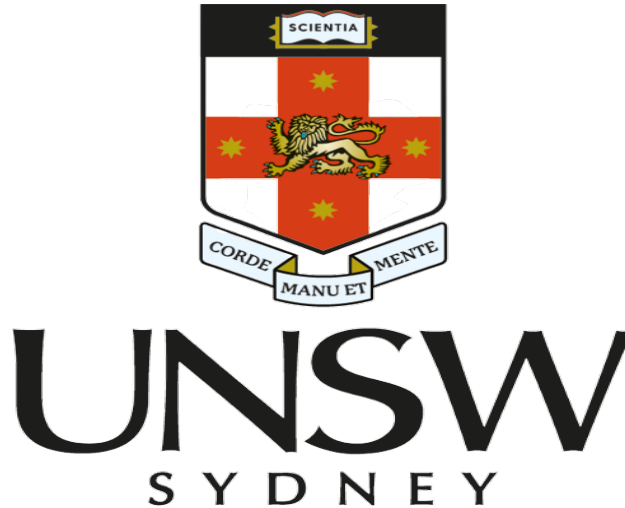


# Final Report



Course code: COMP9900

Course title: Computer Science Project

Project Title: Comparative Analysis of Fine-Tuning Techniques for Large Language Models on Domain-Specific Tasks

Yasemin Silen

[z5398552@ad.unsw.edu.au](mailto:z5398552@ad.unsw.edu.au)

z5398552

Developer

## Content

|   |    |
|---|----|
| A. Installation Manual .....                                  | 1  |
| B. System Architecture Diagram .....                          | 2  |
| C. Design Justifications.....                                 | 3  |
| 1. Changes in project requirements and design iterations..... | 3  |
| 1.1 Task .....  | 3  |
| 1.2 Model .....   | 5  |
| 1.3 Dataset .....   | 6  |
| 1.4 Fine tuning techniques .....                              | 7  |
| 2 Implementation details .....                                | 10 |
| 2.1 Data preparation & preprocessing .....                    | 10 |
| 2.1.1 Dataset Selection .....                                 | 10 |
| 2.1.2 Preprocessing method .....                              | 11 |
| 2.2 Model .....   | 14 |
| 2.2.1 Data loading and pre-processing .....                   | 14 |
| 2.2.2 Model fine-tuning settings .....                        | 14 |
| 2.3 Fine-tuning techs .....                                   | 17 |
| 2.3.1 LoRA .....  | 17 |
| 2.3.2 Adapter .....   | 18 |
| 2.3.3 Prefix .....  | 19 |
| D. User-Driven Evaluation of Solution .....                   | 20 |
| 1. Based on GPT2 .....  | 20 |
| 1.1 Model argument for training .....                         | 20 |
| 1.2 Fine-tuning technique 1: LoRA .....                       | 20 |
| 1.3 Fine-tuning technique 2: Adapter Tuning.....              | 20 |
| 1.4 Fine-tuning technique 3: Prefix Tuning .....              | 20 |

|  |    |
|--|----|
| 1.5 Results analysis.....                          | 21 |
| 1.5.1 Evaluation metrics & confusion matrix.....   | 21 |
| 1.5.2 Learning rate .....                          | 25 |
| 1.5.3 Training and validation loss .....           | 27 |
| 1.5.4 training time&computing resources usage..... | 30 |
| 1.6 Conclusion.....                                | 31 |
| 2. FinBERT Part Results .....                      | 32 |
| 2.1 FinBERT Model .....                            | 32 |
| 2.2 Different Fine-tuning Techniques .....         | 32 |
| 2.3 Fine-tuning Results Parameters & Plots .....   | 33 |
| 2.4 Analysis for FinBERT Model.....                | 36 |
| 2.5 Learning Rate analysis .....                   | 38 |
| 2.5.1 Learning Rate – LoRA .....                   | 39 |
| 2.5.2 Learning Rate - Adapter.....                 | 39 |
| 2.5.3 Learning Rate – Prefix .....                 | 39 |
| 2.6 Training and Validation Loss .....             | 40 |
| 2.6.1 Training & Validation Loss – LoRA .....      | 41 |
| 2.6.2 Training & Validation Loss – Adapter .....   | 41 |
| 2.6.3 Training & Validation Loss – Prefix .....    | 42 |
| 2.7 Computing Resource Usage .....                 | 42 |
| 2.8 Conclusion.....                                | 43 |
| E. limitations & Future work.....                  | 44 |
| 1. Limitations.....                                | 44 |
| 1.1 Dataset limitation .....                       | 44 |
| 1.1.1 Data size Limitation.....                    | 44 |
| 1.1.2 Data quality limitation .....                | 45 |
| 1.1.3 Data comprehensiveness limitation.....       | 45 |

|  |    |
|--|----|
| 1.2 Model Limitation.....  | 45 |
| 1.2.1 FinBERT limitation.....  | 45 |
| 1.2.2 GPT-2 limitation.....  | 46 |
| 1.3 Evaluation Limitation .....  | 46 |
| 1.3.1 Comprehensiveness of the evaluation .....  | 46 |
| 2. Future work .....   | 47 |
| 2.1 Dataset extension .....  | 47 |
| 2.1.1 Collect more data .....  | 47 |
| 2.1.2 Perform better data preprocessing .....  | 47 |
| 2.1.3 Introduce diverse data .....   | 47 |
| 2.2 Model Tuning.....  | 47 |
| 2.2.1 Explore more models.....   | 47 |
| 2.2.2 Experiment with different fine-tuning techniques for a specific model .....                                | 48 |
| 2.2.3 Adjust model hyperparameters to adapt to different datasets and explore the best training combination..... | 48 |
| 2.3 Efficiency Improvement .....   | 48 |
| 2.3.1 Simplify model .....   | 48 |
| 2.3.2 Apply distributed training.....  | 48 |
| 2.3.3 Use more computing resources .....   | 48 |
| 2.4 Evaluation Methods Enhancement .....   | 49 |
| 2.4.1 Introduce more evaluation metrics .....  | 49 |
| 2.4.2 Improve the generalization ability of the model.....   | 49 |
| 2.4.3 Improve the robustness of the model .....  | 49 |
| 2.4.4 Experience more practical evaluation.....  | 49 |
| 2.5 Domain Adaptability Improvement .....  | 50 |
| 2.5.1 Cross domain adaptation .....  | 50 |

|                                  |    |
|----------------------------------|----|
| 2.5.2 Adaptive learning .....    | 50 |
| 3. Conclusion .....              | 50 |
| F. Engineering Practices .....   | 50 |
| 1.Experimental Environment ..... | 50 |
| 2. Libraries and Tools.....      | 51 |
| 3. Data Preparation .....        | 51 |
| 4. Evaluation Metrics .....      | 52 |
| 5. Resource Monitoring.....      | 53 |
| References .....                 | 54 |

## A. Installation Manual

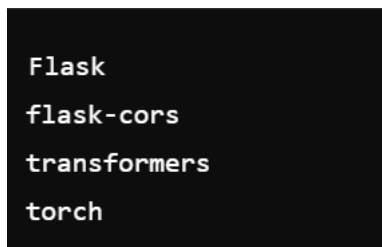
The user interface has a simple design where the user can enter the text, they want to be analyzed in a text box and then click a button to submit it. The front-end interface sends the text entered by the user to the backend. The backend service processes this text, uses the model for sentiment analysis, and returns the result to the frontend. The front-end interface displays this result to the user.

The user simply enters the text and clicks a button, and immediately sees the results of the sentiment analysis. These results include the sentiment tendency of the text (e.g., positive, negative, or neutral) and its confidence score. With these steps, we created an easy-to-use sentiment analysis tool where users can easily enter text and get accurate sentiment analysis results. This process not only enhances the user experience, but also demonstrates the functionality of sentiment analysis techniques in real-world applications.

### 1. Install the necessary packages

#### **Flask:**

*Install Flask and related packages:*



#### **React :**

*Install Node.js and npm:*

Download and install Node.js from the Node.js website, npm will be installed along with Node.js.

*Install all the needed packages in package.json:*

#### **npm install**

*Create a React application:*

If you haven't created a React application yet, you can use create-react-app:

***npx create-react-app app***

*Install necessary npm packages:*

Navigate to the React project directory and run the following command to install Axios:

***npm install axios***

## **2. Running the Project**

Running the Backend:

In the terminal, run the following command to start the Flask application:

***python app.py***

Running the Frontend:

In the React project directory, run the following command to start the React application:

***npm start***

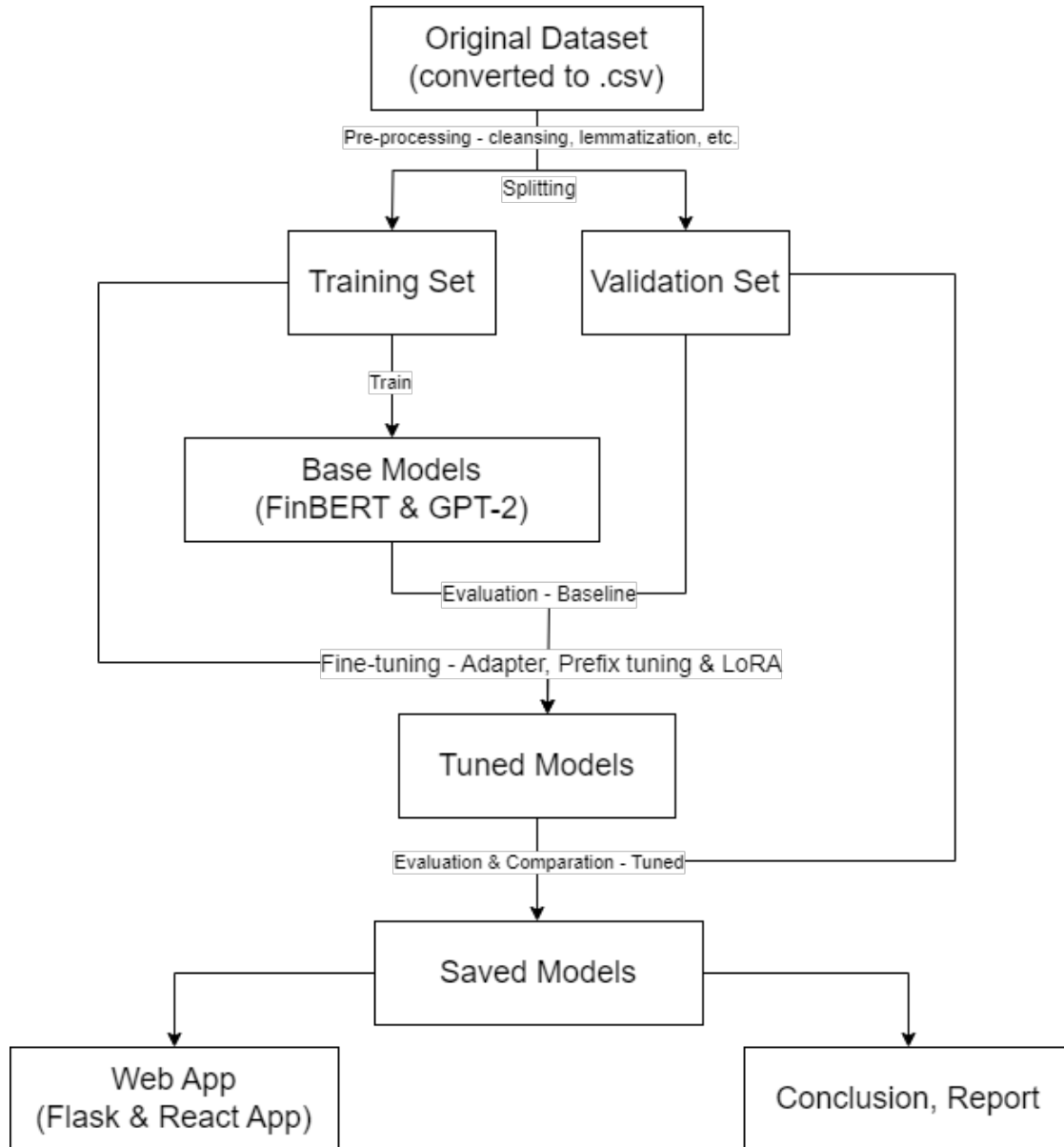
By following these steps, users can run the entire project locally and achieve the text sentiment analysis functionality.

