Outperforming Morningstar Analysts: Applying Enhanced LLMs to Financial Reports

1st Jonas Gottal
LMU Munich
Department of Computer Science
jonas.gottal@campus.lmu.de

2nd Mohamad Hgog
LMU Munich
Department of Computer Science
m.hgog@campus.lmu.de

***Abstract*—This study examines financial analyst reports from Morningstar, revealing two key findings: firstly, analysts appear to select stocks arbitrarily, and secondly, they provide comprehensive textual justifications that enable informed decisions surpassing mere chance. We employ enhanced large language models to efficiently analyze the textual data, facilitating a broad exploration of various pre-trained models to identify the subtle underlying sentiment and extract more value from the reports than the experts themselves. The code is available at github.com/trashpanda-ai.**

***Index Terms*—LLMs, PEFT, Adapters, Hugging Face, Finance**

I. Introduction

Many search for guidance in financial analyst reports for investment purpose. Especially during the COVID-19 pandemic, the number of retail traders has increased significantly. And Morningstar is one of the most popular platforms for financial reports that are widely used by both retail traders and professionals.

However, the question arises whether these reports are actually valuable or just an illusion of expertise. In this paper, we investigate the value of analyst reports and whether they can outperform the market. We hypothesize that the reports contain valuable information that can be extracted using enhanced large language models (LLMs) to outperform the analysts and even the market.

II. Data

Financial analysts provide detailed reports on single companies (equities), detailing their current situation and future prospects. These reports are based on a variety of factors, including their financial strength, economic moat and cost allocation. The reports also contain a rating and a fair price estimate.

A. Data description

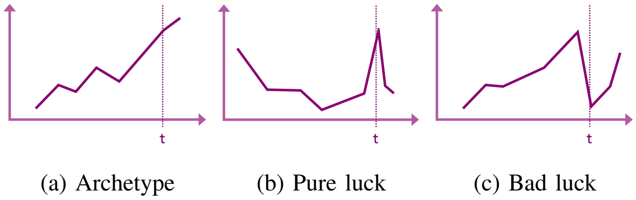
The data is retrieved from Morningstar and contains the following columns:

- `ParseDate`: The date the information was retrieved.
- `Title`: The title of the analyst report.
- `CompanyName`: The name of the company.
- `TickerSymbol`: The ticker symbol (without exchange information) of the underlying stock.
- `Rating`: The analyst rating of the stock.
- `ReportDate`: The date of the release of the report.
- `AuthorName`: The name of the author of the report.
- `Price`: The price of the stock declared in the report.
- `Currency`: The given currency of the price.
- `PriceDate`: The date the price was retrieved from market data (Morningstar).
- `FairPrice`: The estimated fair price from the analyst.
- `Uncertainty`: The company's uncertainty quantified in 'Low', 'Medium', 'High' and 'Very High'.
- `EconomicMoat`: The ability to maintain competitive advantages quantified in 'Narrow' and 'Wide'.
- `CostAllocation`: Decisions on investments categorized as 'Poor', 'Standard', and 'Exemplary'.
- `FinancialStrength`: Rating of the ability to make timely payments and fulfill obligations – quantified in 'A', 'B', ... 'F'.
- `AnalystNoteDate`: The date of the analyst note.
- `AnalystNoteList`: The analyst note.
- `BullsList`: Arguments in favor of the company.
- `BearsList`: Arguments against the company.
- `ResearchThesisDate`: The date of the thesis.
- `ResearchThesisList`: Objective research thesis.
- `MoatAnalysis`: Added research on EconomicMoat.
- `RiskAnalysis`: Company's risk profile.
- `CapitalAllocation`: Text on the CostAllocation.
- `Profile`: Short text on the company profile.
- `FinancialStrengthText`: Short text on the financial strength of the company (conclusion).

The most important information for the application of LLMs is the textual data – starting with `AnalystNoteList`, which we expect to have the most effective content due to its subjective tone. We also analyze more objective texts like `BullsList` and `BearsList` as well as `ResearchThesisList`, `MoatAnalysis`, `RiskAnalysis`, `CapitalAllocation`, `Profile` and `FinancialStrengthText`. Since the target group of financial analyst reports is investors, we focus on the long term perspective and not on short-term speculation. Therefore, we look at the stock price development over a longer period. The data is retrieved from Yahoo Finance and contains the following columns: `Date`, `Open`, `High`, `Low`, `Close`, `Adjusted Close` and `Volume`. We base our target variable for the sentiment analysis on the Adjusted Close price.

B. Data Preprocessing

Since we care about investing and not speculation, we pre-process the data to remove spurious spikes and other noise. We focus on the archetype of a good investment as shown in Figure 1a and ignore spurious spikes as shown in Figure 1b and Figure 1c. This means we consider the average performance within a certain time period – here the maximum amount since the last report was released: 120 days. Inspired by the commonly used moving averages in mathematical finance [1], the preprocessing is done by calculating the average stock price over 120 days since publication and comparing it to the stock price at the time of the report. If the average stock price is higher[1] than the price at time of release, we label it as a good investment, otherwise as a bad investment. This is our binary target variable for the following analysis.



(a) Archetype (b) Pure luck (c) Bad luck

Fig. 1: Different stock developments as charts: Comparing the archetype of a good investment to spurious spikes (positive and negative).

