended to the OpenAI API and those models were referred to as GPT-3.5-Turbo (OpenAI, 2023a). In a chat completion, the prompt can consist of a system message and a series of user messages and assistant messages. The system message gives the model global user instructions that are applied to the entire chat history, a user message represents human input, and an assistant message a model output. To formalize this chat-like prompt input, OpenAI released Chat Markup Language (OpenAI, 2023c). Since a GPT still takes a tokenized string as input, the model needs to learn how to process chat-like interactions with a human. For this reason, Chat Markup Language includes the wrapper `<|im_start|>role ... <|im_end|>`, where `role` is one of `system`,`user` or `assistant`. In the current API implementation, a prompt for a chat completion is given as a list of dictionaries that is parsed into a raw Chat Markup Language string. In figure (2.1), an example for the chat `Human: How are you? Model: I am doing well!` is shown.

```
<|im_start|>system
You are ChatGPT, a large language model trained by OpenAI.<|im_end|>
<|im_start|>user
How are you?<|im_end|>
<|im_start|>assistant
I am doing well!<|im_end|>
```

Figure 2.1.: The prompt of a GPT chat completion in the form of a raw Chat Markup Language string (OpenAI, 2023c).

Note that the wrappers are no special vocabulary tokens. As of the date of this writing, OpenAI has published no information on how GPTs are trained to recognized the chat-like conversation style that should be implied by the wrappers. In fact, OpenAI has even deleted documentation about Chat Markup Language from the main branch of its Python Github repo such that the documentation is only accessible via the commit history (OpenAI, 2023c). Most likely, the model learns to interpret the wrappers during the supervised fine-tuning and reinforcement learning gradient updates in the RLHF paradigm, since this is also the case for Meta's open source model Llama 2. During training of Llama 2, the system message is simply prepended to the first user message and gradients are set to zero for all intermediate conversation steps such that only gradients for the first user message and the last user message-assistant message pair are updated (Touvron et al., 2023). During inference, the system message is also prepended to the first user message (Meta, 2023). If that is also the case for GPTs, it should make no difference for the classification task if the system message is used or more instructions are put into the beginning of the first user message, since those two get concatenated anyway.

Based on the reviewed resources in this section and definitions from the last sections, the GPT-3.5-Turbo architecture can be sketched as in figure (2.2). Suppose $\theta^{\texttt{norm}} = (f, g)$, $\theta^{\texttt{mlp}} = (W_1, b_1, W_2, b_2)$, $\theta_i^{\texttt{attn}} = (W_i^q, W_i^k, W_i^v)$ and $\theta^{\texttt{mha}} = (\theta_1^{\texttt{attn}}, \dots, \theta_h^{\texttt{attn}}, W^O)$ are parameters for a normalization layer, a feed-forward network, an attention head and a MHA layer. For a context length $c$, embedding dimension $d$ and an input $S \in \mathbb{R}^{c \times d}$, the norm function is applied row-wise to $S$, where $\mu$ and $\sigma$ are arithmetic mean and standard deviation of a row $s_i$ of $S$ and $\odot$ is point-wise multiplication. While the MHA function is defined in equation (2.6), layer

normalization and the feed-forward network are defined as

$$\begin{aligned} \texttt{norm}(s_i; \theta^{\texttt{norm}}) &= g \odot \frac{s_i - \mu}{\sigma} + f \\ \texttt{mlp}(S; \theta^{\texttt{mlp}}) &= \texttt{gelu}(SW_1 + b_1)W_2 + b_2 \end{aligned} \tag{2.7}$$

A transformer block has total parameters $\theta^{\texttt{B}} = (\theta^{\texttt{norm}}, \theta^{\texttt{mlp}}, \theta^{\texttt{mha}})$ and is defined as

$$\begin{aligned} A &= \texttt{mha}(\texttt{norm}(S; \theta^{\texttt{norm}}); \theta^{\texttt{mha}}) + S \\ \texttt{block}(S; \theta^{\texttt{B}}) &= \texttt{mlp}(\texttt{norm}(A; \theta^{\texttt{norm}}); \theta^{\texttt{mlp}}) + A \end{aligned} \tag{2.8}$$

Note that in GPT-2, the two norm functions within one block each learn individual parameters. The notation should be seen as providing an overview of which parameters are learned during training, while giving a formulation of the forward pass that is as accurate as possible.

Suppose now $X \in \mathbb{R}^{c \times |V|}$ is a one-hot encoded representation of a text sequence $w_{i-c}^{i-1}$, $W_e$ is a token embedding matrix, $W_p$ is a positional embedding matrix and $n$ is the number of transformer blocks. GPT-3.5-Turbo is a function parameterized by $\Theta = (W_e, W_p, \theta_1^{\texttt{B}}, \cdots, \theta_n^{\texttt{B}})$ and defined as

$$\begin{aligned} S_0 &= XW_e + W_p \\ S_j &= \texttt{block}(S_{j-1}; \theta_j^{\texttt{B}}) \\ L &= \texttt{norm}(S_n; \theta^{\texttt{norm}}) W_e^T \\ \hat{P}(X; \Theta) &= \texttt{sm}(L) \in \mathbb{R}^{c \times |V|} \end{aligned} \tag{2.9}$$

Note that while during training each row of $\hat{P}(X; \Theta)$ can be used for learning, during inference only the last row $l_c$ of the logit matrix $L$ is used for the next token prediction $\hat{P}(w_i | w_{i-c}^{i-1}; \Theta) = \texttt{sm}(l_c)$.

As already highlighted in this section, the architecture described in equation (2.9) and figure (2.2) is trained in a semi-supervised paradigm. The pre-training objective leverages next token prediction as described in equation (2.3). During fine-tuning, $\Theta$ is updated in two separate phases. In the first phase, inputs are given in an instruction or chat-like fashion with the same next token prediction objective as during pre-training. In the second phase, $\Theta$ is updated using a reward model in the objective as described in the paragraph discussing Ouyang et al. (2022).

Finally, it must be pointed out that GPT-3.5-Turbo is not a state-of-the-art model at the date of this writing. In July 2023, GPT-4 was launched for general access via the OpenAI API (OpenAI et al., 2024). While GPT-4 is the last model for which a technical report was launched (OpenAI et al., 2024), the more recently launched GPT-4-Turbo and GPT-4o were only announced via the OpenAI blog (OpenAI, 2023b, 2024b). All of the newer models cannot be used for the experiments in this thesis because they are too expensive or the cut-off date of their training data is too recent.

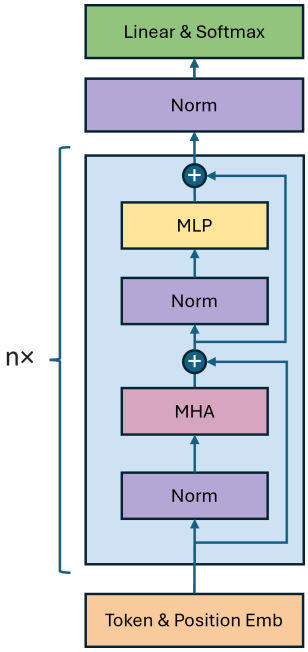


Figure 2.2.: Sketch of the GPT-3.5-Turbo architecture as formulated in equations (2.8) and (2.9). The design is visually based on the depiction in Radford et al. (2018b).

## 2.4. In-Context-Learning and Fine-Tuning

As mentioned in the last section, Brown et al. (2020) contrast fine-tuning and in-context-learning (ICL). Fine-tuning describes learning through gradient-based model parameter updates. ICL is a learning paradigm, in which model performance can be improved without gradient updates, just by being conditioned on an input string. The string input can be referred to as prompt. Finding the optimal prompt for a given task essentially becomes a hyperparameter optimization problem. Liu et al. (2023) call this search for an optimal prompt prompt engineering. The authors further introduce a three-step methodology for prompt creation.

The first step is called prompt addition. In this step, a template is chosen that includes a task instruction, a placeholder `[X]` for the input and a placeholder `[Z]` for the answer. Since GPT-3.5-Turbo is an autoregressive language model, `[Z]` will be at the end of the template. In the second step, called answer search, the optimal answer `Z` given a template filled with an input `X` is searched. For the purpose of this thesis, `Z` will simply be the GPT token output with the highest probability for the filled template. For instance, `X` = `Pepsi increases revenue` could be an input that gets inserted into a template `Classify the following headline as good or bad for Pepsi stock: [X] [Z]`. In the third step, called answer mapping, the model output `Z` is mapped to the actual target class of the classification task. In a trivial example, the model output could be `Good` with capital `G`, but the set of target classes is $\texttt{Y} = \{\texttt{good}, \texttt{bad}\}$, so `Good` must be mapped to `good`.

Liu et al. (2023) explicitly mention two relevant directions for optimization: Prompt template

engineering and prompt augmentation. In prompt template engineering, the task instruction as well as the positions and contents of the placeholders in the template from step 1 are optimized. In this thesis, different prompt templates are created manually based on best practice examples mentioned in the literature (Sahoo et al., 2024; White et al., 2023; Clavié et al., 2023) and the OpenAI online documentation (OpenAI, 2024d). Prompt augmentation also alters the prompt template from step 1 and is essentially what Brown et al. (2020) refer to as few-shot learning. In this optimization paradigm, two pressing problems are sample selection and sample ordering. Liu et al. (2021) propose a k-nearest-neighbor (kNN) based solution for the selection problem, where samples are selected based on embedding distance to the input of `[X]`. The authors hypothesize that the best ordering could be descending such that the closest instance in terms of distance is also closest to the test instance in the prompt. However, they also find evidence that optimal ordering might be dataset dependent. In their algorithm, all training instances are first embedded by an embedding model. Then, the k nearest embeddings to the embedding of the prompt template input are selected. Lastly, the k selected instances together with their respective target labels are inserted to the prompt template before the input placeholder `[X]`. In this thesis, Liu et al. (2021)'s approach is used with a descending sample ordering based on cosine similarity.

Fine-tuning via the OpenAI API is another optimization method available for GPT-3.5-Turbo. Available hyperparameters are the number of epochs, the batch size and a learning rate multiplier. Even though the online documentation gives a detailed instruction on how to fine-tune a model step-by-step, there is no exact documentation about the hyperparameters (OpenAI, 2024a). During implementation, it turned out that no extensive hyperparameter searching or testing of different fine-tuning setups was possible due to cost reasons. For the few fine-tuning tests conducted, all hyperparameters were chosen by default, as this should be seen as a reasonable starting point for hyperparameter search. There could be articles found that were posted on the OpenAI blog and also mostly stuck to default settings (Lin et al., 2022). However, no publications could be found, in which hyperparamter choices for the API were extensively commented on.

# 3. Previous Works in Finance

This chapter first introduces relevant studies that investigate stock price reaction to news. Stock price reaction is examined for aggregate markets and for individual stocks as well as for numerical and string data. Then, articles are examined that use neural networks or language models for a binary classification task to predict stock price returns. Specifically, Lopez-Lira and Tang (2023)'s results are introduced, which can be seen as motivation for the experiments in this thesis.

## 3.1. Stock Price Reaction to News

Stock price reactions to new information have been studied for a long time, particularly news containing numerical data. Earnings announcements are among the first news to be studied in this regard. In 1968, Ball and Brown (1968) already note that after earnings announcements, stock prices continue to drift up for positive news and down for negative news. Bernard and Thomas (1989) point out that the drift can either be explained by risk factors uncaptured by linear market models or by delayed price responses. The authors find evidence that delayed price responses cause the drift and that the delayed responses could be caused both by transaction costs and by failure of investors to instantly process new information. Chan et al. (1996) find similar evidence of investor's slow adjustment to new information. They report price drifts of at least six months after earnings release. The mentioned results suggest that new information causes a price drift in the direction of the initial sentiment that could potentially be exploited by a trading strategy.

While the first studies investigating numerical news data are comparatively old, the broad utilization of textual information in finance started only between 2005 and 2010. This field is commonly referred to as natural language processing (NLP). In the first finance applications, dictionaries were used to convert string data into numerical data that could then be processed for prediction tasks. Those dictionaries map each string directly to a sentiment. Tetlock (2007) investigates the impact of a Wall Street Journal column on stock market returns by using a general purpose dictionary. He shows that daily returns from 1984 to 1999 can be predicted by the number of negative words in the column. Loughran and McDonald (2011) argue that a general purpose dictionary leads to misclassification of financial words and propose one of the first dictionaries specific for financial contexts. Garcia (2013) uses the Loughran-McDonald dictionary and New York Times columns to study the effect of news sentiment on asset prices in the 20th century. His results support Tetlock (2007)'s findings that stock returns are predictable on a daily horizon. He additionally finds that return predictability is higher during recessions, suggesting that investors become driven by media sentiment during crises. Calomiris and Mamaysky (2019) use a machine learning instead of a dictionary approach to predict aggregate

stock market returns on a monthly frequency. In addition to sentiment, they use other features such as word unusualness and topical word context to predict risk and return in stock markets in 51 developed and emerging economies.