Visiting the correct route:

***`http://localhost:5000/analyze`***

## **B. System Architecture Diagram**

System architecture diagram



*Figure 1. System Architecture Diagram*

## C. Design Justifications

### 1. Changes in project requirements and design iterations

#### 1.1 Task

##### Initial design

At the beginning of the project, we had considered choosing a regression type of task, such as training a model through a stock market dataset to try and get the model to perform a stock prediction task. The task itself was very appealing to us, and it has a lot of use in real life.



However, after we dug deeper into the LLM and NLP task itself, we found that adapting the LLM to the regression task would require a drastic adjustment to the model structure; According to (J Devlin et al., 2018) and (Lagler,K et al., 2013) most pre-trained language models (e.g., BERT, GPT, etc.) were originally designed for classification or sequence-to-sequence tasks, and their output layers are usually for discrete labels. To adapt to regression tasks (predicting continuous values), it is usually necessary to modify the output layer of the model, e.g., to output a continuous value instead of a categorical probability distribution. And considering that clients demand generalisation capabilities from models, however, generalisation capabilities for regression tasks are usually more difficult to achieve than for classification tasks, especially if the range or distribution of the target variable is significantly different between the training and test sets.

## Design iterations

So, after considering the difficulties mentioned above, and the skills and resources we actually had at our disposal, we decided to go for a more conventional classification task. The benefits of doing so are:

1. **Explicit evaluation criteria:** The first reason we chose the text classification task in the first place is because scoring metrics such as accuracy, precision, recall, and F1 scores directly illustrate the problems that the model or the data may have during the training phase. These metrics provide a more intuitive picture of the model's performance, whereas the evaluation metrics for the regression task (e.g., mean squared error or absolute error) may be less intuitive, and our users may encounter more technical hurdles when trying to derive key information from the metrics.
2. **Simplified data processing:** Text classification usually involves classifying textual data into different categories, such as positive and negative emotions. These categories are discrete, so we can use a classification model to predict which category the text belongs to. Regression tasks, on the other hand, involve predicting continuous output values, such as stock prices or sales. In regression, we apply more sophisticated techniques of data normalization or normalization so that the model can handle continuous target variables. This would take more unnecessary effort and would also affect our subsequent ability to determine the cause of the model's performance problems.
3. **Broader application scenarios:** Text classification tasks such as sentiment analysis and topic tagging generally have a wider range of application scenarios and business value. This

is one of the main reasons why we chose tasks in the financial domain. This makes it easier for our project results to be accepted by customers and to get project support and funding.

4. **Higher error tolerance:** In some cases, classification errors may be more acceptable than prediction errors in regression tasks. Due to our equipment, the dataset we have chosen will also not be particularly large, and the number of training rounds will be kept within certain limits, which may result in the model not accurately predicting all task classifications. In the sentiment analysis task, judging sentiment as neutral rather than mildly positive most of the time does not have a significant impact on the final decision.
5. **Model training and adjustment:** Classification models are often easier to implement and adapt. Using off-the-shelf NLP models and fine-tuning techniques, it is relatively easy to adapt models to classification tasks. For regression tasks, on the other hand, more customised tuning may be required, such as the choice of loss function and the design of the output layer.
6. **Resource efficiency:** In some cases, classification tasks may be more resource efficient. In particular, when using pre-trained models for migration learning, adapting the model to a classification task typically requires fewer resources than adapting a complex regression task.

## *1.2 Model*

### **Initial design**

Our first idea was to look for state of the art LLMs like Llama 3, GPT3 etc. However, we encountered some problems when trying to learn and apply new models in Sprint 1. The latest models such as GPT-3 or LLaMA 3, these two models usually have a huge number of parameters, which means that very powerful computational resources are needed for training and inference. For example, according to (Floridi, L. et al., 2020), GPT-3 has 175 billion parameters, which is extremely demanding on GPU memory and processing power. Without access to high-end GPUs or large-scale computational resources, it is impractical to use these models for experimentation and development.

And we also encountered problems with accessibility and licensing. The latest commercial models such as GPT-3, which requires access through an API, are not open source. More sophisticated technical support is required to understand these models. Whereas older models such as GPT-2 and FinBERT have wider community support and more implementation examples, making problem solving and learning easier.

## **Design iterations**

Therefore, according to (Araci, D., 2019), considering the factors of practicality, cost, resources and technical support, the choice of FinBERT and GPT-2 may be more in line with the actual needs and realistic conditions of the project. Although these models may not be able to compare with the latest models in terms of performance, they have obvious advantages in terms of resource and cost efficiency, ease of use and community support.

Of course, in Sprint 1 we also tried some other models, such as FinBERT, BERT, T5, GPT2 and so on. Due to the requirements of the project, we need to choose models with different structures, and finally we chose FinBERT, which has been fine-tuned in the financial domain, and GPT2, which has a different structure from it.

FinBERT is great at understanding text in context, considering both the words before and after a given word. This makes it perfect for sentiment analysis, where understanding the whole sentence is crucial. It's also a well-validated model that performs exceptionally in many NLP tasks.

Choosing the GPT-2 was more like choosing a comparative test with FinBERT for us. Theoretically FinBERT has been pre-trained to be more suitable for sentiment analysis tasks. Then we compare it with GPT-2 to see how well the fine-tuned GPT-2 performs on the sentiment analysis task. This helps us to gain a deeper understanding of the respective strengths of the differently structured models

### *1.3 Dataset*

#### **Initial design**

At Sprint1, we considered selecting data from the healthcare domain for the sentiment analysis task, but healthcare data often contains patient's personal health information and private data. This type of data is subject to strict data protection regulations, so it is often difficult to obtain a large amount of relevant data for model training and learning. This limits the amount or type of data that can be used for training and may also increase the complexity of data processing. In addition, medical text data may suffer from significant category imbalances, such as certain mood or symptom descriptions being more common than others. This may cause model training to be biased toward frequently occurring categories, which may affect the fairness and

accuracy of the model.

## **Design iterations**

Finally, considering the ease of data accessibility and data diversity, we chose a dataset for the financial domain task. Data sources in the financial domain are diverse, including but not limited to news articles, stock market commentaries, financial report analyses, and social media content. These data are not only voluminous and frequently updated, providing a rich corpus for the model to learn from, which helps to improve the model's ability to perceive current market sentiment and trends.

In sprint2, we found problems with the initial data preprocessing approach, such as inconsistent labelling leading to poor model training. This prompted us to adjust our data preprocessing strategy, implement a standardised preprocessing process, and improve the data quality, which resulted in improved model training efficiency and accuracy.

### *1.4 Fine tuning techniques*

#### **Initial design**

Initially we intend to apply three fine-tuning techniques, LoRA, Adapter and Prompt tuning.

#### **1. LoRA**

According to (Hu, Edward J., et al., 2021), the LoRA technique stands out for its parametric efficiency, by adding a low-rank matrix to the weight matrix of the pre-trained model, allowing for efficient task-specific fine-tuning while maintaining pre-trained knowledge, and is particularly suited to models with large parameters, and therefore efficiently controlling resource usage and computational cost.

#### **2. Adapter**

According to (He, Ruidan, et al., 2021), the Adapter technique allows models to be adapted to specific downstream tasks by introducing small trainable modules while keeping most of the pre-training weights unchanged, which allows the model to retain generalised performance on a wide range of data while enhancing the understanding of specific datasets such as financial texts.

#### **3. Prompt**

According to (Lester, B., Al-Rfou, R. and Constant, N., 2021.), prompt Tuning can effectively

manipulate and guide the behaviour of large language models through well-designed prompts, and this fine-tuning strategy was chosen based on the following considerations:

1. **Low-intervention fine-tuning:** a central strength of Prompt Tuning is its minimal intervention in the structure and parameters of the model. By changing only, the cued portion of the inputs to the model without directly adjusting the model's weights, this approach allows the model to be guided to generate the outputs required for a particular task while maintaining the model's original robust capabilities.
2. **Rapid Deployment and Adaptability:** Prompt Tuning allows for faster deployment and adaptation than traditional full fine-tuning. Since no large-scale training of the model's underlying architecture is required, this approach allows for rapid adaptation to new tasks or datasets.
3. **Ability to effectively leverage pre-trained models:** Prompt Tuning relies on the linguistic patterns and knowledge already learned by the pre-trained models, and it is possible to motivate the models to exhibit complex behaviours through simple input prompts. In the financial domain, where many concepts and contexts require complex background knowledge, leveraging the rich knowledge of pre-trained models can effectively improve the models' understanding of specific financial terms and contexts without sacrificing performance.

### Pros And Cons Of Fine-Tuning Techniques

|   | Fine-Tuning Technique  | Advantages   | Disadvantages   |
|---|------------------------|--|---|
| 1 | Full Fine-Tuning       | Highest performance;<br>Flexibility for various tasks                  | Resource-intensive;<br>High data requirements                                   |
| 2 | LoRA                   | Resource-efficient;<br>Effective for large models                      | Slightly lower performance than full fine-tuning                                |
| 3 | Prefix Tuning          | Highly efficient;<br>Easily adaptable to different tasks               | Performance may fluctuate on complex tasks                                      |
| 4 | Prompt Tuning          | Minimal parameter updates; Quick adaptation to new tasks               | May not fully utilize model capacity;<br>Performance dependent on prompt design |
| 5 | Knowledge Distillation | Model compression;<br>Maintains good performance with lower complexity | Complex process;<br>Potential performance loss compared to teacher model        |

*Figure 2. Pro and Cons of Fine-Tuning Techs*

### Design iterations

However, after applying prompt tuning in Sprint 1 we found that the accuracy of both models was very low, even well below the baseline capability of the models. Although this approach is effective in some scenarios, it is not effective when we use GPT-2 and FinBERT for the task of sentiment analysis in the financial domain, which may be due to the following reasons.

#### High data dependency:

Prompt tuning usually relies on high quality and high relevance cues to effectively guide the model's output. In sentiment analysis in the financial domain, the need to accurately capture terminology and subtle sentiment differences may be beyond the scope of guidance that simple prompts can provide.