The final data (1586 reports) is split into a training, development and test set with a ratio of 70:10:20 (1103, 165, 318). The training set is used to train the models, the development set to optimize the hyperparameters and the test set to evaluate the models.

III. Foundations

Our research efficiently leverages the capabilities of LLMs by utilizing Transformers and Adapters, primarily sourced from Hugging Face [2]. This approach is fundamentally driven by the need to process extensive textual data accurately and efficiently, particularly focusing on sentiment classification within financial texts.

Utilizing LLMs, including those based on Transformer architectures, has demonstrated unparalleled success in understanding and generating human-like text. Their ability to capture deep linguistic and semantic nuances makes them ideal for complex NLP tasks like sentiment analysis. However, the computational intensity of training such models from scratch demands significant resources [2].

To efficiently handle LLMs, we use the Transformer architecture and models provided by Hugging Face [2]. We also implement Adapters, which offer a parameter-efficient way to fine-tune pre-trained models on domain-specific tasks [3]. By inserting small, trainable modules within the Transformer layers, we can tailor the model's responses to the subtleties of financial language without overhauling the entire network architecture. This approach significantly reduces the computational overhead and facilitates quicker iterations and refinements.

Our research aims to classify the sentiment expressed in Analyst Reports from Morningstar accurately. Analyzing the sentiment, we can extract meaningful insights about market trends and investor perceptions, which are crucial for making informed financial decisions. Sentiment classification in this context involves determining the polarity of the text—whether it conveys a positive, negative, or neutral sentiment—which can significantly influence investment strategies and business outcomes.

A. Transformer models

Transformer models are a class of deep learning architectures that have revolutionized the field of natural language processing (NLP). Based on the encoder-decoder architecture, these models leverage self-attention mechanisms to process data sequences in parallel and capture complex dependencies across the sequence [4] [2]. The encoder maps an input sequence of symbol representations $(x_1, \ldots, x_n)$ to a continuous representation $\mathbf{z} = (z_1, \ldots, z_n)$. The decoder generates an output sequence $(y_1, \ldots, y_m)$ from $\mathbf{z}$, using an inherently auto-regressive method, processing previous symbols to generate the next [4].

Figure 2 provides a detailed view of the Transformer architecture, illustrating both the encoder and decoder components.

The backbone of the Transformer is the self-attention mechanism in both the encoder and decoder stacks, which allows each position in the sequence to attend to all positions in the previous layer of the model. Each encoder and decoder layer consists of multi-head self-attention mechanisms followed by position-wise fully connected feed-forward networks. Residual connections and layer normalization are employed around each sub-layer to facilitate the training of deep networks [4]. Formally, each layer output can be described as:

$$\text{Layer Output} = \text{LayerNorm}(x + \text{Sublayer}(x)),$$

where $\text{Sublayer}(x)$ represents the operations within the sub-layer itself.

Self-attention allows the model to weigh the importance of different words in the sequence irrespective of their positional distances. The attention function can be described as mapping a query and a set of key-value pairs to an output, where the output is a weighted sum of the values. A compatibility function of the query with the corresponding key computes the weight assigned to each value. The scaled dot-product attention is computed as:

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

[1] A sideway movement within a 2% margin is not classified as "higher".

where Q, K, and V are the matrices representing queries, keys, and values, respectively, and d_k is the dimensionality of the keys.

Multi-head attention, a pivotal extension of the basic attention mechanism, projects the queries, keys, and values multiple times with different, learned linear transformations to allow the model to attend to information from different representation subspaces jointly:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$
$$\text{where head} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V).$$

Here, W_i^Q, W_i^K, and W_i^V are parameter matrices for different heads, and W^O is the output projection matrix.

Furthermore, the model incorporates **position-wise feed-forward networks** in each layer, consisting of two linear transformations with a ReLU [5] activation in between:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Positional encodings are added to input embeddings to give the model access to the position of the tokens in the sequence. This is crucial since the model lacks recurrence or convolution:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$
$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right),$$

where pos is the position and i is the dimension.

B. Parameter efficient fine-tuning (PEFT)

When fine-tuning large pre-trained models for various tasks, computational efficiency and scalability require innovative approaches. Parameter-efficient fine-tuning (PEFT) addresses this challenge by enabling the adaptation of large models to specific tasks without the need to update all model parameters. Thus, PEFT conserves computational resources and enhances the feasibility of exploring multiple models [3].

Adapters are central to the PEFT strategy. An adapter is a small neural network module inserted between the layers of a pre-existing model, such as a Transformer. As illustrated in Figure 3, the adapter modifies the Transformer block by inserting additional neural network layers that allow for fine-tuning on specific tasks with minimal updates to the overall model parameters.

Typically, these adapters consist of a down-projection that reduces the dimensionality of the layer output, a non-linear activation function, and an up-projection that restores the dimensionality to its original size [3]. The formal structure of an adapter, shown in Figure 4, can be expressed as:

$$\text{Adapter}(x) = x + U(\sigma(D(x))),$$

where x is the input to the adapter, D represents the down-projection, σ is a non-linear activation function (e.g., ReLU), and U is the up-projection.