Other studies investigate the impact of news on individual stocks instead of aggregate market prices. Tetlock et al. (2008) use positive and negative dictionary words in news articles about individual S&P500 firms from 1980 to 2004 to predict daily returns. They find that stock market prices respond to negative news with a one-day delay. A long-short strategy based on word sentiment yields 21.1% cumulative returns per year before trading cost. However, all returns already disappear if trading costs are higher than 8 basis points, suggesting that market frictions play an important role in how information is incorporated in stock prices. Tetlock (2011) investigates how daily stock prices react to stale information. Staleness is defined as the average Jaccard similarity between a news headline about a firm and the 10 news headlines about the firm which were published before. The author finds that fresh news have a much higher stock price impact on news day than stale news. This suggests that filtering news headlines for freshness can improve returns of the trading strategy described in Tetlock et al. (2008). Jiang et al. (2021) use RavenPack to get news of individual firms published by the Dow Jownes Newswire. The final dataset includes over 5000 companies traded at the NYSE, NASDAQ or AMEX between March 2000 and October 2012. News sentiment is assigned based on initial stock returns on the news publication day. Stocks in the highest return decile are bought and stocks in the lowest return decile are sold short and held for one week. The authors report around 3.53% average monthly returns for an equal-weighted long-short portfolio. Their transaction costs model assumes around 18 basis points per trade, leaving 2.1% monthly average returns net of trading cost.

The reviewed literature suggests that daily stock returns are predictable based on news due to a price drift in the direction of the initial sentiment. There is mixed evidence on the question if positive returns persist after transaction costs. Initial studies used simpler sentiment measures such as negative word count and report lower trading strategy returns compared to more recent studies.

## 3.2. Language Models for Stock Price Prediction

The articles mentioned in the last section mostly included dictionary-based or conventional machine learning regression models to predict stock returns. In contrast to those approaches, Ding et al. (2014) use a neural network to solve event-based stock return prediction as a binary classification task. The authors use news headlines from Reuters and Bloomberg over the period from October 2006 to November 2013 to predict daily, weekly and monthly returns of the S&P500 index as well as 15 individual stocks. The authors use a standard machine learning approach, where the dataset is split into training, validation and test set. They report accuracies between 55 and 70% on the test set and show that accuracies are higher for daily return prediction and company specific news. This suggests that price drift might be higher on the daily level for prediction of individual stock returns based on company specific news. Ding et al. (2015) use the same dataset as Ding et al. (2014) to train two models: an embedding model and a neural network. The first model embeds the news headlines and the second takes the

news embeddings and returns a binary classification label to predict stock return directions. It is shown that this setup leads to approximately 5% higher accuracies than reported in Ding et al. (2014). Additionally, the authors implement a trading strategy with daily trades based on the classification labels and report positive returns before transaction costs.

Inspired by Ding et al. (2015)'s work, dos Santos Pinheiro and Dras (2017) use the same two-part model approach with a character-level language model instead of an embedding model. More precisely, a language model is trained on character level tokens using the same financial news dataset as Ding et al. (2014) and Ding et al. (2015). Then, this trained language model is used for initialization and extended by a feed forward layer. The extended model is then trained on the classification task. The authors show that their model yields similar accuracies as the model proposed by Ding et al. (2015). They also provide visual time-series results for the same trading strategy as proposed by Ding et al. (2015). The plot shows that the strategy consistently outperforms the S&P500 in terms of cumulative returns in the period from October 2006 to November 2013. However, the plot also shows much higher volatility and no risk adjusted metrics such as Sharpe ratio are reported. Xie et al. (2023) use GPT-3.5-Turbo for a binary stock return prediction task. The authors report between 50 and 56% accuracy depending on the dataset and conclude that GPT-3.5-Turbo is no better than simple linear regression models in predicting stock prices. However, in their prompt the authors include both numerical time series data in a csv format as well twitter data. Two potential issues could be that GPTs excel at processing string data instead of numerical data and that social media data might be even noisier than other news sources.

Lopez-Lira and Tang (2023) address the issues in Xie et al. (2023) and improve the classification task and trading strategy setup introduced by Ding et al. (2015). GPT-3.5-Turbo is used to assess whether news headlines are good, neutral or bad for firms' stock prices. Based on those three classification labels the authors implement different trading strategies for the time period between October 2021 and December 2022. They find that GPT labels exhibit high predictability for subsequent daily stock returns and that trading strategies based on the classification labels yield high Sharpe ratios. For example, a self-financing long-short strategy produces cumulative returns over 500% and a Sharpe ratio of 3.09. The authors attribute these findings to limits-to-arbitrage and underreaction to news. Limits-to-arbitrage means that market inefficiencies cannot be exploited due to market frictions such as transaction costs. Underreaction to news means that subsequent to news publication stock prices tend to drift in the direction of the initial news sentiment because investors find it hard to process new information. The underreaction to news explanation is supported by the fact that return predictability is substantial across different firm sizes and across time. Still the authors find that predictability is stronger among small firms and short trades, indicating that limits-to-arbitrage partly cause the captured price drift.

The work by Lopez-Lira and Tang (2023) motivates this thesis, as it not only presents novel results but also leaves interesting research gaps open. First, it is interesting to test whether the results can be replicated. Second, Lopez-Lira and Tang (2023) do not explicitly investigate any optimization methods for large language models. Possible optimization methods are prompt engineering, few-shot learning or fine-tuning.

# 4. Methodology

This chapter introduces the structure of the experiments in this study and the creation process of the dataset as well as how the classification task and the trading strategy are implemented.

## 4.1. Structure of the Experiments

The study can be conceptually split into two parts. In the first part discussed in section (5.1), the results presented by Lopez-Lira and Tang (2023) are replicated. In this part, it is attempted to stay as close as possible to the setup of the original study. Reasons for any deviations are justified in the following sections.

In the second part discussed in section (5.2), the setup from section (5.1) is taken as a baseline for optimization. In a classical machine learning setup, the dataset is split into training, validation and test sets. The setup is optimized for the Sharpe ratio of a self-financing long-short trading strategy that is calculated based on the labels of the classification task. Examined optimization methods include prompt engineering, few-shot learning and fine-tuning.

Finally, the setup with the best performance on the validation set is used to test if cumulative returns on the test set disappear after transaction costs. The test set is used for this purpose to create a realistic scenario, in which the best setup would be deployed to generate positive returns after transaction costs.

## 4.2. Dataset

Two major sources are used to create the dataset. Historical stock prices are acquired from the CRSP database. News headlines are acquired from RavenPack. Additionally, Yahoo Finance was used to get a risk-free rate for the US dollar. The examined time period is October 2021 to December 2023. The creation of the dataset closely follows the methodology of Lopez-Lira and Tang (2023).

First, historical price data in the period between October 2021 and December 2023 are acquired from the CRSP database. A source mapping file from RavenPack is used to get all CUSIPs available in the RavenPack database. For this purpose, the source mapping file is filtered for `ENTITY_TYPE == COMP` and `DATA_TYPE == CUSIP`. The last digit in the DATA_TYPE column is removed because this CUSIP digit is a checksum that is not used in the CRSP database. Duplicates from the DATA_TYPE column are dropped. Next, historical stock prices are downloaded from the CRSP database. The time series data includes features for CUSIP, calendar date, company name, share code, adjusted open price, adjusted close price and market capitalization. Based on the adjusted open and close prices the daily open-to-close returns are calculated.

Next, all companies without share code 10 or 11 are dropped. After this step 4765 unique firms are left. The source mapping file is then merged with the time series on CUSIP to get a mapping from CUSIP to RP_ENTITY_ID. It is noticed that some RP_ENTITY_IDs map to more than one CUSIP. The example of Berkshire Hathaway shows that this is the case for companies that have A and B shares. The first instances of all duplicates are dropped, which should drop all A shares from the sample. This is done because B shares might be more easily tradeable in those cases. After this step 4711 unique firms are left.

Second, all company news headlines available from RavenPack in the period between October 2021 and December 2023 are downloaded. The start date is set to October 2021 because training data cut-off for GPT-3.5-Turbo is September 2021 (OpenAI, 2024c). Identically to Lopez-Lira and Tang (2023), the following filtering steps are performed:

```
RELEVANCE == 100
EVENT_SIMILARITY_DAYS > 90
NEWS_TYPE in [PRESS-RELEASE, FULL-ARTICLE]
CATEGORY not in [stock-gain, stock-loss]
```

Next, all duplicate headlines are dropped. The remaining headlines are filtered for pairwise restricted Damerau-Levenshtein distance. In case of a distance below $0.6$, the first instance in the dataframe is kept.

Third, the datasets from steps 1 and 2 are merged. This means that a stock return of the historical stock return data from step 1 is mapped to every news headline from step 2. Market opening hours on the NYSEArca platform are 9:30 am to 4:00 pm Eastern Time (Intercontinental Exchange, 2024). In contrast to Lopez-Lira and Tang (2023), in this thesis only headlines published outside of trading hours are used. Additionally, the time cut-offs for mapping daily returns to news headlines are changed. In Lopez-Lira and Tang (2023), a stock is traded at the next market open at 9:30 am if the corresponding news headline is released after 4:00 pm of the last trading day and before 6:00 am of the current trading day. In this thesis, a stock is traded at the next market open at 9:30 am if the corresponding news headline is released after 3:45 pm of the last trading day and before 9:15 am of the current trading day. This is done because a 15-minute time period should be enough to process a new headline and execute a trade.

The choice to only use headlines published between 3:45 pm of the last trading day and before 9:15 am of the current trading day is made because the effect size is larger for those news headlines. This is most likely the case because for headlines published within trading hours, almost two trading days can elapse between news release and last trade of a stock. For instance, if a positive news headline is released on a Monday at 6:01 am, then in Lopez-Lira and Tang (2023)'s methodology the stock is bought at market close on Monday and sold on market close at Tuesday, which is at 4:00 pm one day after news release.

In the final pre-processing step, all instances are dropped for which the difference between news publication date and next trading day is larger than 7 days or the return on the next trading day is outside of the open interval $(-100\%, 300\%)$. Those instances are regarded to be caused by data quality issues.

Initial testing showed that filtering the resulting dataset for RavenPack's News Impact Projections (NIP) score is necessary to reduce the dataset size. According to RavenPack's user guide

(RavenPack, 2016), the NIP model is a machine learning model trained on price data to predict the impact a news headline has on the volatility of a firm's stock price over the following two hour period after news publication. The score is centered around zero, indicating no impact on volatility. The NIP has a range of $[-1, 1]$, meaning that $0$ indicates no impact on volatility, and higher values indicate higher impact. The dataset is finally filtered for `NIP >= 0.3`. For a more detailed explanation of why the dataset is filtered at this NIP score, see section (5.1). The final dataset includes 28004 news headlines from 3813 unique companies.

## 4.3. Classification Task

GPT-3.5-Turbo is used to classify each of the 28004 headlines as either positive, neutral or negative for the stock price of the company mentioned in the headline. For further processing, the string labels are mapped to $\{1, 0, -1\}$, where positive is mapped to $1$, neutral to $0$, and negative to $-1$. This setup follows the methodology of Lopez-Lira and Tang (2023). Earlier papers (Ding et al., 2015; dos Santos Pinheiro and Dras, 2017) only use a positive and a negative class. It has been argued that the neutral class is useful for handling class noise (Valdivia et al., 2018). Since conducting stock price prediction purely on single news headlines could contain a lot of noise, a neutral label might be especially useful.

Throughout this study, the model `gpt-3.5-turbo` is used via the OpenAI API. Equivalently to Lopez-Lira and Tang (2023), temperature is set to 0. Additionally, the nucleus sampling hyperparameter `top_p` is set to 1. For all inferences, 24 is used as a seed for model completions.