#### Model sensitivity to inputs:

Models like GPT-2 and FinBERT, while excellent at natural language processing, may not be sensitive enough to specific types of cues, especially when dealing with complex or ambiguous

domain-specific sentiment. If the inputs do not precisely match the context in which the model was trained, their effectiveness may be greatly reduced.

### **Domain adaptation issues:**

Prompt tuning works better in some generalized tasks, but may not be sufficient in tasks that require deeper understanding of domain-specific knowledge (e.g., specific sentiment expressions in financial markets). Our project does not require us to disassemble the model structure, but performing effective prompt tuning requires some adjustments to the model structure. We needed to create the appropriate number of financial statements to be input as prompters into specific layers of the model so that the model could learn quickly. The method we used just set several virtual tokens, which did not seem to give the model the right inputs, the complexity and specificity of the financial text may require a deeper understanding and tuning of the model, and a single prompt may be difficult to guide the model to good performance. The model was not trained as well as it should have been, so we intend to replace prompt tuning with prefix tuning (Li, X.L. and Liang, P., 2021.). This approach not only provides more flexibility and adaptability, but also enhances the model's understanding of and response to finance-specific contexts without destroying the model's original capabilities.

## **2 Implementation details**

In this section, we describe in detail how to fine-tune FinBERT and GPT2 models efficiently in a limited resource environment and apply them to financial text analysis. We introduce dataset preparation and preprocessing, model architecture, fine-tuning process, and selection of evaluation metrics.

### *2.1 Data preparation & preprocessing*

#### **2.1.1 Dataset Selection**

In this project, we focus on the field of financial text sentiment analysis, so there are some requirements for the dataset we choose. First, this dataset should contain at least two types of information. One column is financial information in text format, which can be macroeconomic financial news or financial market information, such as news or financial market information classified as financial on news websites. Second, the dataset should have distinct emotional characteristics as much as possible. Since this experiment focuses on fine-tuning large models in the field of financial text analysis using efficient fine-tuning methods to improve the model's ability to analyze emotions in this field, texts with distinct emotions are very necessary. Third, the dataset should be divided into three categories, namely negative, neutral and positive. For financial trading market information, it can be divided into bear market, neutral and bull market,

which is conducive to the design of labels in the subsequent training model process.

After selection, we selected two suitable datasets, FinancialPhraseBank and twitter-financial-news-sentiment.

### **1. FinancialPhraseBank**

FinancialPhraseBank contains four sub-datasets. Each dataset has two columns, one is financial news, and the other is sentiment classification which contains negative, neutral and positive. The classification criteria of these four datasets are that each piece of financial news will be judged by a group of financial experts. If more than 50% of them agree that the news has the same sentiment, it will be classified as the 50Agree sub-dataset. Similarly, if 100% of the experts agree that a piece of news has the same sentiment, it will be classified as the AllAgree dataset.

That is to say, the other three datasets are all subsets of 50Agree, so we chose the 50Agree dataset. The reason is that this dataset contains the most data, about 4,000. And some of the sentences that 50% of the experts agree on are sentimentally ambiguous. Compared with sentences whose sentiments can be determined 100% by humans, it is more meaningful to use a large model for analysis.

### **2. Twitter-financial-news-sentiment**

Twitter-financial-news-sentiment contains information related to the stock market. This dataset contains two columns of information, the first column contains stock-related news, and the other is sentiment classification. He divides sentiment into three categories: bearish, neutral and bullish.

This dataset has been divided into two subsets, training set and validation set, with a split ratio of 80:20.

#### **2.1.2 Preprocessing method**

In this section, we applied a variety of preprocessing methods to ensure the smooth progress of training and improve training efficiency.

##### **1. Text cleaning**

Based on the observation of the datasets, we mainly used three methods in text data cleaning. First, we used regular expressions to remove all HTML tags. For sentiment analysis, HTML tags are meaningless. Second, use regular expressions to remove all non-letter characters, leaving only letters and spaces. For sentiment analysis, non-letter characters do not affect the emotion expressed by the sentence. Spaces cannot be deleted because the presence or absence of spaces significantly affects the meaning of the statement. Finally, all letters are converted to lowercase to ensure textual consistency and help the model better understand and process these



words.

To prevent NaN values from affecting the training process, all data rows containing real values are removed during preprocessing.

## 2. Stopwords removing

In sentiment analysis, there are some words that do not affect the meaning of the sentence. For example, some copulas, such as am, is, are, etc. The steps to remove stop words are to first split the text into words based on spaces, then filter out all stop words, and then form a string of words that have been filtered out. Here, the stop word list provided by the NLTK library is used.

## 3. Tokenization

During the preprocessing process, tokenizer is also used for word segmentation. Word segmentation divides sentence into word lists, which makes it easier for the model to understand the sentences. In our preprocessing, RegexpTokenizer is used.

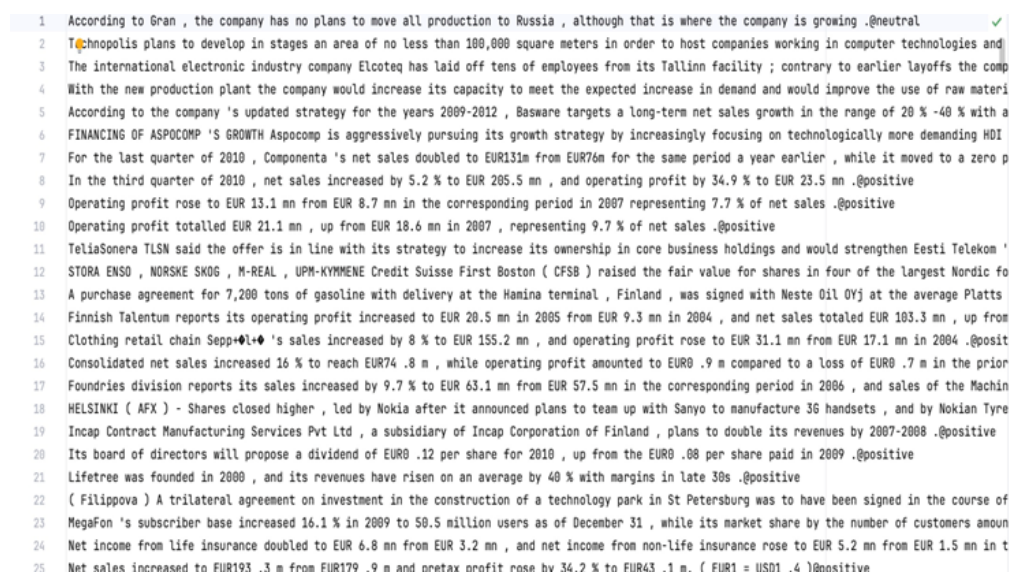
## 4. Word lemmatization

Lemmatization is the process of converting a word into its base or root form. This step is crucial in natural language processing as it helps to simplify different forms of a word into a single item, thereby improving the consistency and accuracy of the analysis. We use the textblob library for lemmatization, which provides an easy-to-use interface for this task.

## 5. Labelling

Labeling & Mapping is a critical step in natural language processing and machine learning projects. First, the labeling scheme was determined, with negative comments marked as 0, neutral comments marked as 1, and positive comments marked as 2. This is very important. Without labelling, the model cannot understand what kind of sentence sentiment negative, what kind of sentence sentiment is neutral, and what kind of sentence sentiment is positive.

## 6. Screenshots before and after preprocessing



```
1 According to Gran , the company has no plans to move all production to Russia , although that is where the company is growing .@neutral ✓
2 Technopolis plans to develop in stages an area of no less than 100,000 square meters in order to host companies working in computer technologies and
3 The international electronic industry company Elcoteq has laid off tens of employees from its Tallinn facility ; contrary to earlier layoffs the comp
4 With the new production plant the company would increase its capacity to meet the expected increase in demand and would improve the use of raw materi
5 According to the company 's updated strategy for the years 2009-2012 , Basware targets a long-term net sales growth in the range of 20 % -40 % with a
6 FINANCING OF ASPOCOMP 'S GROWTH Aspocomp is aggressively pursuing its growth strategy by increasingly focusing on technologically more demanding HDI
7 For the last quarter of 2010 , Componenta 's net sales doubled to EUR131m from EUR76m for the same period a year earlier , while it moved to a zero p
8 In the third quarter of 2010 , net sales increased by 5.2 % to EUR 205.5 mn , and operating profit by 34.9 % to EUR 23.5 mn .@positive
9 Operating profit rose to EUR 13.1 mn from EUR 8.7 mn in the corresponding period in 2007 representing 7.7 % of net sales .@positive
10 Operating profit totalled EUR 21.1 mn , up from EUR 18.6 mn in 2007 , representing 9.7 % of net sales .@positive
11 TeliaSonera TSLM said the offer is in line with its strategy to increase its ownership in core business holdings and would strengthen Eesti Telekom '
12 STORA ENSO , NORSKE SKOG , M-REAL , UPM-KYMMENE Credit Suisse First Boston ( CFSB ) raised the fair value for shares in four of the largest Nordic fo
13 A purchase agreement for 7,200 tons of gasoline with delivery at the Hamina terminal , Finland , was signed with Neste Oil Oyj at the average Platts
14 Finnish Talentum reports its operating profit increased to EUR 20.5 mn in 2005 from EUR 9.3 mn in 2004 , and net sales totaled EUR 103.3 mn , up from
15 Clothing retail chain Seppälä 's sales increased by 8 % to EUR 155.2 mn , and operating profit rose to EUR 31.1 mn from EUR 17.1 mn in 2004 .@posit
16 Consolidated net sales increased 16 % to reach EUR74 .8 m , while operating profit amounted to EUR0 .9 m compared to a loss of EUR0 .7 m in the prior
17 Foundries division reports its sales increased by 9.7 % to EUR 63.1 mn from EUR 57.5 mn in the corresponding period in 2006 , and sales of the Machin
18 HELSINKI ( AFX ) - Shares closed higher , led by Nokia after it announced plans to team up with Sanyo to manufacture 3G handsets , and by Nokian Tyre
19 Incap Contract Manufacturing Services Pvt Ltd , a subsidiary of Incap Corporation of Finland , plans to double its revenues by 2007-2008 .@positive
20 Its board of directors will propose a dividend of EUR0 .12 per share for 2010 , up from the EUR0 .08 per share paid in 2009 .@positive
21 Lifetree was founded in 2000 , and its revenues have risen on an average by 40 % with margins in late 30s .@positive
22 ( Filippova ) A trilateral agreement on investment in the construction of a technology park in St Petersburg was to have been signed in the course of
23 MegaFon 's subscriber base increased 16.1 % in 2009 to 50.5 million users as of December 31 , while its market share by the number of customers among
24 Net income from life insurance doubled to EUR 6.8 mn from EUR 3.2 mn , and net income from non-life insurance rose to EUR 5.2 mn from EUR 1.5 mn in t
25 Net sales increased to EUR193 .3 m from EUR179 .9 m and netax profit rose by 34.2 % to EUR43 .1 m . ( EUR1 = USD1 .4 )@negative
```