Adapters offer a significant advantage in that only the parameters within these modules need to be updated during fine-tuning, leaving the vast majority of the pre-trained model's parameters frozen. This is particularly advantageous for several reasons:

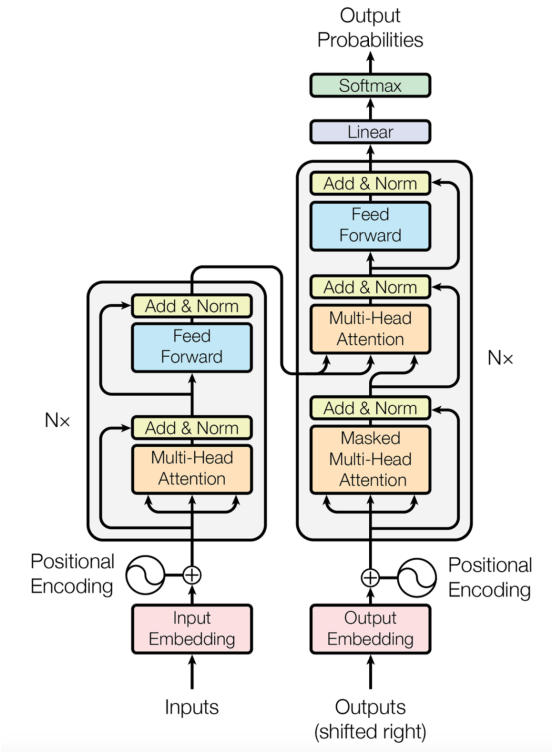


Fig. 2: The architecture of a Transformer model, showing the flow from input through the encoder and decoder to the output [4].

- **Computational Efficiency:** Since only a small fraction of the overall parameters are updated, the computational overhead is significantly reduced compared to full model fine-tuning.
- **Preservation of Pre-trained Knowledge:** By keeping most of the original model parameters unchanged, the risk of catastrophic forgetting is minimized, thus preserving the knowledge that the model has acquired during its initial extensive training.
- **Scalability:** Adapters allow for quick adaptation to multiple tasks without the need for extensive retraining, making it feasible to fine-tune large models on a wide range of tasks, even with limited computational resources.

[6]

C. Hugging Face

Hugging Face is an essential platform in NLP and machine learning. It provides an extensive repository of pre-trained models and tools for training, fine-tuning, and deploying models across various tasks [2]. The platform is renowned for implementing state-of-the-art models based on Transformer architectures like BERT [8], GPT [9], and their derivatives.

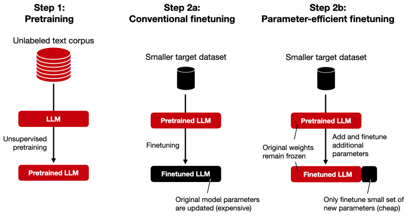


Fig. 3: Process of PEFT [7].

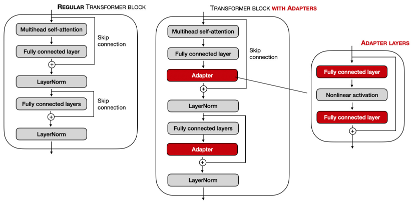


Fig. 4: Process of PEFT (detailed) [7].

Why Use Hugging Face? The primary appeal of Hugging Face lies in its comprehensive ecosystem that simplifies the deployment of machine learning models. There are several reasons why it is widely used:

- **Accessibility of Pre-trained Models:** Hugging Face offers access to thousands of pre-trained models that are easily adaptable to a broad range of tasks without the need for scratch training.
- **Ease of Integration:** The platform supports various programming languages and frameworks, including Python, PyTorch [10], and TensorFlow [11], enabling seamless integration into existing projects.
- **Community and Support:** With a robust community of developers and researchers, Hugging Face facilitates collaboration and sharing of best practices, contributing to the continual improvement and extension of its model offerings.

Benefits in Practical Applications: Utilizing Hugging Face allows us to reduce the time and resources required for model development dramatically. Specifically, the platform provides functionalities for:

- **Fine-Tuning Pre-trained Models:** Users can fine-tune models on specific datasets, which is essential for tailoring the model's responses to the nuances of particular domains or tasks.
- **Experimentation:** The ease of accessing and modifying different models promotes rapid prototyping and experimentation, a valuable feature in the fast-evolving field of AI.
- **Deployment:** Hugging Face also offers solutions for easy deployment of trained models, which is beneficial for bringing AI applications to production.

Application in Our Work: In our project, we leverage Hugging Face to fine-tune pre-trained models specifically for analyzing Analyst Reports from Morningstar. This involves adapting state-of-the-art Transformer models [2] to comprehend and interpret complex financial texts, which is critical for extracting actionable insights and automating parts of the financial analysis process. The ability to fine-tune and deploy models efficiently with Hugging Face accelerates our research and application development, enabling a focused approach to model performance and application-specific adjustments rather than infrastructure and model maintenance.