The classification task is given to the model via an instruction in the prompt. Initially, the same prompt used in Lopez-Lira and Tang (2023) was tested on a few headlines. However, for this prompt the model classified approximately 98% of all non-zero labels as positive. Potential reasons are discussed in section (5.1). In any case, the prompt by Lopez-Lira and Tang (2023) had to be adapted to obtain meaningful results. After concise initial testing, the first prompt that yielded positive trading strategy returns is used in section (5.1) to establish a baseline.

The labels of the individual news headlines are tested for their return predictability. In this thesis, a standard OLS regression of the following form is used:

$$r_{i,t+1} = \alpha + \beta * x_{i,t} + \epsilon_{i,t+1} \tag{4.1}$$

In this formulation, $r_{i,t+1}$ is the return of stock $i$ over a subsequent trading day $t + 1$ as discussed above. $\alpha$ denotes the predicted value for the neutral label, $\beta$ captures the effect for the classification label $x_{i,t}$, and $\epsilon_{i,t+1}$ is the error term. Note that Lopez-Lira and Tang (2023) use a linear regression with two-way fixed effects estimators for firm and date. Results for their regression setup are also reported. More details are laid out in section (5.1).

Lastly, it must be mentioned that the classification task is altered for optimization insofar as the task instruction varies in different prompt setups. More details are laid out in section (5.2).

## 4.4. Trading Strategies

The labels of the classification task described above are used to test different trading strategies. In line with Lopez-Lira and Tang (2023), three strategies are examined throughout the study: a short-only strategy, a long-only strategy and a self-financing long-short strategy.

After the classification task is completed, each headline has a sentiment label associated with it. For any day and firm, these labels are used to act on the stock of the firm for that day - buying if positive sentiment, short selling if negative sentiment, or doing nothing if neutral sentiment. Since for any day and firm it can be the case that more than one headline is included, the individual news headlines must be aggregated into a daily sentiment score.

The starting point is a dataframe of sentiment labels for any headline. It includes features for date, firm and the sentiment label for each news headline. First, for any date and company, the arithmetic mean of the sentiment labels is calculated. If the mean is greater than 0 it is rounded up to 1, if it is less than 0 it is rounded down to -1 and if the mean equals 0, then the firm is dropped from the dataset for this particular date. The rounded mean labels represent the daily sentiment labels for the firm and day. Second, the daily sentiment labels are divided into positive (long-only) and negative (short-only) labels. For both, the daily trading strategy return is the arithmetic mean of the daily returns of all headlines that have a positive or negative daily sentiment associated with them. Third, the arithmetic mean of the long-only and the short-only daily returns is calculated to obtain the daily return of a self-financing long-short strategy. An example of steps 1 to 3 is symbolized by the three arrows in figure (4.1).

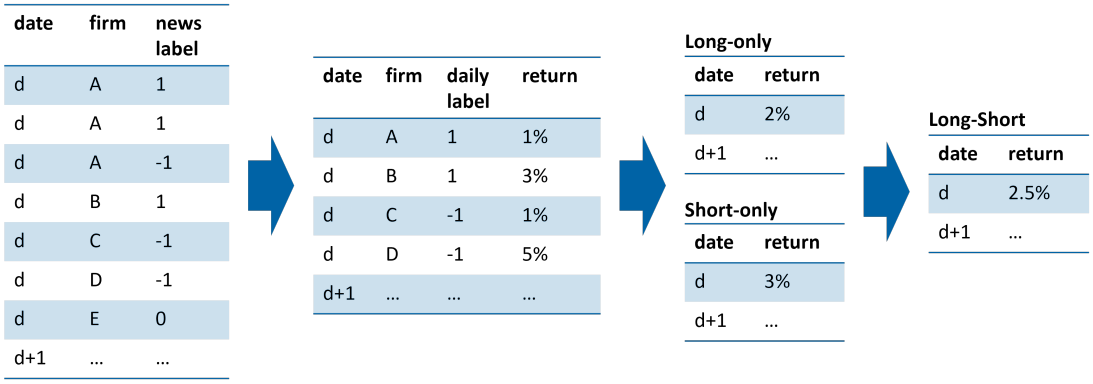


Figure 4.1.: Example of how labels of the classification task ("news label") are first converted into daily sentiment labels ("daily sentiment") and then into different trading strategies ("long-only", "short-only" or "long-short").

# 5. Results

This chapter includes three sections. In the first section, the results of Lopez-Lira and Tang (2023) are replicated to establish baseline results. In the second section, three optimization methods are tested to improve the baseline: Prompt engineering, few-shot learning and fine-tuning. In the third section, the best setup of the second section is applied on the test set to examine how much of the returns of a self-financing long-short strategy persists in different transaction cost scenarios.

## 5.1. Baseline

First, it is attempted to use the same prompt used in Lopez-Lira and Tang (2023) to establish baseline results for optimization. However, initial testing on 7000 instances revealed a heavily skewed label distribution towards neutral and positive labels. Out of these instances, 5260 were labelled as neutral, 1704 as positive, and only 36 as negative. Despite this imbalance, standard OLS regressions indicated that labels predicted next period returns on a statistically significant level. Additionally, a self-financing long-short strategy produced substantial positive returns. Nonetheless, the imbalance between positive and negative classes contributed to instability in regression and trading strategy results.

Two main reasons account for this disparity. First, the dataset in this thesis differs from the one used in Lopez-Lira and Tang (2023). Here, news headlines are directly taken from RavenPack, whereas Lopez-Lira and Tang (2023) webscraped headlines and then matched them with those available from RavenPack. This suggests that the instances for initial testing in this thesis must be a superset of the dataset from the original study. It is possible that the original study only webscraped news outlets containing less noisy data compared to the broader RavenPack dataset. While less noise could explain lower overall performance, it is not obvious why this would lead to such an extensive class imbalance. Therefore, the second explanation seems more plausible: different models are used in this thesis compared to the original study. Although both studies use GPT-3.5-Turbo via the OpenAI API, the specific model that is accessible under this brand name undergoes continual updates by OpenAI. For instance, the beta feature "system fingerprint" is used in this thesis. According to online documentation the fingerprint represents the backend configuration that the model runs with. Even throughout the implementation in this thesis this fingerprint changed, indicating that different GPT-3.5-Turbo models are effectively used for inference throughout this study. While these changes between model versions are likely minor in this case, reproducibility is crucial for research purposes. Thus, a main result of this thesis is that further research into large language models and stock price prediction should be conducted with open-source models. For instance, Llama models could be a viable alternative, even though infrastructure

for inference of large versions might be an obstacle.

After concise initial testing, the first prompt that produced positive trading strategy returns is adopted as the baseline. The prompt can be found in the appendix (A.1). From the perspective that the baseline should later be optimized, it is suboptimal that the original prompt by Lopez-Lira and Tang (2023) does not produce meaningful results, as the prompt eventually chosen as a baseline already incorporates some best practices mentioned in OpenAI's online documentation (OpenAI, 2024d). Specifically, the prompt asks the model to adopt a persona, as is also suggested in White et al. (2023). Additionally, the model is given the instruction which labels to return as a bullet point list. This aligns with White et al. (2023)'s recipe pattern which recommends providing instructions as a sequence of steps to achieve a goal. As Clavié et al. (2023) point out, there is no clear guidance on how to split the prompt content within the system message and the first user message. In the baseline setup, the system message includes both the act-as-a-persona instruction as well as the task description in bullet point form.

During the initial testing phase, it became clear that too many headlines pass the filtering criteria laid out in Lopez-Lira and Tang (2023). The resulting dataset is too large to be processed considering the budget constraints of this study. The search for solutions to this problem lead to two findings. First, captured effects are larger for headlines that are published outside of trading hours. This is most likely the case because for headlines published within trading hours almost two trading days can elapse between news release and last trade of a stock. For instance, if a positive news headline is released on a Monday at 6:01 am, then in Lopez-Lira and Tang (2023)' methodology the stock is bought at market close on Monday and sold on market close at Tuesday, which is at 4:00 pm one day after news release. Second, the dataset still contained a multitude of headlines with vacuous content such as "Firm X publishes annual report". The idea emerged to use the NIP score introduced in section (4.2) to filter for more impactful news headlines. On the one hand, this brings the dataset down to a manageable size in terms of cost. On the other hand, the NIP leverages complementary features for prediction. Whereas GPT-3.5-Turbo is purely trained on text data, the NIP model is trained on numerical data as well. However, it is not clear a priori at which NIP score to filter.

To find an appropriate NIP score for cut-off, headlines in the period between October 2021 and December 2022 are filtered with all steps described as in section (4.2) except for the NIP score filter at 0.3. The resulting dataset is then filtered for different NIP scores. All NIP scores are lower cut-off values, meaning that each headline that has at least this NIP score is included in each dataset slice. For each dataset slice, regressions are conducted and self-financing long-short strategies are calculated. Coefficients for the regression as well as Sharpe ratio and cumulative returns are reported in table (5.1). For all NIP score cut-offs examined, the GPT label coefficient $\beta$ is statistically significant. It ranges between 0.18 and 0.55% which is a similar magnitude as reported in Lopez-Lira and Tang (2023). Cumulative returns before transaction costs are also substantial for all NIP scores examined. Filtering for higher NIP scores produces higher regression coefficients and higher cumulative returns. This is evidence that machine learning models trained on numerical data can complement large language models in a stock price prediction task, if the model is used to filter headlines before the classification task. However, the results also exhibit increased instability for higher NIP scores in two ways. First,

the regression coefficients achieve statistical significance at a lower confidence level. Second, the volatility of the trading strategy increases more than cumulative returns, as indicated by the declining Sharpe ratios. This could be attributed to the small number of observations available for extreme NIP scores. For few observations, it becomes more likely that only one or two stocks are traded on a given day, which can substantially impact volatility. To balance dataset size and result stability, a NIP score cutoff of 0.3 is applied for subsequent experiments. For this NIP score, a label switch is associated with a 0.46% increase in next trading period stock returns and one dollar invested increases to 2.47 dollars at the end of 2022.

Table 5.1.: Regression and trading strategy results for different NIP scores. NIP is the cut-off value for this score, nobs the number of observations, $\alpha$ and $\beta$ the coefficients as explained in equation (4.1), Sharpe the Sharpe ratio of a self-financing long-short strategy and return is the cumulative return for such strategy.

| | | Regression | | Trading Strategy | |
|---|---|---|---|---|---|
| NIP | nobs | $\alpha$ | $\beta$ | Sharpe | return |
| $0.25$ | $21\,129$ | $0.0005$ | $0.0018^{**}$ | $3.11$ | $1.26$ |
| $0.30$ | $15\,052$ | $0.0005$ | $0.0023^{***}$ | $2.83$ | $1.47$ |
| $0.35$ | $10\,095$ | $-0.0001$ | $0.0025^{**}$ | $2.39$ | $1.48$ |
| $0.40$ | $7497$ | $0.0014$ | $0.0031^{**}$ | $2.39$ | $1.68$ |
| $0.45$ | $3697$ | $-0.0003$ | $0.0033^{**}$ | $2.45$ | $3.25$ |
| $0.50$ | $2692$ | $0.0010$ | $0.0035^{*}$ | $2.11$ | $2.31$ |
| $0.55$ | $1618$ | $0.0022$ | $0.0055^{**}$ | $2.24$ | $2.50$ |

$^{*}$ p $< 0.05$, $^{**}$ p $< 0.01$, $^{***}$ p $< 0.001$

Next, a regression and a long-short strategy are calculated for the period between January 2023 and December 2023. Results are presented in table (5.2). The effect sizes slightly decline for all relevant metrics in 2023, but the results remain fairly robust. The label coefficient decreases by 0.01%, and the Sharpe ratio by 0.08. Note that the first period is 3 months longer than the second. The annualized return for the first period equals $2.47^{\frac{12}{15}} - 1 = 1.06$ and is still 10% higher than the $0.96$ in the second period. The decrease in effect sizes could be attributed to the increased accessibility of large language models in 2023. If any investors began utilizing large language models for stock prediction tasks, then the captured effects might diminish. In contrast to the earlier period, the $\alpha$ coefficient also became negative at a statistically significant level. One explanation could be the different label distributions in the two periods. The ratio of positive to negative labels was 1.02 in the earlier period and 1.12 in the later period. It could be the case that more headlines with negative returns were labelled as neutral in the second period, leading to the negative intercept. In general, setups with a positive to negative label ratio close to 1 tended to perform better. Finally, it should be mentioned that the sum of instances in both periods in table (5.2) equals 27735, whereas section (4.2) states that the entire dataset has length 28004. This discrepancy arises from the mapping of GTP string output

to final label. For instance, for a given headline input GPT-3.5-Turbo could in rare instances output "Answer: Positive", instead of "positive". During testing the optimization setups, all 28004 GPT output strings could be mapped to their target labels, but in the baseline setup 269 are lost.