Figure 3. FinancialPhraseBank before preprocessing

| text,label  |  |
|---|--|
| according gran company no plans move production russia although company growing,1   |  |
| technopolis plans develop stages area no less square meters order host companies working computer technologies telecommunications statement said,1      |  |
| international electronic industry company elcoteq laid tens employees tallinn facility contrary earlier layoffs company contracted ranks office worke   |  |
| new production plant company would increase capacity meet expected increase demand would improve use raw materials therefore increase production prof   |  |
| according company updated strategy years basware targets longterm net sales growth range operating profit margin net sales,2                            |  |
| financing aspo comp growth aspo comp aggressively pursuing growth strategy increasingly focusing technologically demanding hdi printed circuit boards p |  |
| last quarter componenta net sales doubled eurm eurm period year earlier moved zero pretax profit pretax loss eurm,2                                     |  |
| third quarter net sales increased eur mn operating profit eur mn,2  |  |
| operating profit rose eur mn eur mn corresponding period representing net sales,2   |  |
| operating profit totalled eur mn eur mn representing net sales,2  |  |
| teliasonera tln said offer line strategy increase ownership core business holdings would strengthen eesti telekom offering customers,2                  |  |
| stora enso norske skog mreal upmkymene credit suisse first boston cfsb raised fair value shares four largest nordic forestry groups,2                   |  |
| purchase agreement tons gasoline delivery hamina terminal finland signed neste oil oyj average platts index september plus eight us dollars per month   |  |
| finnish talentum reports operating profit increased eur mn eur mn net sales totalled eur mn eur mn,2  |  |
| clothing retail chain sepl sales increased eur mn operating profit rose eur mn eur mn,2   |  |
| consolidated net sales increased reach eur operating profit amounted eur compared loss eur prior year period,2  |  |
| foundries division reports sales increased eur mn eur mn corresponding period sales machine shop division increased eur mn eur mn corresponding perio   |  |
| helsinki afx shares closed higher led nokia announced plans team sanyo manufacture g handsets nokian tyres fourthquarter earnings report beat analyst   |  |
| incap contract manufacturing services pvt ltd subsidiary incap corporation finland plans double revenues,2  |  |
| board directors propose dividend eur per share eur per share paid,2   |  |
| lifetree founded revenues risen average margins late,2  |  |

Figure 4. FinancialPhraseBank after preprocessing

|    | text,label   |
|----|--|
| 1  |  |
| 2  | \$BYND - JPMorgan reels in expectations on Beyond Meat <a href="https://t.co/bd0xbF6jKT">https://t.co/bd0xbF6jKT</a> ,0  |
| 3  | \$CCL \$RCL - Nomura points to bookings weakness at Carnival and Royal Caribbean <a href="https://t.co/y6jpt2Re03">https://t.co/y6jpt2Re03</a> ,0              |
| 4  | "\$CX - Cemex cut at Credit Suisse, J.P. Morgan on weak building outlook <a href="https://t.co/KN1g4AWFIb">https://t.co/KN1g4AWFIb</a> ",0                     |
| 5  | \$ESS: BTIG Research cuts to Neutral <a href="https://t.co/MCyfTsXc2N">https://t.co/MCyfTsXc2N</a> ,0  |
| 6  | \$FNKO - Funko slides after Piper Jaffray PT cut <a href="https://t.co/z37IJmCQzB">https://t.co/z37IJmCQzB</a> ,0  |
| 7  | \$FTI - TechnipFMC downgraded at Berenberg but called Top Pick at Deutsche Bank <a href="https://t.co/XKcPDilIuU">https://t.co/XKcPDilIuU</a> ,0               |
| 8  | \$GM - GM loses a bull <a href="https://t.co/tdUf65HbXy">https://t.co/tdUf65HbXy</a> ,0  |
| 9  | \$GM: Deutsche Bank cuts to Hold <a href="https://t.co/7Fv1ZiFZBS">https://t.co/7Fv1ZiFZBS</a> ,0  |
| 10 | \$GTT: Cowen cuts to Market Perform,0  |
| 11 | \$HNHAF \$HNHPD \$AAPL - Trendforce cuts iPhone estimate after Foxconn delay <a href="https://t.co/rlnEwzlzS">https://t.co/rlnEwzlzS</a> ,0                    |
| 12 | \$HOG - Moody's warns on Harley-Davidson <a href="https://t.co/LurHBEadeU">https://t.co/LurHBEadeU</a> ,0  |
| 13 | "\$HXL - Citing aero ties, Wells slashes PT on Hexcel <a href="https://t.co/wU5P2i8WBU">https://t.co/wU5P2i8WBU</a> ",0  |
| 14 | \$I - Intelsat cut to Market Perform at Raymond James <a href="https://t.co/YsvsMSQRIB">https://t.co/YsvsMSQRIB</a> ,0   |
| 15 | \$KRG: Compass Point cuts to Sell <a href="https://t.co/MCyfTsXc2N">https://t.co/MCyfTsXc2N</a> ,0   |
| 16 | \$LK - Muddy Waters goes short Luckin Coffee <a href="https://t.co/8yrbwAjLKG">https://t.co/8yrbwAjLKG</a> ,0  |
| 17 | \$MANT - ManTech downgraded ahead of difficult comps <a href="https://t.co/mJleSrsFXJ">https://t.co/mJleSrsFXJ</a> ,0  |
| 18 | \$MDCO: Oppenheimer cuts to Perform,0  |
| 19 | \$MPLX \$MPC - MPLX cut at Credit Suisse on potential dilution from Marathon strategic review <a href="https://t.co/0BFQy4ZU6W">https://t.co/0BFQy4ZU6W</a> ,0 |
| 20 | \$MSGN - Imperial downgrades MSG Networks amid sports-free airwaves <a href="https://t.co/U12S6XNXw8">https://t.co/U12S6XNXw8</a> ,0                           |

Figure 5. Twitter-financial-news-sentiment before preprocessing

|    |  |
|----|--|
| 1  | text,label   |
| 2  | bynd jpmorgan reel expectation beyond meat,0                                   |
| 3  | ccl rcl nomura point booking weakness carnival royal caribbean,0               |
| 4  | cx cemex cut credit suisse jp morgan weak building outlook,0                   |
| 5  | es btig research cut neutral,0   |
| 6  | fnko funko slide piper jaffray pt cut,0  |
| 7  | fti technipfmc downgraded berenberg called top pick deutsche bank,0            |
| 8  | gm gm loses bull,0   |
| 9  | gm deutsche bank cut hold,0  |
| 10 | gtt cowen cut market perform,0   |
| 11 | hnhaf hnhpd aapl trendforce cut iphone estimate foxconn delay,0                |
| 12 | hog moody warns harleydavidson,0   |
| 13 | hxl citing aero tie well slash pt hexcel,0                                     |
| 14 | intelsat cut market perform raymond james,0                                    |
| 15 | krg compass point cut sell,0   |
| 16 | lk muddy water go short luckin coffee,0  |
| 17 | mant mantech downgraded ahead difficult comp,0                                 |
| 18 | mdco oppenheimer cut perform,0   |
| 19 | mplx mpc mplx cut credit suisse potential dilution marathon strategic review,0 |
| 20 | msgn imperial downgrade msg network amid sportsfree airwave,0                  |

Figure 6. Twitter-financial-news-sentiment after preprocessing

## 2.2 Model

### 2.2.1 Data loading and pre-processing

**1. Data reading:** Use Pandas library to read data from CSV files, this is because Pandas provides powerful data manipulation functions to quickly process and filter a large amount of data.

**2. Dataset Construction:** Use the Dataset.from\_pandas method of the datasets library to convert the Pandas DataFrame into Hugging Face's Dataset object, this conversion makes the data easier to be used by subsequent processing processes and models.

**3. Segmentation and encoding:** Use GPT-2's Segmenter (GPT2Tokenizer) and FinBERT's Segmenter (AutoTokenizer) to process the text data, including truncation and padding operations, to ensure that all the text is of the same length and to adapt to the needs of model training. Set the pad\_token of the disambiguator to eos\_token to adapt to the special requirements of GPT-2 and FinBERT models.

### 2.2.2 Model fine-tuning settings

**Fine-tuning Parameter Configuration:** use TrainingArguments to set training parameters such as training period, batch size, learning rate, etc., as well as evaluation and saving strategies. These parameters are set to optimise the model training process and keep the results optimal.

**Evaluation Metrics Definition:** Define a function compute\_metrics to calculate key metrics of model performance, including accuracy, precision, recall, and F1 score, which help evaluate

and compare the effects of different model configurations.

1. **Learning Rate Warmup:** the warm-up step is a technique for adjusting the learning rate, especially used at the beginning of training. At the beginning of training, the weights of the model are not yet fully optimised, and directly using a larger learning rate may lead to unstable model training or even dispersion. The warm-up step helps the model to gradually adapt to the training by gradually increasing the learning rate at the beginning of training from a relatively low value to the target learning rate, avoiding the problem of training instability caused by too high a learning rate.

The warm-up of the learning rate is achieved by setting the `warmup_steps` parameter. This means that in the first 10% of the total training steps, the learning rate will gradually increase from zero or a very low starting value to the set initial learning rate.

2. **Gradient Accumulation:** Gradient accumulation is another technique used to simulate the effect of training with larger batch sizes with limited hardware resources. Due to GPU memory constraints, it may not be feasible to directly increase the batch size. Gradient accumulation means running a configured number of steps without updating the model variables while accumulating the gradients of those steps, and then using the accumulated gradients to compute variable updates. Running a number of steps without updating any of the model variables is a logical way to split the sample batch into several small Batches. The sample batch used in each step is actually a small batch, while the samples in all these steps combined are actually the global batch. By not updating the variables in all these steps, the gradient is computed using the same model variables for all the small batches. This is mandatory behavior to ensure that the same gradients are computed and updated as if the global batch size were used.

The combined use of these two techniques not only helps to improve the training efficiency and stability of the model, but also maximises the training effect with limited computational resources. The warm-up step and gradient accumulation are very useful strategies for tuning the deep learning training process, especially for complex models and large-scale datasets.

### 3. Training and Callback

TrainerCallback is implemented to monitor various metrics during the training process, such as loss values and learning rates, as well as resource usage such as CPU and memory usage. This monitoring data is critical for optimising the model training environment and parameter tuning.

The ResourceMonitor callback class tracks resource usage during training, including CPU,

memory, and GPU usage, as well as the total time spent training, helping to identify possible performance bottlenecks.

#### 4. Results visualisation and analysis

**Loss and Evaluation Metrics Visualisation:** Plotting the training and validation losses, as well as the trends of each evaluation metric, these graphs help to visualise the progress of model training and performance performance.

**Confusion Matrix:** Generate and visualise confusion matrix to see in detail the model's predictive performance on each category, which is very helpful in understanding the model's strengths and weaknesses.

#### 5. Model Deployment and Usage

**Model saving and loading:** Save the model at the end of training for future loading and execution of prediction tasks.

**Prediction Function:** Implement a prediction function `predict_sentiment`, which uses the trained model to predict the sentiment of a new text, showing how to apply the trained model to a real business scenario.

**Save Resource Data:** Save resource usage data to JSON files, which provides data support for subsequent analysis and optimisation.

#### 6. Use of category weights

**Weight Calculation:** category weights are typically used to deal with category imbalances present in a dataset. This function generates a weight for each class based on the frequency of samples in each class, making a few classes more heavily weighted to offset their lower frequency of occurrence in the dataset.