IV. Literature

As described in [12], there are many different applications of LLMs, and specifically for analyst reports they can provide insights into subtle tone and sentiment to add value. However, at the moment there is only research based on data created by consumers, such as Reddit [13]. Furthermore, it has been shown that reports may have a small positive performance in the right market conditions [14], but it is unclear whether only recommendations themselves can influence markets [15] and reports have no value at all [16]. Thus, the herding factor may have an impact on stock performance [17]. In [18] they use LLMs to better interpret and analyze Korean financial analysts' reports. In this paper, however, we use data from widely spread reports (from retail traders to professionals) on globally traded firms. We show that while analysts can barely outperform the current market, they provide comprehensive textual justifications that enable informed decisions. We use enhanced large language models to efficiently analyze the textual data, allowing a broad exploration of different pre-trained models to identify the subtle underlying sentiment and extract more value from the reports than the experts themselves.

V. Approach

Our ultimate objective can be interpreted as a binary classification problem based on sentiment analysis. We use the textual data from the reports to predict whether the stock is a good investment or not. In order to do so we compare the performance of different models and configurations to find the best one. We also compare the performance of the models to the analysts' own predictions and other metrics from the report to create a benchmark. In order to explore many different models and configurations, we use the parameter efficient fine-tuning approach. We use adapters to fine-tune the models and Hugging Face to access the pre-trained models. Even though literature has shown adapters to be more efficient while retaining the performance of the underlying model, we setup a preliminary experiment with the Stanford Sentiment Treebank v2 (SST2) data set for sentiment classification to validate our approach and compared the efficiency and performance of the models:

Therefore, we used PEFT to find the best performing setup and built a automated training and evaluation pipeline to efficiently explore the model landscape of Hugging Face.

A. Model selection

Since our approach is centered around sentiment classification of financial reports, we allocated the most fitting

	Fine-tuned LLM	Fine-tuned Adapter
Training Runtime	1h 58m	**24m**
Evaluating Accuracy	**0.908**	0.902
Evaluation Runtime	6.83s	**1.84s**
Evaluation Loss	0.42	**0.31**

TABLE I: Comparison of Fine-tuned LLM and Fine-tuned Adapter

pre-trained models from both worlds – sentiment analysis and finance:

Sentiment:

- `kwang123/bert-sentiment-analysis`
- `siebert/sentiment-roberta-large-english`
- `distilbert/distilbert-base-uncased-finetuned-sst-2-english`

Finance:

- `ProsusAI/finbert`
- `yiyanghkust/finbert-tone`
- `bardsai/finance-sentiment-pl-fast`
- `RashidNLP/Finance-Sentiment-Classification`
- `ahmedrachid/FinancialBERT-Sentiment-Analysis`
- `soleimanian/financial-roberta-large-sentiment`
- `nickmuchi/sec-bert-finetuned-finance-classification`
- `nickmuchi/deberta-v3-base-finetuned-finance-text-classification`

Within the financial subset, there are also models fine-tuned to the task of sentiment classification on financial text data, such as publicly available SEC filings.

B. Adapter configuration

Configuring adapters for machine learning models, particularly in Transformer architectures, allows for targeted modifications to model behavior while preserving the bulk of the pre-trained parameters. Adapters offer a flexible approach to fine-tuning, where specific components are added or adjusted to alter the model's functionality for specialized tasks [3].

Adapter Configuration Options: The configuration of adapters can vary widely depending on the desired outcome and the specific model architecture. Common configurations include altering the number of adapter layers, the size of the feed-forward networks, the non-linearity type, and the training strategy. The choice of configuration typically balances between computational efficiency and task-specific performance [19].

Industry Standard and Our Usage: For the industry, especially in areas involving complex sequence understanding like financial analysis, configurations that offer a good trade-off between performance and computational demand are preferred. In our work, we experimented with several adapter configurations to identify the most effective setup for analyzing financial reports.

Adapter Configurations Used in Our Work:

- **LoRAConfig:** LoRA (Low-Rank Adaptation) employs rank decomposition to modify attention and feed-forward layers within a Transformer. By adjusting only a small subset of model parameters, LoRAConfig focuses on the adaptability of the model with minimal computational overhead [20].
- **UniPELTConfig:** The Unified Framework for Parameter-Efficient Language Model Tuning (UniPELT) is a framework that combines various parameter-efficient techniques as sub-modules. UniPELTConfig eliminates the need for model selection by optimizing sub-module activation based on their contribution to the task, ensuring consistently high performance while maintaining model efficiency [21].
- **MAMConfig:** The Mix-And-Match (MAM) adapter optimizes attention sub-layers efficiently. It adapts to new tasks quickly with fewer parameters updated and achieves comparable results to full fine-tuning. MAM adapter has strong performance in adjusting to new data distributions and task requirements, making it an optimal choice for rapid deployment scenarios [22].
- **ConfigUnion (LoRAConfig, PrefixTuningConfig, SeqBnConfig):** This configuration combines LoRA, Prefix Tuning [23], and Sequence Bucketing strategies [24]. Prefix Tuning involves prepending a sequence of trainable vectors to the input, while Sequence Bucketing adjusts the model based on sequence length. This union aims to leverage the strengths of each individual approach for enhanced overall performance [3].

C. Exploration and Exploitation

Since running all possible combinations of configurations is intractable, we assume independence of our various parameters in our experimental setup and base our approach on exploration and exploitation. We explore all possible valid models and configurations and exploit the best ones in the next iteration. First, we compare the different models, then we optimize for the best adapter configuration, and after that we search for the best text input. Finally, we compare our PEFT approach to fine-tuning the LLM itself. All experiments are performed on the same data, using the same training and test split to ensure comparability, and we use the same evaluation metrics, training parameters and hyperparameters for all models. Everything is run on the same hardware.