Table 5.2.: Regression and trading strategy results for different time periods. Period is the time period, nobs is the number of observations, $\alpha$ and $\beta$ the coefficients as explained in equation (4.1), Sharpe the Sharpe ratio of a self-financing long-short strategy and return is the cumulative return for such strategy.

| | | Regression | | Trading Strategy | |
|---|---|---|---|---|---|
| period | nobs | $\alpha$ | $\beta$ | Sharpe | return |
| 10/21-12/22 | 15 052 | $0.0005$ | $0.0023^{***}$ | $2.83$ | $1.47$ |
| 01/23-12/23 | 12 683 | $-0.0031^{***}$ | $0.0022^{**}$ | $2.75$ | $0.96$ |

$^{*}$ p $< 0.05$, $^{**}$ p $< 0.01$, $^{***}$ p $< 0.001$

As mentioned in section (4.3), Lopez-Lira and Tang (2023) use a linear regression with two-way fixed effects estimators for firm and date. For this setup, the regression coefficients and p-values for different NIP values depicted in table (5.1) are the following: (0.0020, 0.0006), (0.0027, 0.0002), (0.0035, 0.0002), (0.0042, 0.0004), (0.0032, 0.1041), (0.0015, 0.5796), (0.0074, 0.1623). Results are shown in ascending order of NIP scores, where (0.0020, 0.0006) corresponds to a cut-off value of 0.25, (0.0027, 0.0002) to 0.3, and so on. A similar tendency as observed in table (5.1) becomes more pronounced: Higher NIP score cut-offs tend to yield higher coefficients, but these coefficients are less statistically significant. For different time periods as depicted in table (5.2) the regression coefficients and p-values are (0.0027, 0.0002) and (0.0005, 0.4828). If a regression with a one-way fixed time effect is estimated for the second period, the coefficient is 0.0020 with a p-value of 0.0016. This suggests that stock returns are driven by firm-specific characteristics in the year 2023. It would be interesting to further investigate why return predictability disappears in 2023 when controlling for firm-specific effects. However, given that establishing baseline results is only the initial part of this thesis and the implementation scope of optimization methods is extensive, it is decided not to further investigate this question in this thesis.

In both time periods, the trading strategy returns based on GPT labels are substantial. Lopez-Lira and Tang (2023) explain this with underreaction to news and limits-to-arbitrage. To test if limits-to-arbitrage influence the results of this study, the regressions and trading strategies are calculated for different trading position and firm size. For trading position, the GPT label is split into two dummy variables: one for positive labels, which imply long trades, and one for negative labels, which imply short trades. Additionally, Sharpe ratios and cumulative returns for long-only and short only trading strategies are calculated. The results are shown in table (5.3). In this regression setup, only coefficients for short trades are non-zero on a statistically significant level. This holds true across both time periods. Furthermore, in 2023 the effect declines for long trades, whereas returns of a short-only strategy are higher both in cumulative

and risk-adjusted terms. This is consistent with the idea that overpricing is harder to arbitrage away than underpricing because short trades are either more costly or sometimes not possible due to legal restrictions. The performance decline of the long-only strategy in 2023 could be explained by wider adaption of large language models or other advanced NLP systems for stock price prediction.

Table 5.3.: Regression and trading strategy results for different trade positions. Period is the time period, $\beta$-long and $\beta$-short are dummy coefficients for the $\beta$ coefficient as explained in equation (4.1) and Sharpe and returns are the Sharpe ratio and cumulative returns for long-only and short-only strategies.

| period | Regression | | | Trading Strategy | | | |
|---|---|---|---|---|---|---|---|
| | $\alpha$ | $\beta$ (long) | $\beta$ (short) | Sharpe (long) | Sharpe (short) | return (long) | return (short) |
| 10/21-12/22 | $0.0009$ | $0.0018$ | $-0.0029^{*}$ | $1.32$ | $2.12$ | $0.88$ | $1.82$ |
| 01/23-12/23 | $-0.0020^{**}$ | $0.0006$ | $-0.0037^{**}$ | $0.19$ | $2.91$ | $0.06$ | $2.36$ |

$^{*}$ p $< 0.05$, $^{**}$ p $< 0.01$, $^{***}$ p $< 0.001$

To test the impact of firm size on return predictability and trading strategy returns, all instances of the dataset are split into either small or large cap. There is no universal definition of what constitutes small cap. Lopez-Lira and Tang (2023) define small cap as firms whose market capitalization is below the 10th percentile NYSE market capitalization distribution. In the dataset of this study, the 10th percentile of all daily sentiment labels are calculated for this purpose. Daily sentiment is defined as explained in section (4.4). The 10th percentile lies at 101 million USD for the period 10/21-12/22 and at 60 million USD for the period 01/23-12/23. To approximate the 10th percentile across both periods small cap is defined as firms whose market cap is less than 100 million USD. The results for regressions and trading strategies can be seen in table (5.4). Effects are larger for small cap than for large cap in both time periods. In the first period, the regression coefficient for GPT labels is approximately six times larger for small caps. For the regression with two-way fixed effects estimators, the regression coefficients and p-values are (0.0128, 0.0148) for small cap and (0.0021, 0.0028) for large cap. Lopez-Lira and Tang (2023) report coefficients of 0.00653 for small cap and 0.00148 for large cap. For the second time period, the trading strategy results decline for both market cap categories in 2023. For the two-way fixed effects, regression coefficients disappear for both market cap categories. If only time-fixed effects are accounted for, only coefficients for large caps disappear. The effects decline most for large caps in 2023. This is analogous to the results in table (5.3), where the effects for long trades decline most in 2023.

Overall, the results by Lopez-Lira and Tang (2023) can be replicated. Stock returns are predictable on a daily horizon with the help of GPT-3.5-Turbo. However, the effect sizes captured in this study are smaller than in Lopez-Lira and Tang (2023) and essential parts of the setup had to be adapted because closed-source datasets and models are used in the original study. Additional evidence from the year 2023 points in the same direction, even though stock return

Table 5.4.: Regression and trading strategy results for different firm sizes. Period is the time period, cap the market cap category, nobs the number of observations, $\alpha$ and $\beta$ the coefficients as explained in equation (4.1), Sharpe the Sharpe ratio of a self-financing long-short strategy and return is the cumulative return for such strategy.

| | | | Regression | | Trading Strategy | |
|---|---|---|---|---|---|---|
| period | cap | nobs | $\alpha$ | $\beta$ | Sharpe | return |
| 10/21-12/22 | small | 1708 | $-0.0105^{***}$ | $0.0085^{*}$ | 2.62 | 5.30 |
| | large | 13 344 | $0.0021^{***}$ | $0.0014^{*}$ | 2.31 | 1.08 |
| 01/23-12/23 | small | 1756 | $-0.0152^{***}$ | $0.0098^{**}$ | 2.72 | 4.71 |
| | large | 10 927 | $-0.0008$ | $0.0004$ | 1.18 | 0.32 |

$^{*}$ $p < 0.05$, $^{**}$ $p < 0.01$, $^{***}$ $p < 0.001$

predictability seems to be driven by firm-specific effects in this time period. Limits-to-arbitrage seem to play an important role in return predictability during both periods. Effect sizes are larger for small caps and for short trades. The role of underreaction to news is further investigated in section (5.3).

## 5.2. Optimization Methods

In this section, three optimization methods to improve the baseline from section (5.1) are examined: Prompt engineering, few-shot learning and fine-tuning. For this purpose, the dataset is split into training, validation and test sets. The training set spans the first 9 months, the validation set the next 8 months and the test set the last 10 months. Initially, it was planned that all slices should cover 9 months. An error in the split was only discovered after first prompt setups were used for inference. Since this error is only a minor problem from a data analytics perspective and GPT inference is costly, the uneven 9/8/10 split is used throughout this study. The train split includes 8985 instances, the validation split 8602 and the test split 10417.

### 5.2.1. Prompt Engineering

In this subsection, the performance of seven different prompt setups is compared to the baseline. The setups are named setup1 to setup7. The prompts for all setups can be found in the appendix (A.1). The Python package Jinja2 is used for template management.

Setup1 is arguably the most different from the baseline prompt. As suggested by Clavié et al. (2023), the model is given a name by which it can be referred to. The task instruction is adapted and the target labels are changed to `buy`, `neutral` and `sell` so that it is clear to the model that the class labels induce trades. In the OpenAI online documentation, it is suggested to use delimiters to clearly indicate distinct parts of the input (OpenAI, 2024d). Since simple quotation marks might not be clear delimiters, the delimiters for the target labels are changed to <. . .>.

Further following Clavié et al. (2023), a mock dialog between the user and model is inserted in the prompt. In contrast to the baseline prompt, the task instruction is not part of the system message. Key instructions are reinforced by repeating them and a zero-shot chain-of-thought answer is induced by asking the model to reason step by step. Sahoo et al. (2024) also report that chain-of-thought prompting has been shown to improve task performance in different setups. To stay in line with the best prompt in Clavié et al. (2023), the instruction which target labels to use is given to the model in running text and not in a bullet point list. In setup2, the target label names and delimiters for target labels are the same as in setup1. The task instruction is also given in the user message. Additionally, triple quotation marks are used to indicate the placeholder for the headline, time stamp and trading day. In setup3, the task instruction is changed such that the placeholder for the company name is not part of the system message. This prepares the setup for potential use in a few-shot learning scenario. As in setup2, the placeholder for the headline, timestamp and trading day is indicated by triple quotation marks. Apart from those adaptions, the prompt is equal to the baseline prompt. Setup4 is changed such that the target labels are again adapted to `buy`, `neutral` and `sell`. The trading period description in the user prompt is also slightly adapted. This is done to account for the fact that a stock is only bought at market open, if the daily sentiment is positive. Apart from those changes, setup4 is equal to setup3. In setup5, only the headline of a dataset instance is given, no timestamp and trading period is included. The task instruction is changed such that the problem is posed to the model as a pure sentiment classification task, without any mention of trading stocks. In setup6, labels are again changed to `buy`, `neutral` and `sell`. The task instruction used in setup3 is adapted. It is formulated more explicitly that `buy` and `sell` sentiments induce a trading strategy. Setup7 is changed such that the same task instruction as in setup3 is taken and appended to the beginning of the user message instead of the system message. The delimiters for the target labels are changed to <. . .>.

When searching for an optimal prompt, evaluation can be made on the union of training and validation sets. This is the case since no parameter updates are done during prompt engineering and thus no information can leak from the training to the validation set. The results are displayed in table (5.5) and sorted by Sharpe ratio. Time series plots of the corresponding trading strategy returns can be found in the appendix (A.2). The setups were created and evaluated on training and validation sets in the sequence of their naming. The first two prompts included most additional prompting techniques. Especially setup1 underperformed the baseline substantially. That is why from setup3 onwards the tactic was changed to only slightly adapt the baseline setup.

Setup3 performs best and substantially outperforms the baseline in terms of Sharpe ratio and cumulative returns. Setup5 outperforms the baseline substantially in terms of Sharpe ratio. It did not hurt performance to exclude the company name from the system prompt. Additionally, the prompts in both setups tend to be shorter. While in setup3 the task instruction only vaguely suggests that labels are used for a trading strategy, in setup5 the task instruction is purely given as a classification task with no mentioning of a trading strategy. In setups 1 and 6 the task instruction are explicitly formulated such that it is obvious that trading strategies are executed based on the classification labels. Both setups significantly underperform in terms of Sharpe ratio and cumulative returns. Additionally, the underperformance of setup1

Table 5.5.: Accuracy and trading strategy results for different prompt engineering setups. Dataset and prompt are the dataset split and prompt used for inference. All prompt setups can be found in the appendix (A.1). acc in % is the accuracy of all non-zero news labels in percent, Sharpe the Sharpe ratio of a self-financing long-short strategy and return is the cumulative return for such strategy.

| dataset | prompt | acc in % | Sharpe | return |
|---|---|---|---|---|
| train+val | Setup3 | 51.52 | 3.11 | 2.03 |
| | Setup5 | 51.48 | 3.10 | 1.71 |
| | Setup2 | 51.21 | 2.92 | 1.66 |
| | Baseline | 51.37 | 2.86 | 1.64 |
| | Setup4 | 51.09 | 2.53 | 2.28 |
| | Setup1 | 51.00 | 2.45 | 1.21 |
| | Setup6 | 51.09 | 2.34 | 1.48 |
| | Setup7 | 51.11 | 2.19 | 1.27 |
| test | Setup3 | 50.42 | 3.19 | 1.04 |
| | Baseline | 50.19 | 3.08 | 0.95 |

supports Xie et al. (2023) who find that chain-of-thought prompting tends to worsen results for a stock return prediction. In general, it seems that Sharpe ratio and cumulative returns decline in parallel. Setup4 is the only exception to this. This setup has the highest cumulative return but also a substantially lower Sharpe ratio than the baseline and the best performing prompts.