**Weights application:** once set, these weights are converted into a `torch.tensor` object and moved to the appropriate device based on the available computing device (e.g., GPU) to be used during model training.

#### 7. Customising the loss function

**Loss function selection:** In our project, the `CrossEntropyLoss` is used, which is a common loss function used in classification tasks because of its ability to work directly with the probability distribution of categories and its built-in support for category weights.

**Loss function implementation:** a custom WeightedLossTrainer class was created by inheriting from the Trainer class. In this class, the compute\_loss method was overridden to use the cross-entropy loss with weights.

## 2.3 Fine-tuning techs

### 2.3.1 LoRA

**LoRA configuration:** configure LoRA through the PEFT library to modify the weights by inserting low-rank matrices in specific layers of the model, which improves the model's adaptability to specific tasks without significantly increasing the number of parameters.

**task\_type:** specifies the task type, I set it to TaskType.SEQ\_CLS, meaning that the configuration is intended for sequence classification tasks. Sequence classification tasks usually involve classifying an entire input sequence, such as sentiment analysis or intent recognition.

**r:** Indicates the rank of the low-rank matrix used in LoRA. The lower the rank, the fewer new parameters are added and the more computationally efficient it is, but the representational power may be limited. Here,  $r=32$  provides a balance aimed at enhancing the model capability without significantly increasing the number of parameters.

**lora\_alpha:** controls how well the low-rank matrix is fused with the original weight matrix. Here it is set to 64, which means that the learning rate of the LoRA layer is 64 times higher than the learning rate of the original layer, which helps to quickly adapt to new task characteristics and speeds up the fine-tuning process.

**lora\_dropout:** the dropout ratio applied after the LoRA layer, set to 0.1. This helps to prevent overfitting of the model and increases the model's ability to generalise across different data samples.

**bias:** Set whether or not to use bias in the LoRA layer, here set to 'none', means no bias is used. In some cases, removing bias can simplify the model and reduce the risk of overfitting.

**target\_modules:** Specify the model modules to which LoRA is to be applied. Here it is usually

set to the attention module of the model. The reason for choosing the attention module is that it plays a key role in handling dependencies in sequence data. By strengthening this component, the model's ability to capture intra-sequence relationships can be effectively improved. The model's ability to capture intra-sequence relationships can be effectively improved

### 2.3.2 Adapter

We use `AutoAdapterModel` to load the pre-trained model weights and configure the adapter layer, an approach that allows us to adapt the model specifically to the financial sentiment analysis task. The process of model initialization and adapter configuration is explained in detail below:

#### 1. Model Initialisation

**Load pre-trained models:** Use `AutoAdapterModel` to load weights from pre-trained models.

**Add Classification Head:** To enable the model to perform financial sentiment analysis tasks, we add a dedicated classification head (`add_classification_head`) to the model. This head is a fully connected layer that is used to map the model's output to the three sentiment categories (positive, neutral, and negative). This design allows the model to directly output sentiment predictions for each input text.

#### 2. Adapter Configuration

**Pfeiffer Configuration:** The adapter configuration approach was chosen to use Pfeiffer (Pfeiffer, Jonas, et al., 2021), which is an effective adapter architecture that enables fine-tuning of the model by introducing additional small network layers (adapter layers) while keeping most of the weights of the pre-trained model unchanged. The Pfeiffer Configuration puts special emphasis on keeping the original model architecture stability of the original model architecture, while capturing task-relevant features with a limited number of additional parameters.

**Adapter Layer Addition:** Adapter layers are added to each Transformer layer of the model. These adapter layers are relatively small but can learn the key features of the new task without destroying the original model weights.

**Training strategy:** during training, only the parameters of the adapter layer are updated and the other weights of the model remain unchanged. This strategy reduces the risk of overfitting and reduces the required training resources. At the same time, by training a different adapter layer for each task, the model can be applied more flexibly to many different tasks.

In this way, our model is optimized with relatively small adjustments based on pre-training to

specifically handle sentiment analysis tasks in the financial domain. This approach effectively balances training efficiency and model performance, allowing the model to quickly adapt to new tasks while maintaining its ability to generalize to large-scale data.

### 2.3.3 Prefix

#### 1. Key parameters of PrefixTuningConfig

**Number of virtual tokens:** defines the number of virtual tokens added to the input sequence front. Each virtual token is associated with a set of trainable parameters. These tokens serve as additional context, and the model uses them to adjust its internal state before processing the actual input sequence.

**Prefix Projection:** this Boolean parameter determines whether to apply a linear projection on the embedding of virtual tokens. If True, it allows to transform the dimensionality of the prefix embedding, potentially enhancing the model's ability to effectively integrate prefix information with the input data.

**Encoder Hidden Size:** refers to the dimension of the hidden layer of the encoder part of the model. This parameter is set to ensure that the prefix embedding is compatible with the internal architecture of the model, in particular matching the size of the hidden state within the model. This compatibility is critical to ensure that added prefixes can be seamlessly integrated into the model's forward propagation, affecting it in a way that matches the existing model architecture.

#### 2. Integration with model architecture

Prefix parameters are initially randomly initialised and then learned during training. As training progresses, these parameters are tuned to capture task-specific nuances that help the model perform better on the target task. By updating only these prefixes rather than the entire model, we significantly reduce computational costs and the risk of catastrophic forgetting, where the model loses its ability to perform previously trained tasks.

#### 3. Practical Implications

For sequence classification tasks, prefix fine-tuning helps direct the model to focus on the aspects of the input data that are most informative for classifying sequences into categories (e.g., emotions). This is particularly useful in adapting to specialised domains or situations with nuanced tasks, where the context provided by the prefix can lead to better performance than



standard fine-tuning methods.

## D. User-Driven Evaluation of Solution

### 1. Based on GPT2

#### *1.1 Model argument for training*

| parameter                   | value       |
|-----------------------------|-------------|
| <b>epochs</b>               | <b>3</b>    |
| Train batch size per device | 8           |
| Eval batch size per device  | 8           |
| Evaluation strategy         | steps       |
| Evaluation steps            | 100         |
| <b>Learning Rate</b>        | <b>2e-5</b> |
| Weight decay                | 0.01        |
| Best model metric           | accuracy    |
| Logging steps               | 100         |
| <b>Warm-up steps</b>        | <b>100</b>  |
| Gradient accumulation steps | 2           |

#### *1.2 Fine-tuning technique 1: LoRA*

| Parameter     | value  |
|---------------|--------|
| Rank          | 32     |
| Lora_alpha    | 64     |
| Dropout rate  | 0.1    |
| bias          | none   |
| Target module | c_attn |

#### *1.3 Fine-tuning technique 2: Adapter Tuning*

| Parameter        | Value    |
|------------------|----------|
| Adapter config   | Pfeiffer |
| Reduction factor | 16       |

#### *1.4 Fine-tuning technique 3: Prefix Tuning*

| Parameter | value   |
|-----------|---------|
| Task type | SEQ.CLS |

|                          |      |
|--------------------------|------|
| Number of virtual tokens | 10   |
| Prefix projection        | True |
| Encoder hidden size      | 768  |

Next, the results obtained from this project will be analyzed.

### 1.5 Results analysis

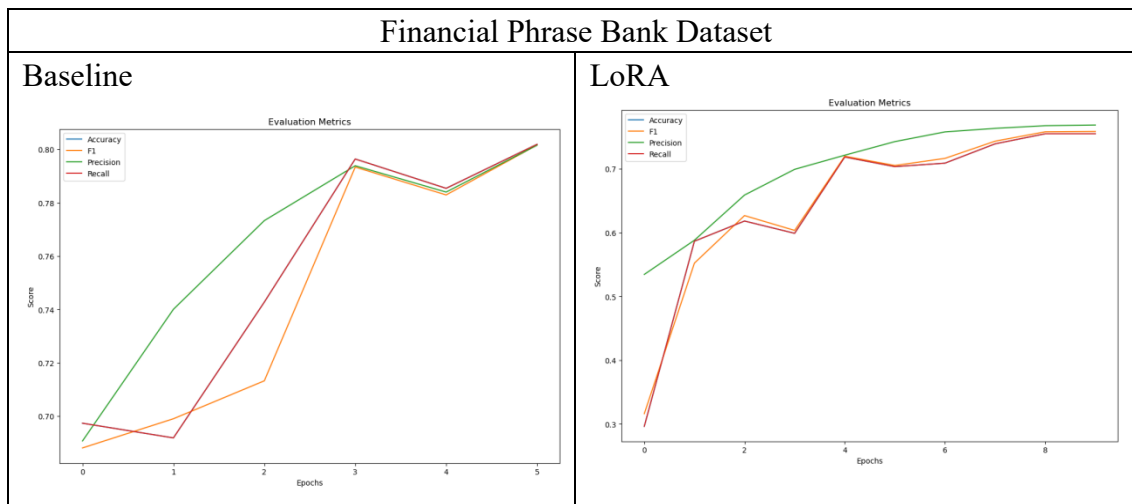
#### From the perspective of results

##### 1.5.1 Evaluation metrics & confusion matrix

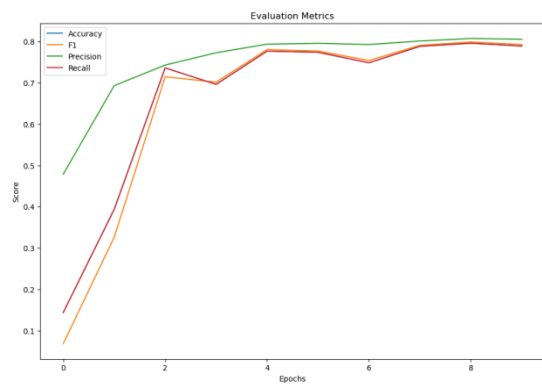
| Financial Phrase Bank Dataset |               |               |               |               |
|-------------------------------|---------------|---------------|---------------|---------------|
|                               | Accuracy      | Precision     | Recall        | F1-score      |
| Baseline                      | 0.7950        | 0.7921        | 0.7950        | 0.7926        |
| LoRA                          | 0.7916        | 0.7984        | 0.7916        | 0.7937        |
| AdapterTuning                 | <b>0.8088</b> | <b>0.8086</b> | <b>0.8088</b> | <b>0.8087</b> |
| prefixTuning                  | 0.7552        | 0.7626        | 0.7552        | 0.7576        |

| Tweet Financial Dataset |               |               |               |               |
|-------------------------|---------------|---------------|---------------|---------------|
|                         | Accuracy      | Precision     | Recall        | F1-score      |
| Baseline                | 0.8229        | 0.8247        | 0.8229        | 0.8225        |
| LoRA                    | 0.8258        | 0.8262        | 0.8258        | 0.8258        |
| AdapterTuning           | <b>0.8392</b> | <b>0.8427</b> | <b>0.8392</b> | <b>0.8406</b> |
| prefixTuning            | 0.8070        | 0.8099        | 0.8070        | 0.8083        |