D. Evaluation

In order to validate learned models we compare predictions made from the model with previously separated test data. In this way, we can objectively evaluate them according to their predictive qualities. For this comparison, we utilize these fundamental principles:

Sensitivity or true positive rate (TPR) is derived from the true positives TP, i.e., the correctly identified positives P from the test set: $\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}}$
Specificity or true negative rate (TNR) is derived from the true negatives TN, i.e., the correctly identified negatives N from the test set: $\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{N}}$

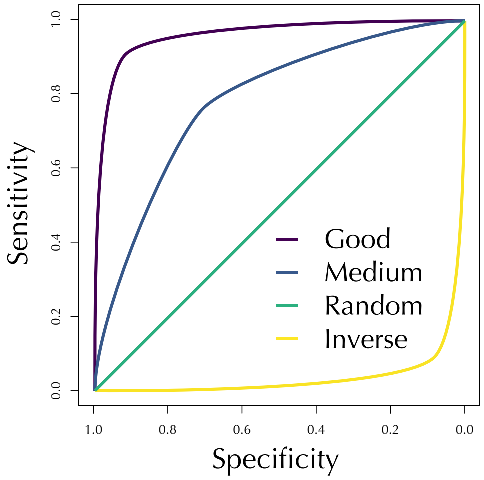


Fig. 5: Exemplary receiver operating curves for a *good, medium, random* and *inverse* model fit. The perfect score would be 1.0 on each dimension – specificity and sensitivity.

By plotting both the sensitivity and specificity in relation for various threshold values, we obtain the so called Receiver Operating Characteristic (ROC) curve. For the results of both measures 1.0 is the optimum, and if the curve is the diagonal, we observed a random process. In Figure 5, we can see some common examples of curves. In order to further summarize these evaluations, we can calculate the Area Under the Curve (AUC) to rank the models. Again, a value of 0.5 indicates a random process. The reason we use ROC rather than accuracy, precision, recall and f1 scores is that ROC is a more comprehensive metric that provides a more detailed view of model performance. In addition, ROC curves make it easier to compare the performance of different models across all classification thresholds. This also makes it more robust to class imbalance, which is very important for inherently unbalanced data such as financial markets, which often follow trends and alternate between bear and bull markets. It also helps to understand how the performance of the model changes with different classification thresholds, which can be crucial for decision making in different conditions where the cost of false positives and false negatives can vary. [25] [26] [27]

VI. Results

In this section we present the results of our experiments. We compare the models with classical approaches based on the analysts' forecasts themselves and other metrics from the report to establish a benchmark. We present the results of our model training in three steps: First, we compare the different models, then we optimize the best adapter configuration, and finally we search for the text input that provides the highest predictive power. We also directly fine-tune the underlying model to see if PEFT loses significant predictive power. Furthermore, we use feature attribution via CAPTUM to gain insight into the models and provide an overview of training time, accuracy and number of parameters.

A. Benchmarking

In order to evaluate our models accurately, we first create a benchmark based on the analysts' own predictions. We use the analysts' `FairPrice` to create binary predictions and use them as input for a logistic regression model as a first benchmark. We also use all the categorical (and ordinal) information from the reports as input for an gradient boosted tree model (XGBoost [27]) as a benchmark: `Rating`, `AuthorName`, `Uncertainty`, `EconomicMoat`, `CostAllocation`, `FinancialStrength`, `Sector`, `Exchange`, `Prediction`. We assume that certain authors might be better than others, and that the uncertainty, economic moat, cost allocation and financial strength might be valid indicators of a firms success. We also assume that the sector and exchange might have an impact on the stock performance. We use the same training and test split as for the LLMs and the same evaluation metrics. The results are shown in Figure 6.

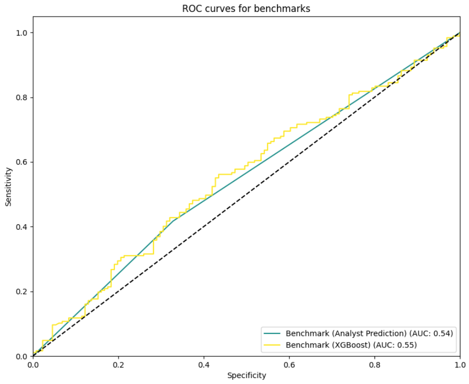


Fig. 6: Benchmark Experiment: a diagonal line represents a random process. For financial markets it means reproducing the current distribution (following the trend).

The results show that neither the analysts nor the XGBoost model can outperform randomness – it only reproduces the current distribution of the market (trend following). This is a strong indication that the reports themselves may not contain any valuable information.

B. Models in comparison

Our approach assumes that all changes in our experiments are independent so that the results are comparable. As first step we compare the different models on the same adapter configurations and text inputs to find the best one.