GPT-3.5-Turbo is used to conduct a classification task. In machine learning, it is common to optimize such a task for accuracy or similar metrics such as recall or precision. In the setup for this thesis, classification labels are used to implement a trading strategy. Arguably, the optimization efforts should focus on trading strategy performance and not classification task performance. Table (5.5) shows some tendency that accuracy increases with trading strategy performance. However, in the next subsection about few-shot learning evidence is presented that performance on trading strategy and classification task can diverge substantially.

The best setup on training and validation sets which is setup3 is evaluated on the test set. The performance improvements of this setup compared to the baseline are persistent in the test set. Inference on the test set only cost approximately 1 USD. The results suggest that prompt engineering is a cost effective and fairly straightforward approach to improve GPT's performance.

To test whether performance improvements shown in table (5.5) are statistically significant, the difference between the label of the new setup and the baseline classification label is added as a second independent variable to the regression setup displayed in equation (4.1). For the test set, this coefficient is slightly positive if the dependent variable is subsequent trading day return sign and slightly negative if the dependent variable is subsequent trading day return. However, in both cases the p-values are not statistically significant. This suggests that

while performance improvements in terms of trading strategy are substantial, they are not statistically robust in a linear regression.

Prompt engineering is essentially a hyperparameter optimization problem with a considerable number of possible setups. Since this is among the first attempts to study the impact of large language model optimization methods on stock price prediction, the prompts tested are each different in several aspects. This gives a first indication of which prompting techniques can be beneficial or detrimental for a stock return prediction task. The experiments point to further investigation of shorter and concise prompts with a task instruction that emphasizes the classification task instead of the trading strategy. Future studies can take this as a starting point. However, an ablation study approach, where only one single prompt component is removed or added for each setup, can be used in future studies to give a clearer result on which individual prompting techniques add performance.

### 5.2.2. Few-Shot Learning

In this subsection, the performance of six different few-shot learning setups is compared to the baseline. The best prompt from subsection (5.2.1) is setup3. The prompt of this setup is taken and extended by few-shot examples. Few-shot learning is a variant of in-context-learning as described in Brown et al. (2020). In few-shot learning, the prompt contains the task description and more than one input-output demonstration.

It is well known that few-shot learning can enhance performance in a classification task, but it is not a priori obvious which few-shot examples are the best for a given task and what ordering of the examples should be used in the prompt. Liu et al. (2021) propose a k-nearest-neighbor (kNN) based solution for the selection problem, where samples are selected based on embedding distance to the instance that is being labelled. The authors hypothesize that the best ordering could be descending such that the closest instance in terms of distance is also closest to the instance in the prompt that is being labelled. In this subsection, kNN based few-shot learning with a descending few-shot example ordering is used. First, the news headlines of all dataset instances are embedded to a dimension of 3072 by OpenAI's embedding model `text-embedding-3-large`. Second, for every validation (test) headline the k nearest headlines are retrieved from the training set based on cosine similarity. Third, the k retrieved headlines together with their target labels are appended to the prompt between task instruction and validation (test) headline.

In this setting, it is not clear which news labels should be mapped to the `neutral` label, as this class is only supposed to filter out noise and does not technically exist as a target class. Nevertheless, different percentage thresholds are tested to map news headlines to the `neutral` label. For a given threshold, all headlines associated with a return whose absolute value is lower than the threshold are mapped to the `neutral` label. All different few-show learning setups are evaluated on the validation set. Results are displayed in table (5.6) and sorted by Sharpe ratio. Time series plots of the corresponding trading strategy returns can be found in the appendix (A.2).

The first two setups evaluated used a threshold of 1%. It was noticed that a higher k value of 5 lead to a worse Sharpe ratio. This could be the case because the training set might be

Table 5.6.: Accuracy and trading strategy results for different few-shot learning setups. Dataset and prompt are the dataset split and prompt setups used. k is the number of few shot examples and thr in % is the threshold used for the neutral label. acc in % is the accuracy of all non-zero news labels in percent, Sharpe the Sharpe ratio of a self-financing long-short strategy and return is the cumulative return for such strategy.

| dataset | prompt | k | thr in % | acc in % | Sharpe | return |
|---|---|---|---|---|---|---|
| val | Baseline | - | - | 51.98 | 3.97 | 0.99 |
| | Setup3 | 2 | 5 | 52.30 | 3.49 | 0.75 |
| | | 3 | 5 | 52.60 | 3.43 | 0.87 |
| | | 1 | 5 | 52.46 | 3.41 | 0.69 |
| | | 3 | 1 | 52.29 | 3.13 | 0.69 |
| | | 5 | 1 | 52.32 | 2.81 | 0.71 |
| | | 3 | 0 | 52.72 | 2.63 | 0.60 |
| test | Baseline | - | - | 50.19 | 3.09 | 0.95 |
| | Setup3 | 2 | 5 | 49.90 | 1.43 | 0.38 |

too small, meaning that for a given validation headline, there might be on average less than 5 semantically relevant headlines in the training set. Liu et al. (2021) find that increasing the training set size can substantially increase performance on a task. Upon manual inspection, it is noticed that the k nearest headlines are mostly either news about the same company, a company from a similar industry or the headlines follows the same wording. Liu et al. (2021) also find that maximum results for a tested task were already achieved with 10 to 20 nearest neighbors. In this thesis, higher k values than 5 are not tested due to cost reasons. With low k values of 3 or 5, the cost is already two to three times higher than inference in the baseline setup.

The best setup on the validation set in terms of Sharpe ratio uses k$= 2$ and a threshold of 5%. However, this setup substantially underperforms the baseline in terms of Sharpe ratio. The worst performing setup in terms of Sharpe ratio uses k$= 3$ and a threshold of 0%, effectively meaning that no `neutral` labels are used in the few-shot examples of this setup. This is further evidence that a `neutral` label is useful in this task, most likely to filter out noise. An unexpected finding is the divergence of accuracy and trading strategy performance. While all few-shot learning setups have a higher accuracy, they all have lower Sharpe ratios and cumulative returns. In subsection (5.2.1), a small accuracy increase of 0.15% is associated with a 0.25 increase in Sharpe ratio for setup3. In few-shot learning, one setup increases accuracy by 0.74%, but is the overall worst setup in terms of Sharpe ratio with a decrease of 1.34. This indicates that the classification task should be directly optimized for Sharpe ratio.

The best setup is evaluated on the test set. The Sharpe ratio is substantially worse than the Sharpe ratio of the baseline. In the test set, the accuracy improvements are not persistent. A

possible explanation for the declining performances of trading strategy and classification task could be that the training set is not recent enough. This explanation is supported by the time series plot of the trading strategy (A.12). The plot shows that the few-shot setup especially diverges from the baseline towards the end of the test set time series.

To test whether performance differences shown in table (5.6) are statistically significant, the difference between the label of the new setup and the baseline classification label is added as an independent variable to the regression setup displayed in equation (4.1). For the test set, this coefficient is -0.00046 if the dependent variable is subsequent trading day return sign and -0.00342 if the dependent variable is subsequent trading day return. Only in the case of trading returns the coefficient is significant at the 1% level. This means that labels of the new setup that disagree with labels of the baseline setup have a negative impact on return predictability. Considering the underperformance of the few-shot learning setups in terms of Sharpe ratio, this finding is not surprising.

In general, few-shot learning can be considered for performance improvements as indicated by the accuracy improvements. However, future research needs to investigate why performance divergence between classification task and trading strategy can occur and how performance improvements of the classification task can be translated into performance improvements of the trading strategy.

### 5.2.3. Fine-Tuning

In this subsection, the performance of two different fine-tuning setups is compared to the baseline. The best prompt from subsection (5.2.1) is setup3. This prompt is used to fine-tune GPT-3.5-Turbo on instances of the training set. The training set contains 8985 instances. With prompt setup3, the total training set contains the prompt template and the news headline for each instance, which in total amounts to approximately 2 million tokens. Since those 2 million tokens must be iterated through for every epoch, the total number of training tokens increases to 4 or 6 million. Additionally, inference of a fine-tuned model is six times more expensive than for a standard GPT-3.5-Turbo model. Fine-tuning cost and higher inference cost lead to a total cost of approximately 40 USD per training on the training set and inference on the validation set. This cost made hyperparemter search and the testing of different setups impossible. To still conduct a first test on how fine-tuning impacts the performance on this stock return prediction task, two setups are tested. For both setups the default parameters of the OpenAI API are used for batch size, learning rate multiplier and number of epochs.

In one setup, the model is fine-tuned on the full training set. To map any instances to the target label `neutral` a threshold must be assumed as explained in subsection (5.2.2). In this setup, a threshold of 1% is used.

In the other setup, the model is fine-tuned only on a small subset of the training set. This subset includes 114 instance of which each of the target classes `positive`, `neutral` and `negative` is represented by 38 instances. The instances are hand selected based on manual inspection. During manual inspection instances are found, where the GPT label in the baseline setup was `positive` or `negative`, but the sentiment upon inspection is mixed. 38 of such instances are selected. For example, `Humana Beats Profit Expectations But Misses`

`On Revenue` has an associated subsequent trading return of 5.3%, meaning that the target label is `positive`. The baseline label is `negative`, but the sentiment is mixed, so purely from the standpoint of the sentiment classification task, the correct label should be `neutral`. To get a balanced distribution in the training set, 38 instances are selected that have a positive subsequent trading return and a positive news sentiment based on manual inspection. 38 instances are selected that have a negative subsequent trading return and a negative news sentiment based on manual inspection. According to OpenAIs online documentation, fine-tuning should already be feasible with a small training set size of around 100 instances (OpenAI, 2024a).

The fine-tuning setup with only 114 instances is initially tested because of cost constraints. However, the two different setups address a question to which an answer is not obvious a priori: Should the model be trained on observed subsequent trading returns or on somewhat subjective news sentiment?

The results are shown in table (5.7) and sorted by Sharpe ratio. Time series plots of the corresponding trading strategy returns can be found in the appendix (A.2).

Table 5.7.: Accuracy and trading strategy results for different fine-tuning setups. Dataset and prompt are the dataset split and prompt setups used. train part is the part of the training set used for learning, thr in % is the threshold used for the neutral label and epochs the number of epochs trained. acc in % is the accuracy of all non-zero news labels in percent, Sharpe the Sharpe ratio of a self-financing long-short strategy and return is the cumulative return for such strategy.

| dataset | prompt | train part | thr in % | epochs | acc in % | Sharpe | return |
|---|---|---|---|---|---|---|---|
| val | Baseline | - | - | - | 51.98 | 3.97 | 0.99 |
| | Setup3 | partial | 1 | 3 | 51.84 | 3.39 | 0.75 |
| | | partial | 1 | 1 | 51.83 | 3.33 | 0.74 |
| | | full | 1 | 2 | 50.37 | 2.79 | 0.69 |
| test | Setup3 | partial | 1 | 3 | 50.08 | 3.12 | 0.90 |
| | Baseline | - | - | - | 50.19 | 3.09 | 0.95 |

The best setup uses the smaller training set. This setup substantially underperforms the baseline in terms of Sharpe ratio and cumulative returns. To see whether the model is overfitting with 3 training epochs, the result for 1 training epoch is also presented in the table. The results for 1 training epoch and 3 training epochs are almost identical, indicating that most learning happens during the first epoch. The worst setup is achieved, if the model is trained on the full training set. Two reasons could account for this disparity. First, hyperparameters chosen by default could be suboptimal for training on a stock return predition task. Hyperparameter searching was not possible due to budget constraints. Second, the training dataset contains too much noise to be effectively used for training. For instance, it could easily be the case that for a company a positive and a negative headline is published on the same day. If the

negative news outweigh positive news, then the target label of the positive headline might be `negative` in terms of subsequent stock returns, even though the correct sentiment label is `positive`. Even though it is not inconceivable that within a large training set, useful patterns could be learned if the model is trained on observed subsequent trading returns, it might be a better approach to purely assign the target label based on news sentiment. However, also the model trained on hand selected instances based on sentiment underperformed the baseline substantially. Consequently, conclusions from fine-tuning results are very tentative in this thesis. Future research could address the question, if fine-tuning should be based on observed subsequent trading returns or news sentiment. It must be considered that compared to prompt engineering and few-shot learning, much higher time and resource efforts are necessary to construct an appropriate training set and pay for training and inference.