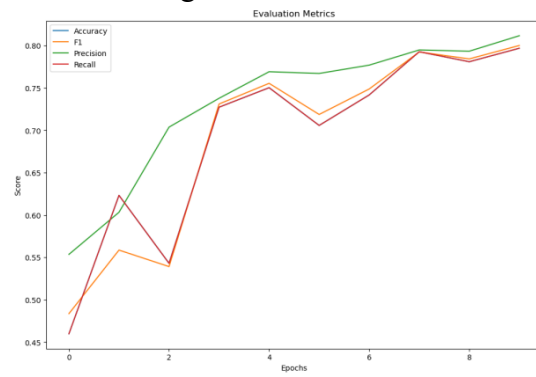
#### Evaluation Metrics



## adapter

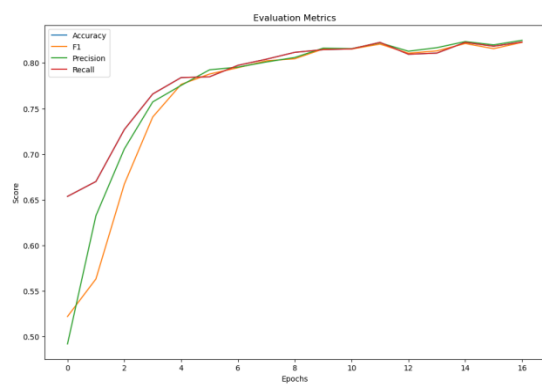


## Prefix-tuning

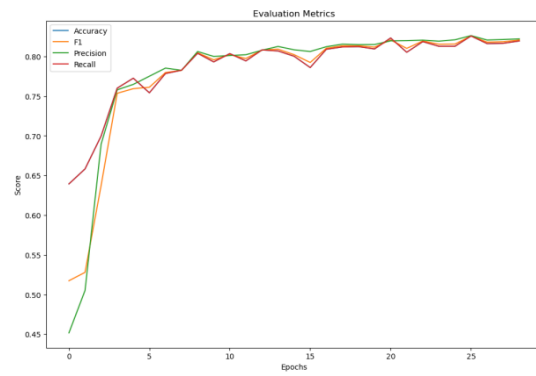


## Tweet Financial Dataset

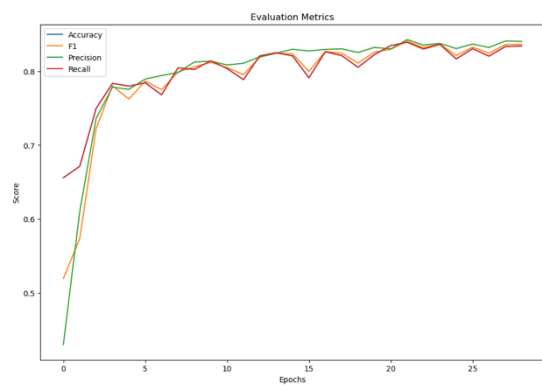
### Baseline



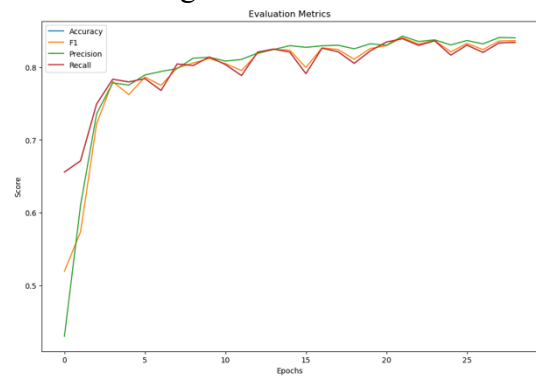
### LoRA



## adapter



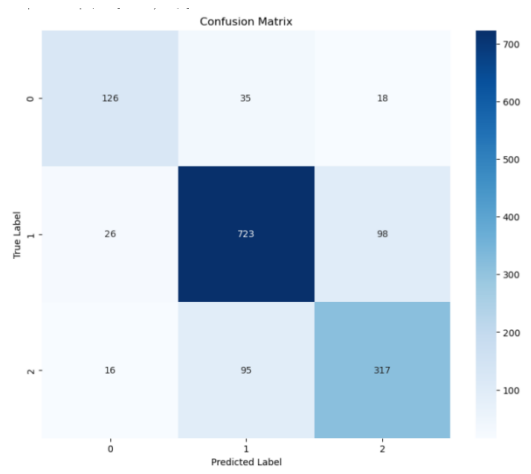
## Prefix-tuning



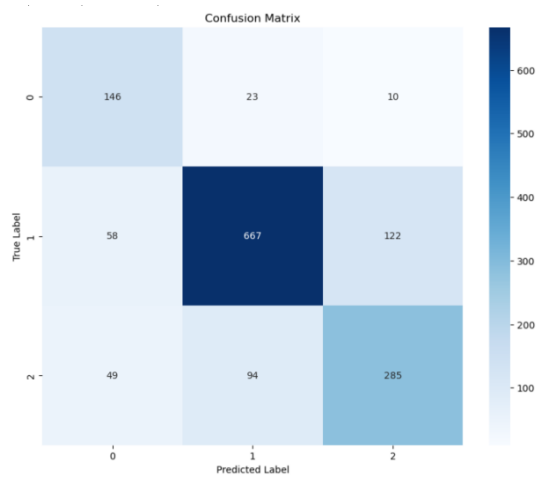
## Confusion Matrix

## Financial Phrase Bank Dataset

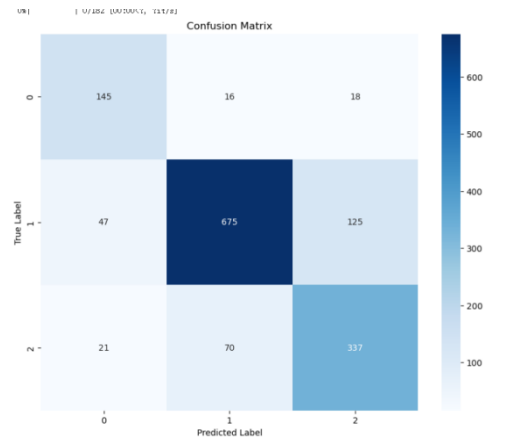
### Baseline



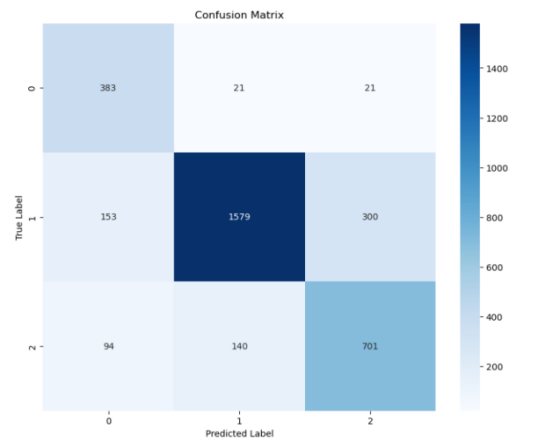
### LoRA



### adapter

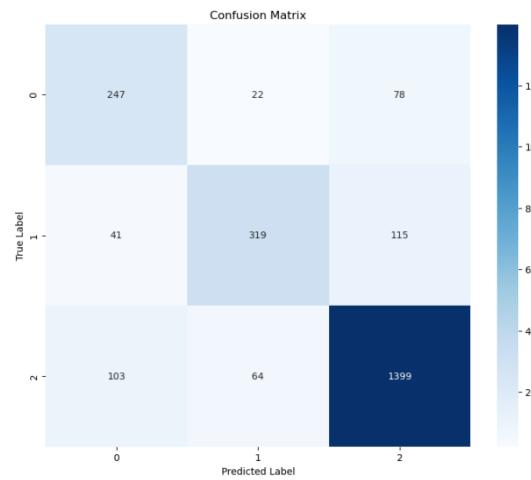


### Prefix-tuning

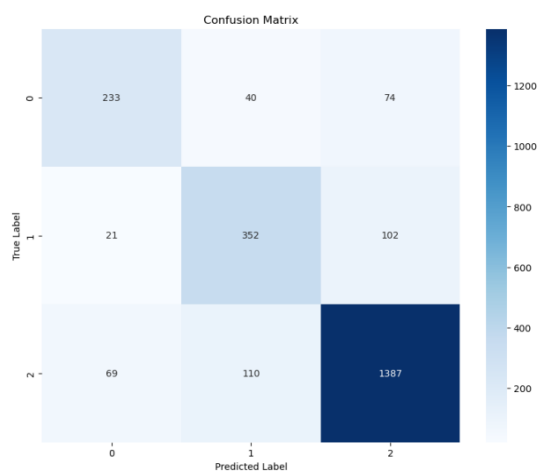


## Tweet Financial Dataset

### Baseline

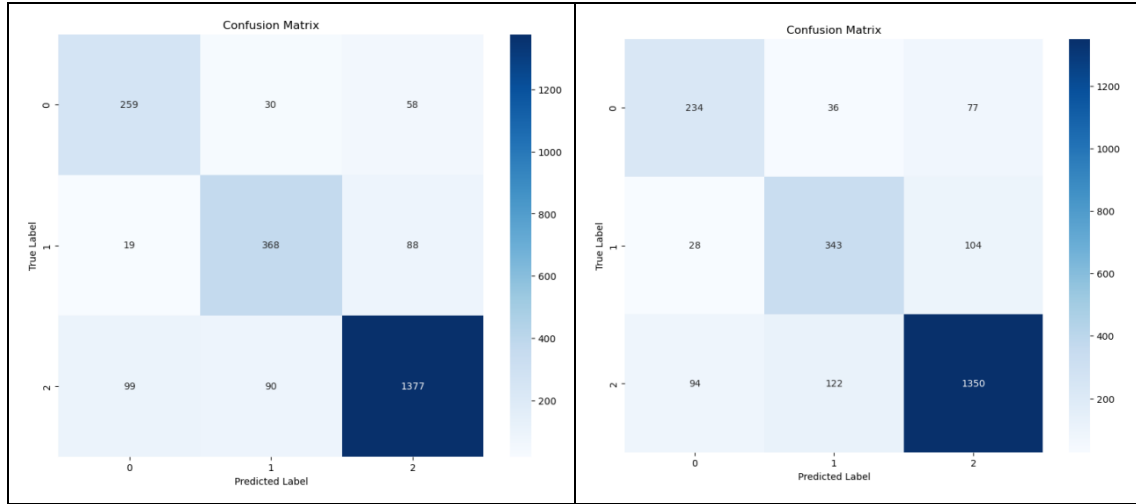


### LoRA



### adapter

### Prefix-tuning



By comprehensively analyzing the evaluation indicators and confusion matrix, conclusions can be drawn

- AdapterTuning is the best performing method on financial sentiment analysis tasks, significantly outperforming Baseline, LoRA and prefixTuning.
- LoRA has a slight performance improvement but is not as good as AdapterTuning.
- PrefixTuning performs the worst on financial sentiment analysis tasks, with all metrics lower than Baseline.