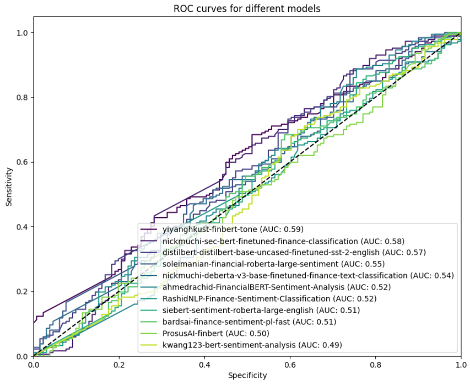


Fig. 7: First Experiment

We continue for the two best models (both trained on financial sentiment data) and compare the best adapter configurations found in literature:

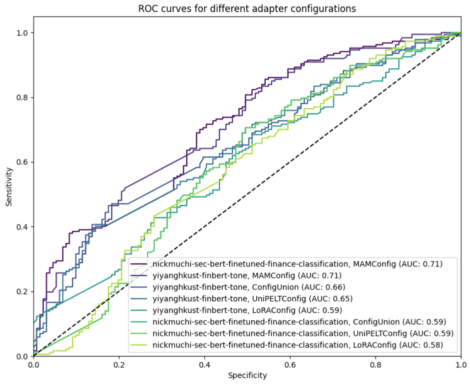


Fig. 8: Second Experiment

Next to `AnalystNoteList` we also explore `CapitalAllocation`, `BullsList`, `BearsList`, `MoatAnalysis`, `ResearchThesisList`, `Risk Analysis`, `Profile`, and `FinancialStrengthText` as input data. We find that `FinancialStrengthText` as a concise summary provides the best predictive power, as shown in Figure 9.

Finally, we use the best combination found to directly fine tune the underlying model. This leads to only a small improvement of 0.0048 AUC and the training time was slightly higher (from 480 sec to 520 sec), but the number of parameters could be drastically decreased from 110 million to 23 million. The results are shown in Figure 10.

While there is no overall cutoff value and it highly depends on the specific application, with values above 0.7 [28], we reached a robust predictive quality.

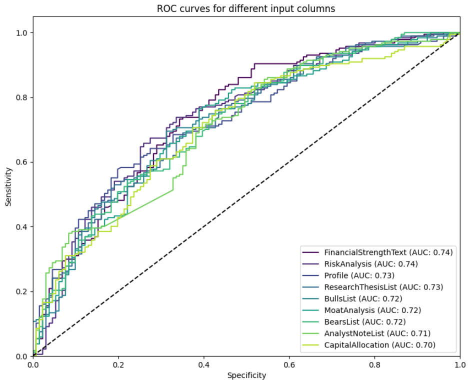


Fig. 9: Third Experiment

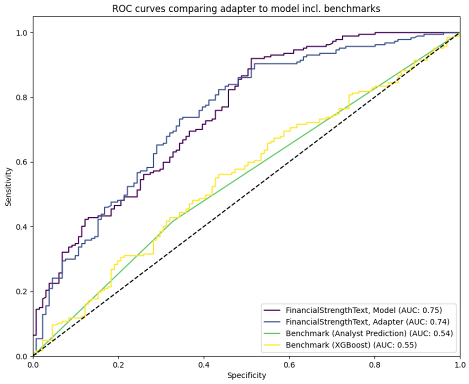


Fig. 10: Final Result.

C. *Insights via Captum*

Captum, derived from the Latin word for ”understanding,” is an interpretability library developed by Meta AI for the PyTorch framework [10]. It features several algorithms to identify data feature contributions. It helps validate model behavior, ensure fairness, and debug complex machine learning applications [29].

A key method in Captum for feature attribution is the *Layer Integrated Gradients*, which extends the *Integrated Gradients* approach [30]. This method is particularly adept at handling models with deep architectures, such as BERT [8], by attributing importance to input features relative to a specific layer's output, thus offering more granular insight [30].

In our project, we utilize Captum (as in the tutorial [31]) to analyze and validate the feature attributions of our fine-tuned models on financial texts from Morningstar analyst reports. This ensures that our models' predictive decisions are based on relevant and justifiable data, fostering trust and transparency in automated financial analysis.

The first case (see Figure 11) showcases an input where the model's prediction of financial strength was substantially higher. This text highlights a high financial position characterized by a high level of cash reserves, no debt, and a positive free cash flow, indicating strong financial health and operational efficiency.

In contrast, the second plot (see Figure 12) corresponds to an input scenario where the model's prediction of financial strength was at its lowest. The text discusses significant concerns about a company's ability to meet near-term obligations due to a notable increase in its debt-equity ratio and a sharp decline in interest coverage ratios.

These contrasting scenarios underscore the effectiveness of Captum in providing meaningful insights into how specific features in the text influence the model's output, thus assisting in understanding the underlying reasons behind model predictions.

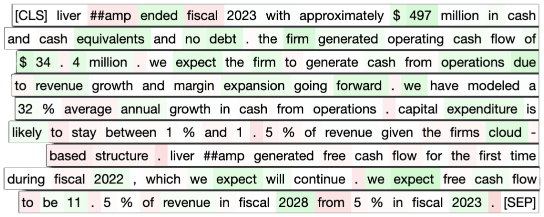

[CLS] liver ##amp ended fiscal 2023 with approximately $ 497 million in cash and cash equivalents and no debt . the firm generated operating cash flow of $ 34 . 4 million . we expect the firm to generate cash from operations due to revenue growth and margin expansion going forward . we have modeled a 32 % average annual growth in cash from operations . capital expenditure is likely to stay between 1 % and 1 . 5 % of revenue given the firms cloud - based structure . liver ##amp generated free cash flow for the first time during fiscal 2022 , which we expect will continue . we expect free cash flow to be 11 . 5 % of revenue in fiscal 2028 from 5 % in fiscal 2023 . [SEP]

Fig. 11: Text input from Analyst Report indicating high financial strength prediction.