The best setup on the validation set is evaluated on the test set. On the test set, this setup even slightly outperforms the baseline in terms of Sharpe ratio. However, it underperforms in terms of accuracy and cumulative returns.

To test whether performance differences shown in table (5.7) are statistically significant, the difference between the label of the new setup and the baseline classification label is added as an independent variable to the regression setup displayed in equation (4.1). For the test set, this coefficient is slightly positive if the dependent variable is subsequent trading day return sign or subsequent trading day return. In both cases, the coefficient is not statistically significant.

## 5.3. Returns after Transaction Costs

In this section, the impact of transaction costs on a self-financing long-short strategy is investigated. It is argued in section (5.1) that limits-to-arbitrage plays an important role to explain the returns of trading strategies based on the sentiment labels. Lopez-Lira and Tang (2023) find similar effects regarding limits-to-arbitrage. However, they also argue that underreaction to news can explain trading strategy returns. To check whether underreaction to news impacts trading strategy returns, it is tested if cumulative returns disappear after transaction costs. Unfortunately, the dataset does not include bid and ask prices. To still tackle this question, different transaction cost scenarios are assumed for small, mid and large cap firms. For every news headline, the market cap of the corresponding firm at publication date is taken to categorize the firm as small, mid or large cap. In line with section (5.1), small cap is defined as a market cap below 100 million USD. Mid cap is defined as greater than 100 million USD and smaller than 1 billion USD. Sharpe ratios and cumulative returns are calculated based on self-financing long-short strategies for four different scenarios. For each scenario and market cap category, transaction costs for a round trip trade (buy and sell) are assumed.

The different scenarios are calculated based on the results of setup3 on the test set. For more details on this setup see subsection (5.2.1). The results are displayed in table (5.8). A time series plot of the corresponding trading strategy returns can be found in the appendix (A.2).

In all three scenarios with non-zero transaction costs, a substantial part of cumulative returns disappears. This should not be surprising, since the trading strategy based on the

Table 5.8.: Trading strategy results for different transaction costs scenarios. Cost setup is the cost setup used as explained in this chapter, small, mid and large are the transaction costs assumptions for a round trip trade in percent. Sharpe is the Sharpe ratio of a self-financing long-short strategy and return is the cumulative return for such strategy.

| cost setup | small | mid | large | Sharpe | return |
|---|---|---|---|---|---|
| none | 0 | 0 | 0 | 3.19 | 1.04 |
| low | 0.50 | 0.20 | 0.10 | 1.45 | 0.39 |
| medium | 0.75 | 0.25 | 0.15 | 0.68 | 0.18 |
| high | 1.00 | 0.50 | 0.20 | −0.53 | −0.10 |

classification labels is very trading intensive. It can be concluded that most of the trading strategy returns are attributable to limits-to-arbitrage effects. If any, only a small part of trading strategy returns can be attributed to underreaction to news. Still, this conclusion should be tentative with the methods used in this section, as the possibility to short stock and transaction costs can greatly vary among investors. With the effect size captured in this study, it is clear that individual private investors cannot profitably exploit return predictability of GPT-3.5-Turbo, as their possibility to short stocks is limited and transaction costs tend to be higher. However, for institutional investors it could be possible to exploit some of the returns after transaction costs. Based on historical data, Frazzini et al. (2018) estimate average transaction costs per trade of a large institutional investor to be around 0.1% per trade for large cap and 0.2% per small cap, meaning round trip trades would cost 0.2% and 0.4%. Since they use different methodologies to categorize firms as small or large cap, their results are not directly comparable. Still, it suggests that it could be possible for large institutional investors to achieve an outcome close to the low or medium cost scenarios. Additionally, such institutions will have more resources available to implement more advanced optimization methods such as fine-tuning.

# 6. Conclusion

In this study, the potential of GPT-3.5-Turbo in predicting stock returns using sentiment analysis of news headlines is investigated. In the first part of the study, the results by Lopez-Lira and Tang (2023) are replicated. It can be concluded that stock returns are predictable on a daily horizon with the help of large language models such as GPT-3.5-Turbo. However, the effect sizes captured in this study are smaller than in the original study. Additionally, essential parts of the setup in the original study had to be adapted to achieve the reported results. For future research, it would be useful to use a standardized dataset and open-source models to achieve reproducibility of results across different studies using the exact same prompt.

Apart from reproducing Lopez-Lira and Tang (2023)'s results, this study finds strong evidence that a machine learning model trained on numerical data can complement large language models in the return prediction task of this study, if this model is used to filter news headlines before the sentiment classification task. Additionally, return predictability persists outside the time period examined in Lopez-Lira and Tang (2023) and still exists in 2023, albeit with smaller effect size than in the period between October 2021 and December 2022. Over the entire time period of October 2021 to December 2023, limits-to-arbitrage seem to play an important role in stock return predictability. Predictability is larger for small caps and for short trades. With the effect sizes captured in this study, most of the returns of a self-financing long-short strategy disappear after transaction costs, indicating that the remaining returns attributable to underreaction to news are at best small. For large institutional investors, positive returns could still persist after transaction cost. However, no definitive answer is possible with the dataset used in this study.

In the second part of the study, three different optimization methods for large language models are tested: Prompt engineering, few-shot learning and fine-tuning. Prompt engineering is a cost effective and fairly straightforward approach to improve performance on the prediction task of this study. Shorter prompts with task instructions emphasizing the classification task seem to work best. Few-shot learning results show that performance between the classification task, measured in accuracy, and trading strategy, measured in Sharpe ratio, can diverge. This suggests that the classification task should be directly optimized for Sharpe ratio. Fine-tuning underperformed compared to the baseline setup on the validation set and showed similar performance on the test set. Hyperparameter search and testing for various other design choices was severely limited by budget constraints. Based on this study, no definitive conclusions on the merits of fine-tuning for performance improvements should be drawn.

Due to the novelty of this research field, the results of this study point to several interesting directions for future research. First and foremost, it would be useful to create and publish a standardized dataset for this stock return prediction task and demonstrate return predictability with an open-source language model such as Llama. It would also be interesting to calculate trading strategy returns based on intraday data, as appropriate timestamps for news headlines

are available in RavenPack's data and effect sizes are expected to be larger. While return predictability persists in 2023 in the regression setup of equation (4.1) and in a regression with time-fixed effects, the label coefficient is statistically insignificant for a two-way fixed effects regression. It would be interesting to further investigate, why return predictability of GPT-3.5-Turbo seems to be driven by firm-specific effects in 2023. Furthermore, it would be worthwhile to further investigate prompt engineering, few-shot learning and fine-tuning methods. For prompt engineering, an ablation study approach would be sensible, in which only one prompt component in the task instruction is changed per setup. For few-shot prompting, further investigation of the performance divergence in terms of accuracy and Sharpe ratio would be beneficial. Starting at a study budget of 500 to 1000 USD, it would be feasible and useful to further investigate fine-tuning of GPT-3.5-Turbo. For fine-tuning, it would be valuable to investigate whether training should be conducted based on observed subsequent stock returns or on news sentiment.

In short, this thesis makes two novel contributions to the research topic of stock price prediction with large language models. First, it is among the first to broadly confirm the results by Lopez-Lira and Tang (2023), who demonstrate that GPT-3.5-Turbo can predict stock returns on a daily horizon. Second, it demonstrates the viability of different optimization methods for large language models in the stock return prediction setup of this thesis and points to future research questions in the areas of prompt engineering, few-shot learning and fine-tuning.

# Bibliography


Ackley, D. H., G. E. Hinton, and T. J. Sejnowski (1985). A learning algorithm for boltzmann machines. *Cognitive science 9*(1), 147–169.

Ba, J. L., J. R. Kiros, and G. E. Hinton (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bahdanau, D., K. Cho, and Y. Bengio (2016). Neural machine translation by jointly learning to align and translate.

Bahl, L. R., F. Jelinek, and R. L. Mercer (1975). Design of a linguistic statistical decoder for the recognition of continuous speech. *IEEE Transactions on Information Theory 21*(3), 250–256.

Bahl, L. R., F. Jelinek, and R. L. Mercer (1983). A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5*(2), 179–190.

Ball, R. and P. Brown (1968). Empirical evaluation of accounting income numbers. *Journal of Accounting Research 6*, 159–178.

Bengio, Y., R. Ducharme, P. Vincent, and C. Jauvin (2003). A neural probabilistic language model. *Journal of Machine Learning Research 3*, 1137–1155.

Bernard, V. L. and J. K. Thomas (1989). Post-earnings-announcement drift: delayed price response or risk premium? *Journal of Accounting research 27*, 1–36.

Blitzstein, J. and J. Hwang (2019). *Introduction to Probability, Second Edition*. Chapman & Hall/CRC Texts in Statistical Science. CRC Press.

Brown, P., P. DeSouza, R. Mercer, V. Della Pietra, and J. Lai (1992). Class-based n-gram models of natural language. *Comput. Linguist* (1950).

Brown, P. F., S. A. Della Pietra, V. J. Della Pietra, J. C. Lai, and R. L. Mercer (1992). An estimate of an upper bound for the entropy of english. *Computational Linguistics 18*(1), 31–40.

Brown, T., B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei (2020). Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin (Eds.), *Advances in Neural Information Processing Systems*, Volume 33, pp. 1877–1901. Curran Associates, Inc.

Calomiris, C. W. and H. Mamaysky (2019). How news and its context drive risk and returns around the world. *Journal of Financial Economics 133*(2), 299–336.

Chan, L. K., N. Jegadeesh, and J. Lakonishok (1996). Momentum strategies. *The Journal of finance 51*(5), 1681–1713.

Clark, K., U. Khandelwal, O. Levy, and C. D. Manning (2019). What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

Clavié, B., A. Ciceu, F. Naylor, G. Soulié, and T. Brightwell (2023). Large language models in the workplace: A case study on prompt engineering for job type classification. In *International Conference on Applications of Natural Language to Information Systems*, pp. 3–17. Springer.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Ding, X., Y. Zhang, T. Liu, and J. Duan (2014). Using structured events to predict stock price movement: An empirical investigation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1415–1425.

Ding, X., Y. Zhang, T. Liu, and J. Duan (2015). Deep learning for event-driven stock prediction. In *Twenty-fourth international joint conference on artificial intelligence*.

dos Santos Pinheiro, L. and M. Dras (2017). Stock market prediction with deep learning: A character-based neural language model for event-based trading. In *Proceedings of the Australasian Language Technology Association Workshop 2017*, pp. 6–15.

Frazzini, A., R. Israel, and T. J. Moskowitz (2018). Trading costs. *Available at SSRN 3229719*.

Garcia, D. (2013). Sentiment during recessions. *The journal of finance 68*(3), 1267–1300.

Geva, M., R. Schuster, J. Berant, and O. Levy (2020). Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*.

He, K., X. Zhang, S. Ren, and J. Sun (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hendrycks, D. and K. Gimpel (2016). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Hinton, G. E. (1986). Learning distributed represent ations of concepts. *Proceedings of 8th meeting of the Cognitive Scienve Society*, 1–12.

Hochreiter, S. and J. Schmidhuber (1997). Long short-term memory. *Neural computation 9*(8), 1735–1780.

Holtzman, A., J. Buys, L. Du, M. Forbes, and Y. Choi (2019). The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Intercontinental Exchange (2024). Nyse trading hours. `https://www.nyse.com/markets/nyse-arca/market-info`. Accessed on 2024-06-12.

Jiang, H., S. Z. Li, and H. Wang (2021). Pervasive underreaction: Evidence from high-frequency data. *Journal of Financial Economics 141*(2), 573–599.

Lin, S., J. Hilton, and O. Evans (2022). Teaching models to express their uncertainty in words.

Liu, J., D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen (2021). What makes good in-context examples for gpt-$3$? *arXiv preprint arXiv:2101.06804*.

Liu, P., W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys 55*(9), 1–35.

Lopez-Lira, A. and Y. Tang (2023). Can chatgpt forecast stock price movements? return predictability and large language models.