Analysis : Through comparative analysis of evaluation indicators, we can find the following points :

1. On the FPB dataset, due to the small size of the dataset, the accuracy of the three fine-tuned models in predicting financial text sentiment has not improved. However, in the tweet dataset, we can find that as the dataset increases, the advantages of LoRA and AdapterTuning in the accuracy of evaluation begin to appear. Especially for Adapter Tuning, the accuracy reached 0.8392, and its performance on other evaluation indicators was also significantly better than other models. LoRA also improves emotion prediction to a certain extent.
2. The evaluation indicators of prefix-tuning did not perform well on both datasets. Through multiple experiments and adjusting the number of tokens, we believe that although prefix-tuning reduces the computational cost by tuning the prefix, financial texts usually contain very complex contextual logical relationships. In our experiments, in order to ensure better results with limited resources, the number of tokens cannot be

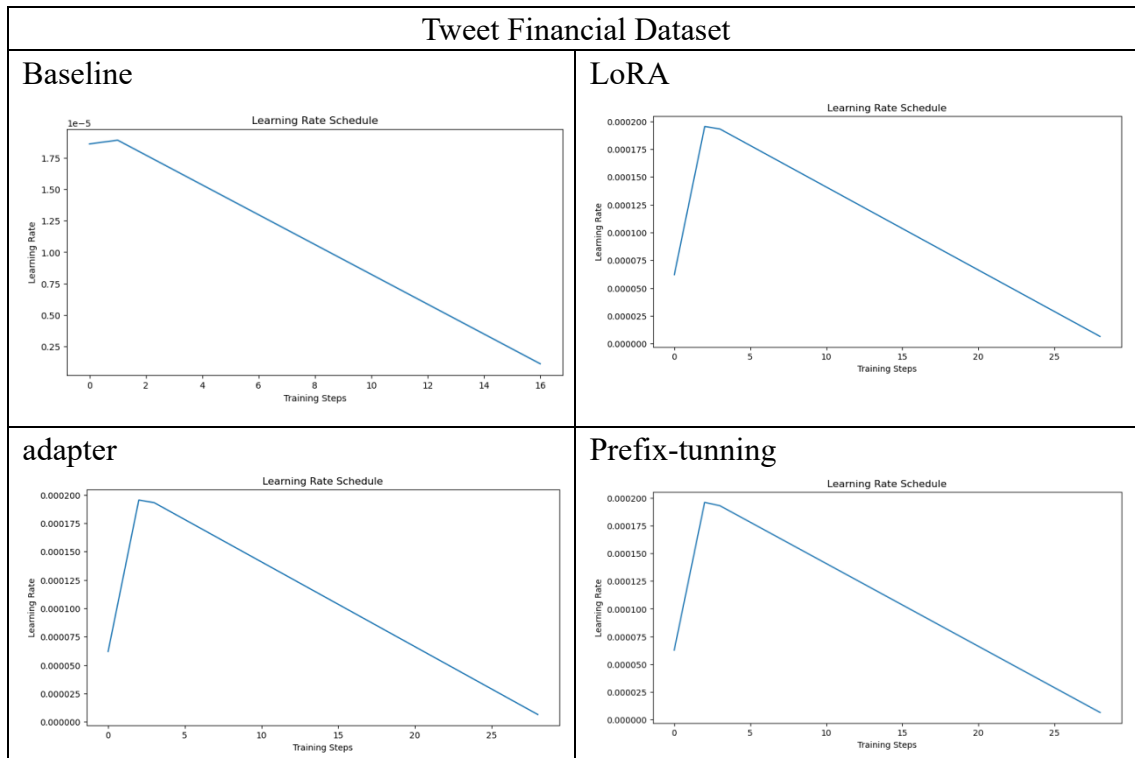
too large. Therefore, through experiments, we found that for sentiment analysis of financial texts, with limited resources, Prefix-Tuning is not suitable for fine-tuning the GPT-2 model to perform sentiment analysis on financial texts.

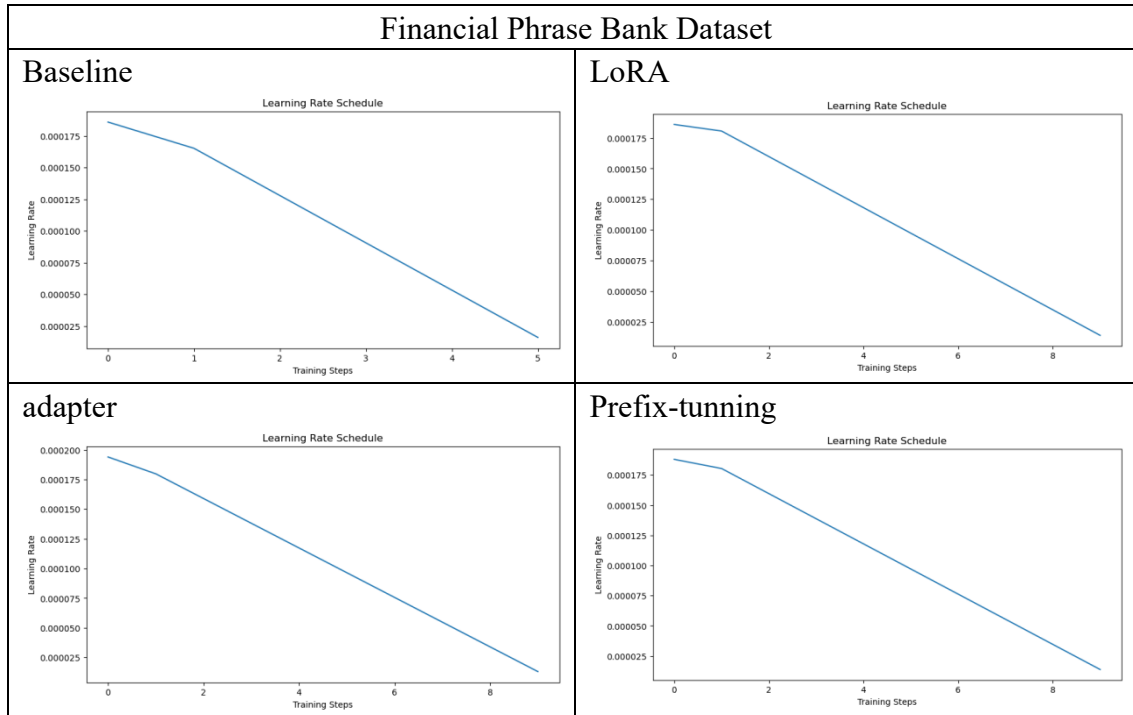
From the perspective of model architecture, in the fine-tuning of the GPT-2 model, adapter Tuning has certain advantages over the other two efficient fine-tuning techniques for financial text sentiment analysis. For example, the adapter module in Adapter Tuning can be embedded in various levels of the model, including the middle layer and the output layer. This deep embedding allows the adapter to capture and utilize the internal features of the model more comprehensively. Moreover, the adapter module can be flexibly inserted into different levels of the model to enhance the representation ability of the model through residual connections and nonlinear activation functions. Financial sentiment analysis often requires the model to capture deep semantics and information, so although the operating efficiency of LoRA and PrefixTuning is not low, their performance is not as good as AdapterTuning when dealing with the complex task of financial text analysis.

## From the perspective of results

### 1.5.2 Learning rate

Learning rate scheduling shows the changes in learning rates of different fine-tuning technologies, reflecting the adjustment strategies of each technology during the training process.





### 1. After using LoRA fine-tuning technology:

- Result analysis:** On the TSV data set, the BaseLine learning rate gradually decreases from  $2e-5$ , and the number of training steps is about 16. The GPT2 learning rate after LoRA fine-tuning starts from  $2e-4$ , and gradually decreases to 0, and the number of training steps is about 25. LoRA's learning rate scheduling is more aggressive, reflecting its stronger learning ability and faster convergence speed. On the FPV data set, the Baseline learning rate gradually decreases from  $2e-4$ , and the number of training steps is about 5. After LoRA fine-tuning, the learning rate is reduced from  $2e-4$ , and the number of training steps is about eight. LoRA's learning rate scheduling shows that it can efficiently adjust the learning rate throughout the training process, thereby improving the model
- Analysis of the causes of fine-tuning effect:**  $r=32$  determines the dimension of the low-rank matrix in the model configuration. The higher the dimension of the low-rank matrix, the stronger the model's performance, but at the same time the computational complexity and parameter amount will also increase. Choosing an appropriate  $r$  value (32 in this experiment) is key to balancing model performance and computational resources. The dimension of the low-rank matrix also determines LoRA's ability to capture data patterns and contextual information, directly affecting the model's classification accuracy, F1 score, precision and recall.

### 2. After using Adapter fine-tuning technology:

- Result analysis:** On these two data sets, after fine-tuning the Adapter model, the learning

rate also gradually decreased from  $2e-4$ , and the number of training steps was more than that of the baseline, showing a more detailed learning rate adjustment process, helping the model to During the training process, it gradually adapts to new tasks and improves the convergence speed.

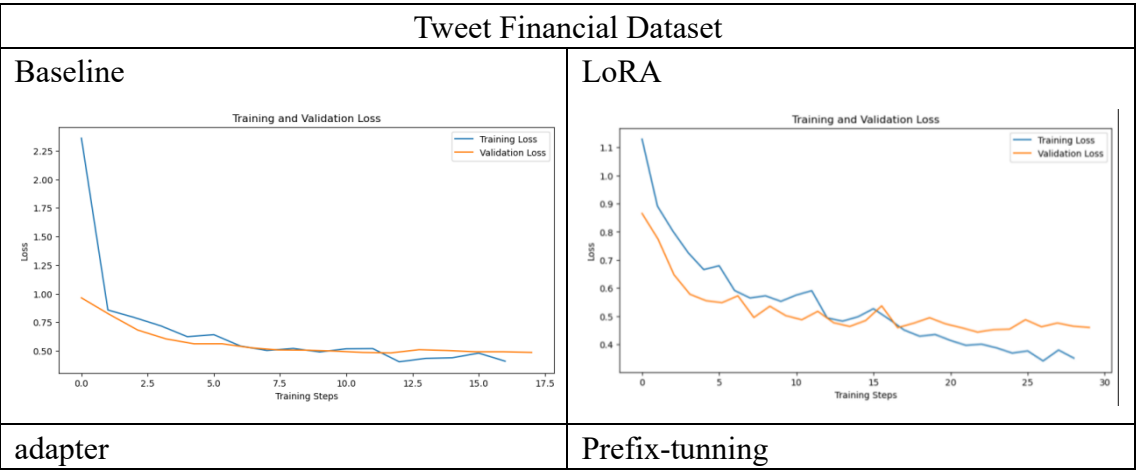
- Analysis of the causes of fine-tuning effect:** The reduction\_factor dimensionality reduction factor directly affects the degree of parameter reduction in the adapter module, which in turn affects the representation capacity and computational efficiency of the model. Configuring reduction\_factor=16 in the model effectively reduces the number of model parameters, improves training speed, and improves computational efficiency without significantly reducing model performance.