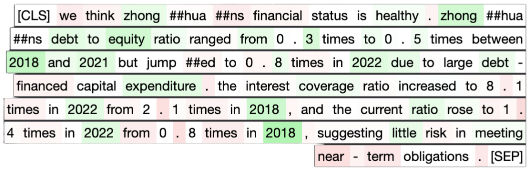

[CLS] we think zhong ##hua ##ns financial status is healthy . zhong ##hua ##ns debt to equity ratio ranged from 0 . 3 times to 0 . 5 times between 2018 and 2021 but jump ##ed to 0 . 8 times in 2022 due to large debt - financed capital expenditure . the interest coverage ratio increased to 8 . 1 times in 2022 from 2 . 1 times in 2018 , and the current ratio rose to 1 . 4 times in 2022 from 0 . 8 times in 2018 , suggesting little risk in meeting near - term obligations . [SEP]

Fig. 12: Text input from Analyst Report indicating low financial strength prediction.

D. *Training analysis*

Training analysis involves systematically evaluating model performance metrics across different configurations to optimize accuracy, computational efficiency, and effectiveness. In our work, we analyze the impact of various Adapter configurations on Transformer-based models dedicated to sentiment classification tasks within the financial domain.

Number of Parameters vs. Accuracy: The first plot in Figure 13 in our analysis illustrates the relationship between the number of parameters in different Adapter configurations and the achieved accuracy. The configurations exhibit diverse accuracies; notably, MAMConfig, which possesses a higher parameter count, tends to achieve greater accuracy. On the other hand, configurations like LoRAConfig and UniPELTConfig show a broad spread in results, highlighting their varying effectiveness dependent on specific implementation and training data nuances.

Training Time vs. Accuracy: The second plot in Figure 14 correlates the training time with accuracy, shedding light on the computational efficiency of each Adapter configuration. Here, MAMConfig configurations demonstrate high accuracy but require longer training times, which may not be feasible in scenarios necessitating rapid model deployment. Conversely, ConfigUnion presents a compromise, offering a balanced trade-off between training duration and accuracy, suggesting its potential suitability for practical applications.

Model Accuracy with LoRA Adapter Configuration: The third plot in Figure 15 displays the accuracies achieved by different models using the LoRA Adapter configuration. While models such as ”yiyanghkust/finbert-tone” and ”nickmuchi/sec-bert-finetuned-finance-classification” exhibit high accuracies, they do not consistently outperform in all training metrics, suggesting that accuracy alone might not reliably indicate a model's effectiveness in real-world applications.

Given that high accuracy during training does not invariably predict superior real-world performance, our selection of optimal models was guided by additional metrics, including the AUC scores. This approach ensures the final model selection performs well in different contexts and aligns with practical needs.

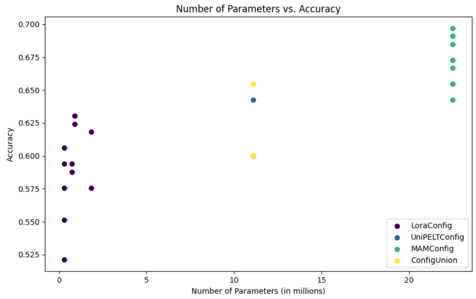


Fig. 13: Number of parameters vs. Accuracy, clustered by adapter configuration.

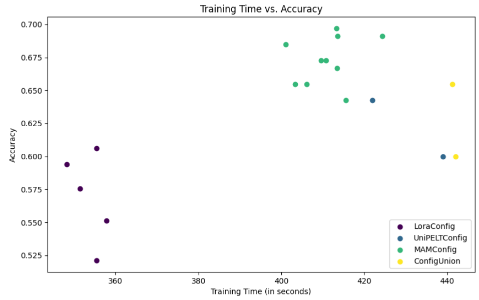


Fig. 14: Training time vs. Accuracy, clustered by adapter configuration.

VII. Conclusion

The results show that neither the analysts nor the XGBoost model, which is based on the hard facts in the report,

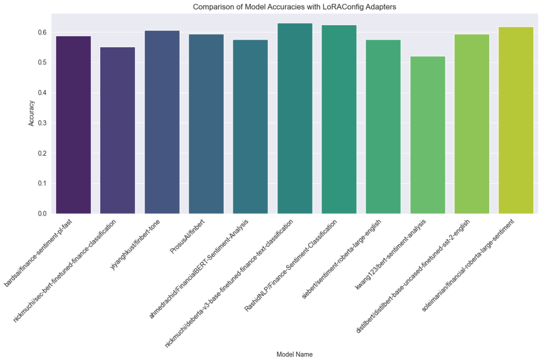


Fig. 15: All models and their corresponding accuracy on LoRA adapter configuration.

can beat randomness. They only reproduce the current distribution – i.e. they follow the market trend. However, we have also found that the reports themselves contain valuable information that can be extracted. But to do so, we need LLMs that are fine-tuned to large amounts of financial data. By using Hugging Face's easily accessible models, we have shown that we can extract more value from the reports than the experts themselves.