Loughran, T. and B. McDonald (2011). When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *The Journal of finance 66*(1), 35–65.

Meta (2023). Meta llama. `https://github.com/meta-llama/llama/`.

Mikolov, T., M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur (2010). Recurrent neural network based language model. *Interspeech 2010*, 1045–1048.

Mikolov, T., W.-t. Yih, and G. Zweig (2013). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pp. 746–751.

OpenAI, :, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell, A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutsch, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges, C. Gibson, V. Goel, T. Gogineni, G. Goh, R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo, C. Hallacy, J. Han, J. Harris, Y. He, M. Heaton, J. Heidecke, C. Hesse, A. Hickey, W. Hickey, P. Hoeschele, B. Houghton, K. Hsu, S. Hu, X. Hu, J. Huizinga, S. Jain, S. Jain, J. Jang, A. Jiang, R. Jiang, H. Jin, D. Jin, S. Jomoto, B. Jonn, H. Jun, T. Kaftan, Łukasz Kaiser, A. Kamali, I. Kanitscheider, N. S. Keskar, T. Khan, L. Kilpatrick, J. W. Kim, C. Kim, Y. Kim, J. H. Kirchner, J. Kiros, M. Knight, D. Kokotajlo, Łukasz Kondraciuk, A. Kondrich, A. Konstantinidis, K. Kosic, G. Krueger, V. Kuo, M. Lampe, I. Lan, T. Lee, J. Leike, J. Leung, D. Levy, C. M. Li, R. Lim, M. Lin, S. Lin, M. Litwin, T. Lopez, R. Lowe, P. Lue, A. Makanju, K. Malfacini, S. Manning, T. Markov,

Y. Markovski, B. Martin, K. Mayer, A. Mayne, B. McGrew, S. M. McKinney, C. McLeavey, P. McMillan, J. McNeil, D. Medina, A. Mehta, J. Menick, L. Metz, A. Mishchenko, P. Mishkin, V. Monaco, E. Morikawa, D. Mossing, T. Mu, M. Murati, O. Murk, D. Mély, A. Nair, R. Nakano, R. Nayak, A. Neelakantan, R. Ngo, H. Noh, L. Ouyang, C. O'Keefe, J. Pachocki, A. Paino, J. Palermo, A. Pantuliano, G. Parascandolo, J. Parish, E. Parparita, A. Passos, M. Pavlov, A. Peng, A. Perelman, F. de Avila Belbute Peres, M. Petrov, H. P. de Oliveira Pinto, Michael, Pokorny, M. Pokrass, V. H. Pong, T. Powell, A. Power, B. Power, E. Proehl, R. Puri, A. Radford, J. Rae, A. Ramesh, C. Raymond, F. Real, K. Rimbach, C. Ross, B. Rotsted, H. Roussez, N. Ryder, M. Saltarelli, T. Sanders, S. Santurkar, G. Sastry, H. Schmidt, D. Schnurr, J. Schulman, D. Selsam, K. Sheppard, T. Sherbakov, J. Shieh, S. Shoker, P. Shyam, S. Sidor, E. Sigler, M. Simens, J. Sitkin, K. Slama, I. Sohl, B. Sokolowsky, Y. Song, N. Staudacher, F. P. Such, N. Summers, I. Sutskever, J. Tang, N. Tezak, M. B. Thompson, P. Tillet, A. Tootoonchian, E. Tseng, P. Tuggle, N. Turley, J. Tworek, J. F. C. Uribe, A. Vallone, A. Vijayvergiya, C. Voss, C. Wainwright, J. J. Wang, A. Wang, B. Wang, J. Ward, J. Wei, C. Weinmann, A. Welihinda, P. Welinder, J. Weng, L. Weng, M. Wiethoff, D. Willner, C. Winter, S. Wolrich, H. Wong, L. Workman, S. Wu, J. Wu, M. Wu, K. Xiao, T. Xu, S. Yoo, K. Yu, Q. Yuan, W. Zaremba, R. Zellers, C. Zhang, M. Zhang, S. Zhao, T. Zheng, J. Zhuang, W. Zhuk, and B. Zoph (2024). Gpt-4 technical report.

OpenAI (2020). Openai api. `https://openai.com/blog/openai-api`. Accessed on 2024-03-26.

OpenAI (2022). Introducing chatgpt. `https://openai.com/blog/chatgpt`. Accessed on 2024-03-26.

OpenAI (2023a). Introducing apis for gpt-3.5 turbo and whisper. `https://openai.com/index/introducing-chatgpt-and-whisper-apis/`. Accessed on 2024-06-07.

OpenAI (2023b). New models and developer products announced at devday. `https://openai.com/index/new-models-and-developer-products-announced-at-devday/`. Accessed on 2024-06-07.

OpenAI (2023c). Openai python repository github. `https://github.com/openai/openai-python/blob/release-v0.28.0/chatml.md`. Accessed on 2024-03-26.

OpenAI (2024a). Fine-tuning. `https://platform.openai.com/docs/guides/fine-tuning/fine-tuning`. Accessed on 2024-06-11.

OpenAI (2024b). Hello gpt-4o. `https://openai.com/index/hello-gpt-4o/`. Accessed on 2024-06-07.

OpenAI (2024c). Model documentation. `https://platform.openai.com/docs/models/gpt-3-5-turbo`. Accessed on 2024-06-07.

OpenAI (2024d). Prompt engineering. `https://platform.openai.com/docs/guides/prompt-engineering`. Accessed on 2024-06-11.

OpenAI (2024e). Tokenizer - openai platform. `https://platform.openai.com/tokenizer`. Accessed on 2024-03-22.

Ouyang, L., J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. (2022). Training language models to follow instructions with human feedback. *Advances in neural information processing systems 35*, 27730–27744.

Radford, A., K. Narasimhan, T. Salimans, I. Sutskever, et al. (2018a). Gpt-1 repository. `https://github.com/openai/finetune-transformer-lm`.

Radford, A., K. Narasimhan, T. Salimans, I. Sutskever, et al. (2018b). Improving language understanding by generative pre-training.

Radford, A., J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. (2019a). Gpt-2 repository. `https://github.com/openai/gpt-2`.

Radford, A., J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. (2019b). Language models are unsupervised multitask learners. *OpenAI blog 1*(8), 9.

RavenPack (2016). *RPNA 4: RavenPack News Analytics - User Guide and Service Overview for WRDS Users*. RavenPack. Version 4.0, Last Updated: June 10, 2016.

Sahoo, P., A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha (2024). A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*.

Sennrich, R., B. Haddow, and A. Birch (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Sundermeyer, M., R. Schlüter, and H. Ney (2012). Lstm neural networks for language modeling. In *Interspeech*, Volume 2012, pp. 194–197.

Sutskever, I., O. Vinyals, and Q. V. Le (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems 27*.

Tetlock, P. C. (2007). Giving content to investor sentiment: The role of media in the stock market. *The Journal of Finance 62*(3), 1139–1168.

Tetlock, P. C. (2011). All the news that's fit to reprint: Do investors react to stale information? *The Review of Financial Studies 24*(5), 1481–1512.

Tetlock, P. C., M. Saar-Tsechansky, and S. Macskassy (2008). More than words: Quantifying language to measure firms' fundamentals. *The journal of finance 63*(3), 1437–1467.

Touvron, H., L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Valdivia, A., M. V. Luzón, E. Cambria, and F. Herrera (2018). Consensus vote models for detecting and filtering neutrality in sentiment analysis. *Information Fusion 44*, 126–135.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, Volume 30. Curran Associates, Inc.

Vig, J. (2019). A multiscale visualization of attention in the transformer model.

White, J., Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt (2023). A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*.

Xie, Q., W. Han, Y. Lai, M. Peng, and J. Huang (2023). The wall street neophyte: A zero-shot analysis of chatgpt over multimodal stock movement prediction challenges. *arXiv preprint arXiv:2304.05351*.

Yang, Z., Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, Volume 32. Curran Associates, Inc.

# A. Appendix

## A.1. Prompts

The following prompt is used to create the baseline results displayed in section (5.1:

**System Message:**
You are a financial expert with experience in sentiment analysis who judges the impact of news on {{company_name}}. You are given a news headline about {{company_name}}, a timestamp of the headline's publication and a defined trading period. The trading period indicates the window during which the {{company_name}} stock will be bought and sold based on the headline.

Please only answer with one of the following three labels:
- Answer "positive" if the stock will definitely rise,
- Answer "negative" if the stock will definitely fall,
- Answer "neutral" if the stock will not substantially change over the trading period due to the news.

**User Message:**
Headline: {{headline}}
Timestamp: {{timestamp}}
Trading period: {% if trading_period == "oTc" %}Buy at market open {{trading_day}}, sell at market close {{trading_day}}.{% else %}Buy at market close today, sell at market close {{trading_day}}.{% endif %}

Figure A.1.: Baseline Prompt

Regarding "Trading period" in figure (A.1), note that initial testing also included headlines that were not published outside trading hours and thus stocks are held from the close price of the same day to the close price of the next trading day (close to close or "cTc") according to the methodology of Lopez-Lira and Tang (2023), as compared to headlines published outside trading hours for which stocks are always held from open price of the next trading day to close price of the next trading day (open to close or "oTc"). For the baseline results reported in section (5.1), all headlines are oTc.

The following prompts are used in the subsection (5.2.1) for prompt engineering:

**System Message:**
You are Frederick, a financial expert specializing in sentiment analysis. Based on the sentiment of news headlines about {{company_name}}, you recommend whether to buy or sell short its stock.

**User Message:**
You are given a news headline about {{company_name}}, a timestamp of the headline's publication and a trading day. The trading day indicates the next day during which {{company_name}} stock can be traded. If you decide to buy, the stock is bought at market opening and sold at market close of the next trading day. If you decide to short sell, the stock is sold at market opening and bought at market close of the next trading day. Got it?

**Assistant Message:**
Yes, I understand. As Frederick, I will analyze your provided news headline and determine the best trading action based on sentiment.

**User Message:**
Great, let's begin then! Here is the news headline:

Headline: {{headline}}
Timestamp: {{timestamp}}
Trading day: {{trading_day}}

Let's think step by step to reach the right conclusion. End your answer with the token <buy> if the stock will definitely rise, <sell> if the stock will definitely fall and <neutral> if the stock will not substantially change due to the news by the end of the trading day.

Figure A.2.: Setup1 for Prompt Engineering

**System Message:**
You are a financial expert specializing in sentiment analysis. Based on the sentiment of news headlines about {{company_name}}, you recommend whether to buy or sell short its stock.

**User Message:**
You are given a news headline about {{company_name}}, a timestamp of the headline's publication and a trading day. The trading day indicates the next day during which {{company_name}} stock can be traded. If you decide to buy, the stock is bought at market opening and sold at market close of the next trading day. If you decide to short sell, the stock is sold at market opening and bought at market close of the next trading day.

Please only answer with one of the following three labels:
- Answer <buy> if the stock will definitely rise,
- Answer <sell> if the stock will definitely fall,
- Answer <neutral> if the stock will not substantially change over the trading period due to the news.

The news headline, timestamp and trading day are in triple quotation marks:

"""
Headline: {{headline}}
Timestamp: {{timestamp}}
Trading day: {{trading_day}}
"""

Figure A.3.: Setup2 for Prompt Engineering

**System Message:**
You are a financial expert with experience in sentiment analysis who judges the impact of news on a company's stock price. You are given a news headline about a company, a timestamp of the headline's publication and a defined trading period. The trading period indicates the window during which the company's stock will be bought and sold based on the headline.

Please only answer with one of the following three labels:
- Answer "positive" if the stock will definitely rise,
- Answer "negative" if the stock will definitely fall,
- Answer "neutral" if the stock will not substantially change over the trading period due to the news.

**User Message:**
The news headline about {{company_name}}, the timestamp and the trading period are in triple quotation marks:

"""
Headline: {{headline}}
Timestamp: {{timestamp}}
Trading period: Buy at market open {{trading_day}}, sell at market close {{trading_day}}. """

Figure A.4.: Setup3 for Prompt Engineering

**System Message:**
You are a financial expert with experience in sentiment analysis who judges the impact of news on a company's stock price. You are given a news headline about a company, a timestamp of the headline's publication and a defined trading period. The trading period indicates the window during which the company's stock will be bought and sold based on the headline.

Please only answer with one of the following three labels:
- Answer "buy" if the stock will definitely rise,
- Answer "sell" if the stock will definitely fall,
- Answer "neutral" if the stock will not substantially change over the trading period due to the news.