References

[1] J. C. Hull, *Options, Futures, and other Derivatives*, 11th ed. Boston: Pearson, 2021.

[2] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Huggingface's transformers: State-of-the-art natural language processing," 2020.

[3] J. Pfeiffer, A. Rücklé, C. Poth, A. Kamath, I. Vulić, S. Ruder, K. Cho, and I. Gurevych, "Adapterhub: A framework for adapting transformers," 2020.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[5] A. F. Agarap, "Deep learning using rectified linear units (relu)," 2019.

[6] C. Poth, H. Sterz, I. Paul, S. Purkayastha, L. Engländer, T. Imhof, I. Vulić, S. Ruder, I. Gurevych, and J. Pfeiffer, "Adapters: A unified library for parameter-efficient and modular transfer learning," 2023.

[7] S. Raschka, "Finetuning llms efficiently with adapters," May 2023. [Online]. Available: https://magazine.sebastianraschka.com/p/finetuning-llms-with-adapters

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

[9] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. 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. M. 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, "Language models are few-shot learners," 2020.

[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

[11] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[12] H. Zhao, Z. Liu, Z. Wu, Y. Li, T. Yang, P. Shu, S. Xu, H. Dai, L. Zhao, G. Mai, N. Liu, and T. Liu, "Revolutionizing finance with llms: An overview of applications and insights," 2024.

[13] X. Deng, V. Bashlovkina, F. Han, S. Baumgartner, and M. Bendersky, "What do llms know about financial markets? a case study on reddit market sentiment analysis," in *Companion Proceedings of the ACM Web Conference 2023*, ser. WWW '23 Companion. New York, NY, USA: Association for Computing Machinery, 2023, pp. 107 – 110. [Online]. Available: https://doi.org/10.1145/3543873.3587324

[14] C. Su, H. Zhang, and R. S. Hudson, "The time-varying performance of uk analyst recommendation revisions: Do market conditions matter?" *Financial Markets, Institutions & Instruments*, vol. 29, no. 2, pp. 65 – 89, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/fmii.12126

[15] M. Brauer and M. Wiersema, "Analyzing analyst research: A review of past coverage and recommendations for future research," *Journal of Management*, vol. 44, no. 1, pp. 218 – 248, 2018. [Online]. Available: https://doi.org/10.1177/0149206317734900

[16] V. Panchenko, "Impact of analysts' recommendations on stock performance," *The European Journal of Finance*, vol. 13, no. 2, pp. 165 – 179, 2007. [Online]. Available: https://doi.org/10.1080/13518470500459782

[17] M. Palmer, M. Eickhoff, and J. Muntermann, "Detecting herding behavior using topic mining: The case of financial analysts," in *Research Papers*, vol. 97, 06 2018.

[18] S. Kim, S. Kim, Y. Kim, J. Park, S. Kim, M. Kim, C. H. Sung, J. Hong, and Y. Lee, "Llms analyzing the analysts: Do bert and gpt extract more value from financial analyst reports?" in *Proceedings of the Fourth ACM International Conference on AI in Finance*, ser. ICAIF '23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 383 – 391. [Online]. Available: https://doi.org/10.1145/3604237.3627721

[19] B. Min, H. Ross, E. Sulem, A. P. B. Veyseh, T. H. Nguyen, O. Sainz, E. Agirre, I. Heintz, and D. Roth, "Recent advances in natural language processing via large pre-trained language models: A survey," *ACM Computing Surveys*, vol. 56, no. 2, pp. 1–40, 2023.

[20] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021.

[21] Y. Mao, L. Mathias, R. Hou, A. Almahairi, H. Ma, J. Han, W. tau Yih, and M. Khabsa, "Unipelt: A unified framework for parameter-efficient language model tuning," 2022.

[22] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig, "Towards a unified view of parameter-efficient transfer learning," 2022.

[23] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," 2021.

[24] J. Pfeiffer, I. Vulić, I. Gurevych, and S. Ruder, "Mad-x: An adapter-based framework for multi-task cross-lingual transfer," 2020.

[25] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. New York, US: Springer New York, 2009.

[26] G. Kauermann, H. Küchenhoff, and C. Heumann, *Statistical Foundations, Reasoning and Inference*. Cham, Switzerland: Springer Nature Switzerland, 2021.

[27] S. J. Russell and P. Norvig, *Artificial Intelligence – A Modern Approach, Global Edition*, 4th ed., ser. Pearson Education. Harlow, United Kingdom: Prentice Hall, 2021.

[28] D. W. Hosmer Jr., S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, 3rd ed. John Wiley & Sons, 2013.

[29] N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, B. Alsallakh, J. Reynolds, A. Melnikov, N. Kliushkina, C. Araya, S. Yan, and O. Reblitz-Richardson, "Captum: A unified and generic model interpretability library for pytorch," 2020.

[30] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," 2017.

[31] "Interpreting bert models using captum," https://captum.ai/tutorials/Bert_SQUAD_Interpret, Captum AI, 2024, accessed: 2023-04-30.