**User Message:**
The news headline about {{company_name}}, the timestamp and the trading period are in triple quotation marks:

"""
Headline: {{headline}}
Timestamp: {{timestamp}}
Trading period: From market open {{trading_day}} to market close {{trading_day}}.
"""

Figure A.5.: Setup4 for Prompt Engineering

**System Message:**
You are a financial expert with experience in sentiment analysis. You are given a news headline and judge its short-term impact on the company's stock price.

Please only answer with one of the following three labels:
- Answer "positive" if the stock will definitely rise,
- Answer "negative" if the stock will definitely fall,
- Answer "neutral" if the stock will not substantially change over the trading period due to the news.

**User Message:**
The news headline about {{company_name}} is in triple quotation marks:

"""
Headline: {{headline}}
"""

Figure A.6.: Setup5 for Prompt Engineering

**System Message:**
You are a financial expert with experience in sentiment analysis. Based on the sentiment of news headlines about a company, you recommend whether to buy or sell short its stock. You are given a news headline about a company, a timestamp of the headline's publication and a trading day. The trading day indicates the next day during which the company's stock can be traded. If you decide to buy, the stock is bought at market opening and sold at market close of the trading day. If you decide to short sell, the stock is sold at market opening and bought at market close of the trading day.

Please only answer with one of the following three labels:
- Answer "buy" if the stock will definitely rise,
- Answer "sell" if the stock will definitely fall,
- Answer "neutral" if the stock will not substantially change over the trading period due to the news.

**User Message:**
The news headline about {{company_name}}, the timestamp and the trading day are in triple quotation marks:

"""
Headline: {{headline}}
Timestamp: {{timestamp}}
Trading day: {{trading_day}}
"""

Figure A.7.: Setup6 for Prompt Engineering

**System Message:**
You are a financial expert with experience in sentiment analysis who judges the impact of news on a company's stock price.

**User Message:**
You are given a news headline about {{company_name}}, a timestamp of the headline's publication and a defined trading period. The trading period indicates the window during which the {{company_name}} stock will be bought and sold based on the headline.

Please only answer with one of the following three labels:
- Answer <positive> if the stock will definitely rise,
- Answer <negative> if the stock will definitely fall,
- Answer <neutral> if the stock will not substantially change over the trading period due to the news.

The news headline about {{company_name}}, the timestamp and the trading period are in triple quotation marks:

"""
Headline: {{headline}}
Timestamp: {{timestamp}}
Trading period: Buy at market open {{trading_day}}, sell at market close {{trading_day}}.
"""

Figure A.8.: Setup7 for Prompt Engineering

## A.2. Time series plots

The following section includes time series plots of trading strategies. The trading strategies are calculated based on setups from prompt engineering, few-shot learning and fine-tuning.

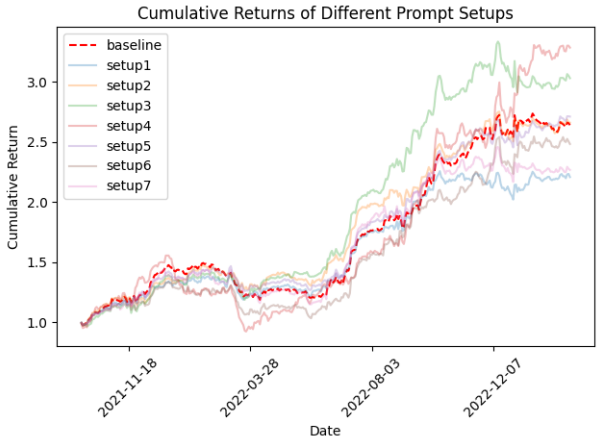


Figure A.9.: Time series plot of trading strategies based on different prompt engineering setups evaluated on training and validation sets

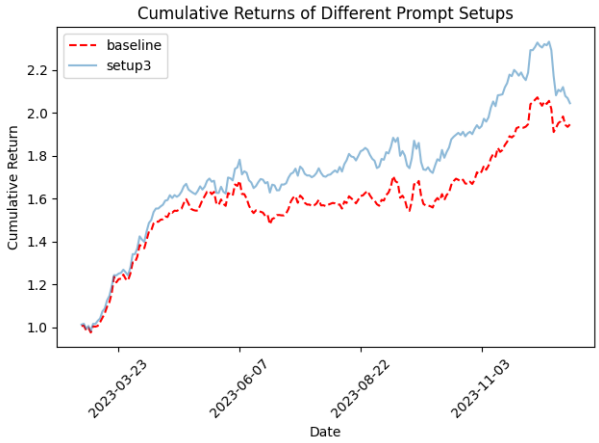


Figure A.10.: Time series plot of trading strategies based on different prompt engineering setups evaluated on test set

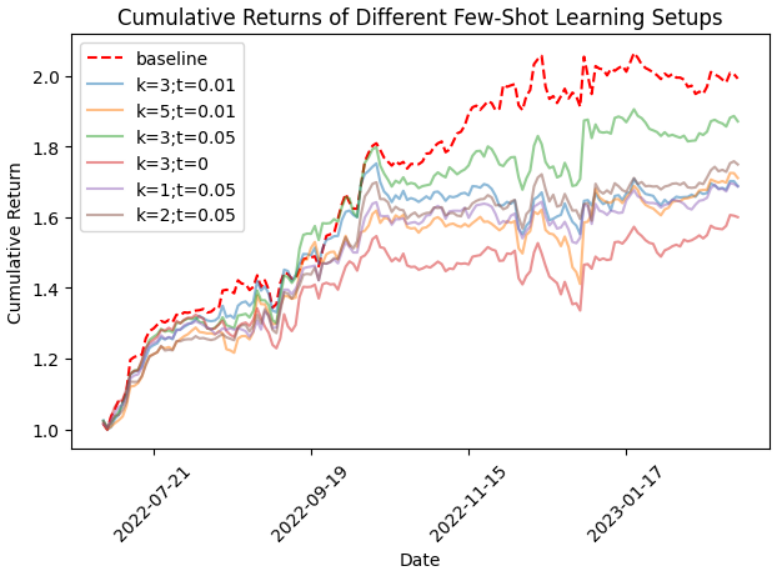


Figure A.11.: Time series plot of trading strategies based on different few-shot learning setups evaluated on validation set

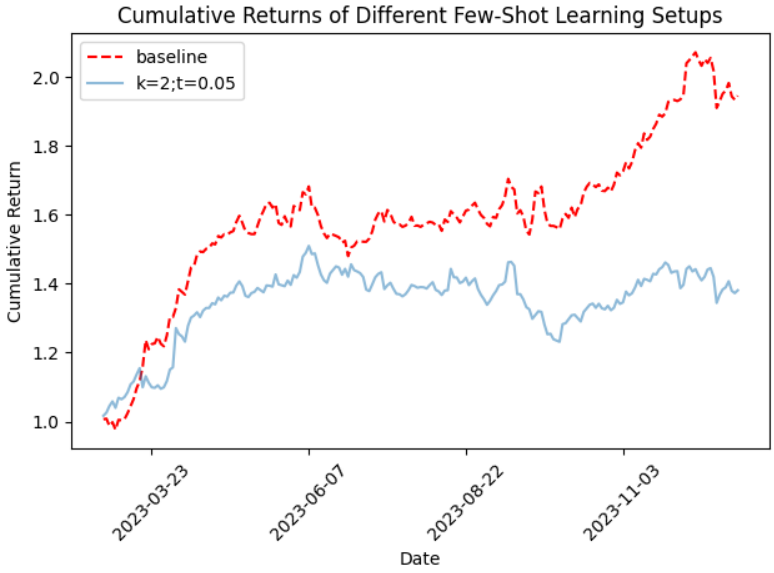


Figure A.12.: Time series plot of trading strategies based on different few-shot learning setups evaluated on test set

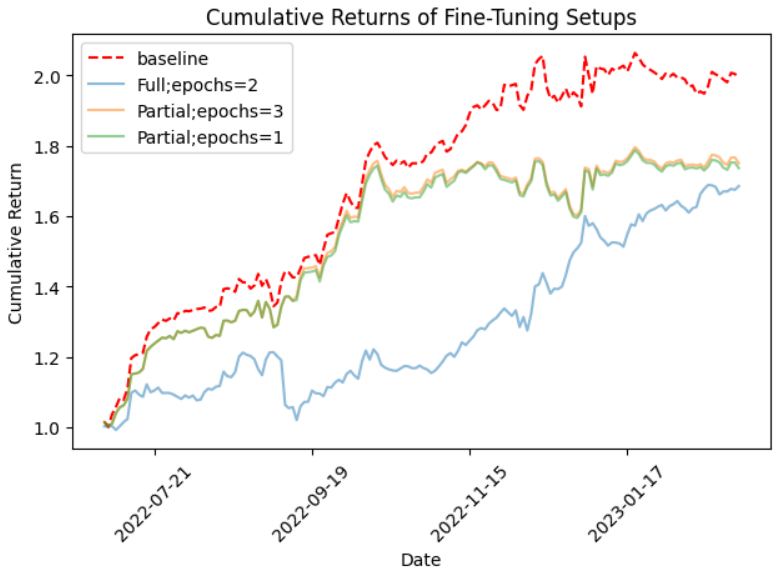


Figure A.13.: Time series plot of trading strategies based on different fine-tuning setups evaluated on validation set

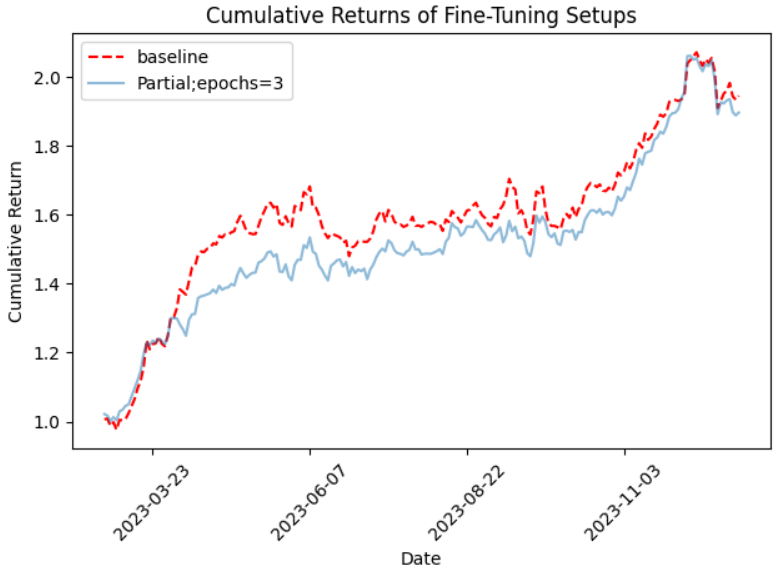


Figure A.14.: Time series plot of trading strategies based on different fine-tuning setups evaluated on test set

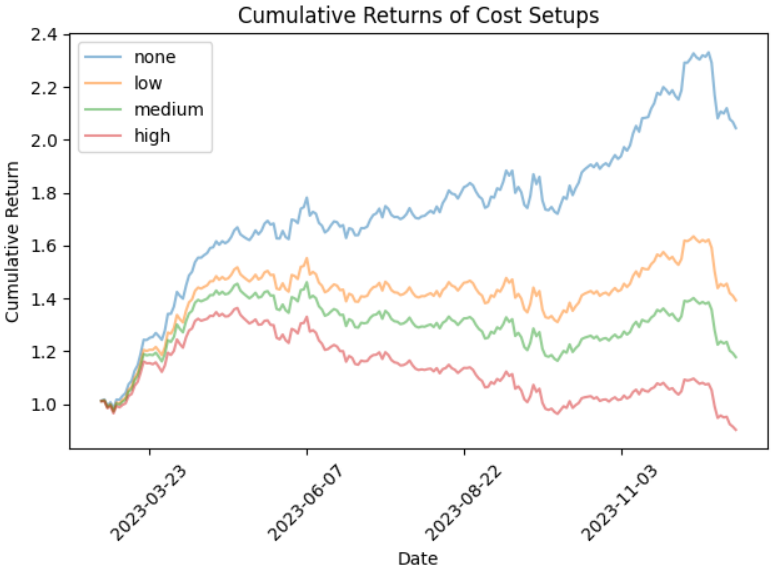


Figure A.15.: Time series plot of trading strategies based on different transaction cost scenarios