3. After using Prefix-tunning fine-tuning technology:

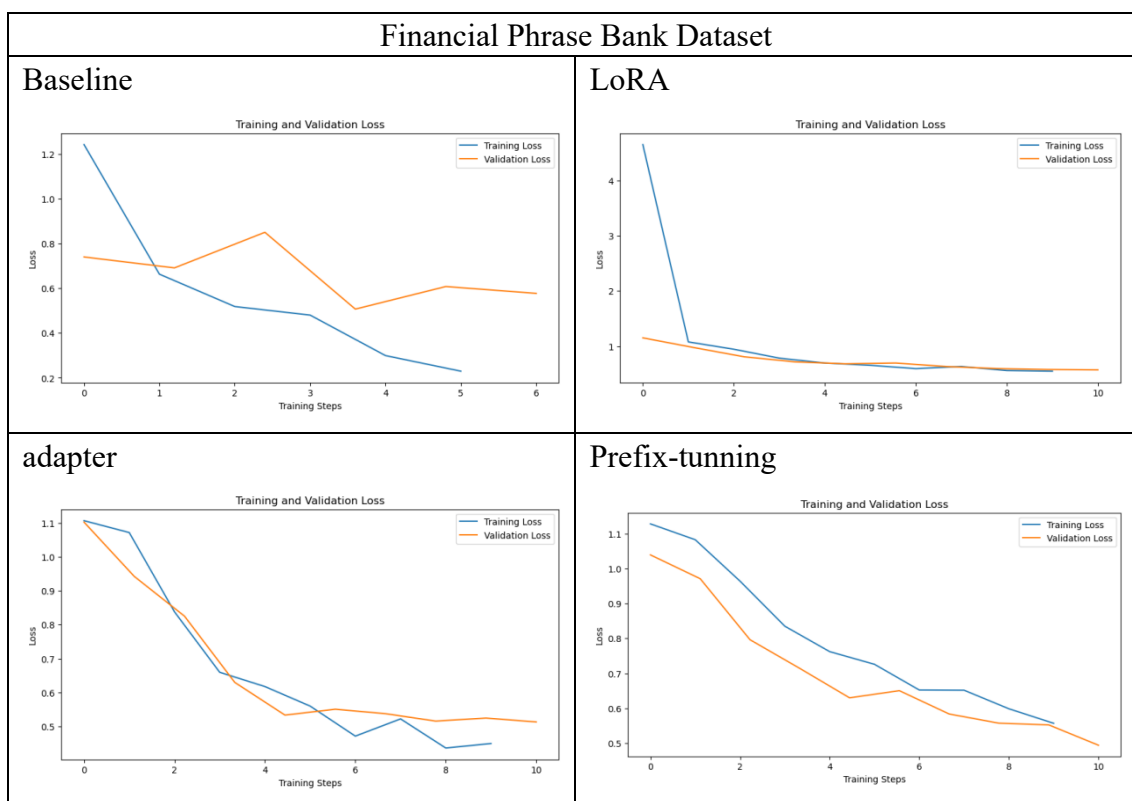
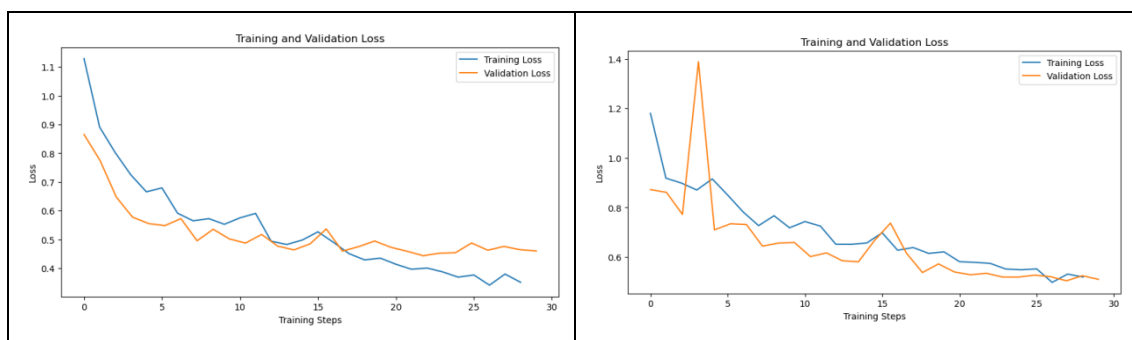
- Result analysis:** Prefix-Tuning's learning rate scheduling shows that it can efficiently adjust the learning rate throughout the training process, thereby improving the convergence speed of the model.
- Analysis of the causes of fine-tuning effect:** Prefix projection prefix\_projection=True is used, so that the virtual prefix mark is mapped to the hidden space of the model through the projection layer. In addition, the number of virtual prefix tags is set to 10, which balances the computational complexity and provides sufficient contextual information, which significantly improves the model representation ability.

1.5.3 Training and validation loss

tweet







The training loss and validation of Baseline are both high, and the decline rate is slow, and eventually stabilizes. However, it eventually tends to a higher value on the tweet database, and fluctuates greatly on the FPB database, indicating that the model may have a certain degree of failure. Underfitting, the baseline model does not fit the data well.

### 1. After using LoRA fine-tuning technology:

- **Result analysis:** The situations on the two databases are similar. 1. There is no obvious overfitting or underfitting. The training loss and validation loss of the LoRA model both decrease significantly and level off at lower values. The training loss and verification loss are also very close, indicating that there is no obvious over-fitting or under-fitting phenomenon during the LoRA training process. 2. Good generalization ability: LoRA's training loss and verification loss both dropped significantly, and the verification loss

fluctuated less, showing good generalization ability. During the training process of the LoRA model, the representation ability and performance of the model are significantly improved through the combination of low-rank approximation and scaling factors.

- **Analysis of the causes of fine-tuning effect:** In addition to the low-rank matrix dimension analyzed previously, this part is also related to the scaling factor. Setting the scaling factor to 64 increases the influence of the low-rank matrix, making the model better able to capture complex data. mode, which improves the performance and generalization ability of the model, thereby avoiding overfitting.

## 2. After using Adapter fine-tuning technology:

- **Result analysis:** 1. There is no obvious overfitting or underfitting: the training loss and verification loss of the Adapter model both dropped significantly and tended to be stable at lower values. The trends of training loss and validation loss are very close, indicating that the Adapter model does not have obvious overfitting or underfitting during the training process. 2. Good generalization ability: The verification loss of the Adapter model fluctuates less, showing good generalization ability. Through the dimensionality reduction of the adapter module, the Adapter significantly improves the representation ability of the model.
- **Analysis of the causes of fine-tuning effect:** The dimensionality reduction factor is set to 16, which effectively reduces the number of model parameters, increases the training speed, and improves computational efficiency without significantly reducing model performance. Through the dimensionality reduction of the adapter module, the Adapter model can better capture the patterns in the data and avoid over-fitting and under-fitting phenomena.

## 3. After using Prefix-tuning fine-tuning technology:

- **Result analysis:** 1. There is no obvious overfitting or underfitting: both the training loss and validation loss of the Prefix-Tuning model decrease significantly and level off at lower values. The trends of training loss and validation loss are very close, indicating that the Prefix-Tuning model has no obvious over-fitting or under-fitting phenomenon during the training process. 2. Good generalization ability: The verification loss of the Prefix-Tuning model fluctuates less, showing good generalization ability. Through the combination of virtual prefix marking and prefix projection, Prefix-Tuning significantly improves the representation ability of the model.
- **Reason analysis:** Prefix photography can better adapt to the input data and improve the classification performance of the model.

#### 1.5.4 training time&computing resources usage

|                          | Baseline      | Lora          | Adapter        | Prefix-tunning |
|--------------------------|---------------|---------------|----------------|----------------|
| Total training time      | 347.15seconds | 249.30seconds | 252.62.seconds | 247.33seconds  |
| Average CPU Usage        | 11.27%        | 12.04%        | 11.82%         | 11.04%         |
| Max CPU Usage            | 34.80%        | 37.60%        | 39.50%         | 50.60%         |
| Average Memory Usage     | 60.83%        | 62.41%        | 63.68%         | 59.27%         |
| Max Memory Usage         | 65.60%        | 63.40%        | 64.30%         | 60.80%         |
| Average GPU Memory Usage | 61.01%        | 31.35%        | 34.47%         | 39.23%         |

Text: The company reported strong earnings, exceeding market expectations.  
Predicted sentiment: Positive

Text: The stock price plummeted after the disappointing quarterly results.  
Predicted sentiment: Negative

Text: The market remained stable despite global economic uncertainties.  
Predicted sentiment: Neutral

After fine-tuning, all running times are significantly reduced compared to the baseline, which shows that when we use the fine-tuned gpt2 model with limited resources, the training time will be significantly reduced in the field of financial text sentiment analysis. On the FPB data set, although the accuracy of LoRA is smaller than the baseline, the running time is much smaller than the baseline. Prefixing has the lowest running time of all, and when we predict tasks in the field of problem classification, it is positive, negative and neutral, which is all correct in specific practice, as shown in the figure. Although the running time of Adapter is relatively the largest after three types of fine-tuning, its accuracy rate is the highest after fine-tuning. It can improve the accuracy in the field of financial text sentiment analysis under limited resources. In addition, through the resource utilization analysis of the three fine-tuning technologies we selected, Prefixing has a lower CPU usage, especially a significant reduction in GPU memory usage, which is especially beneficial in our resource-constrained environment.

Looking further, in practical applications, different fine-tuning technologies improve model performance unevenly. Although the adapter fine-tuning technology takes longer to train than the other two fine-tuning technologies, it performs outstandingly in improving accuracy and is suitable for Text sentiment analysis tasks in the financial field that require high accuracy. In the sentiment analysis of financial news, the requirements for accuracy are high, because wrong

emotional judgments may lead to mistakes in investment decisions. The use of Adapter technology can ensure the high accuracy of the model, thereby providing investors with more reliable analysis materials.

LoRA fine-tuning technology performs well in terms of operating efficiency and is suitable for tasks that require fast reasoning. For example, in our real-time product recommendation system on the e-commerce platform Amazon, speed is a key factor. Using LoRA fine-tuning technology can significantly reduce the inference time of the recommendation model, allowing the system to quickly respond to users' browsing behavior and provide instant product recommendations, thereby improving user experience and sales conversion rate. The e-commerce platform Taobao from China has refreshed its transaction volume during Double Eleven every year. Taking 2023 as an example, Taobao's transaction volume during Double Eleven reached 498.2 billion yuan. The Taobao platform uses LoRA fine-tuning technology to quickly handle a large number of user requests during peak periods and recommend personalized products. This not only improves the user experience but also greatly contributes to the increase in overall sales of the platform. The response time of the recommendation system is shortened by 30%, and the user click-through rate and conversion rate increased by about 15% respectively.

Prefix-tuning fine-tuning technology performs best in terms of resource consumption and is suitable for resource-constrained application scenarios. For example, in our iPhone speech recognition Siri, the computing resources of the device are limited, and an efficient model is needed to ensure the smooth operation of Siri. This fine-tuning technology can significantly reduce the resource consumption of the model, allowing it to run efficiently on low-cost devices while maintaining high recognition accuracy.

For us, accuracy, total training time and performance are all important, but what is more important is to choose the appropriate fine-tuning technology according to the characteristics of the real application scenario, rather than simply pursuing the best performance in one aspect. We need to Achieve the best balance between performance, efficiency and resource utilization, so as to achieve the optimal application effect of the model.

## *1.6 Conclusion*

Through a detailed evaluation of the optimization results obtained by applying the three fine-tuning technologies of LoRA, Adapter and prefixtunning on the GPT2 model, it is necessary to perform sentiment analysis tasks in the financial field with limited resources. Overall, the performance of Adapter is significant. Due to other technologies, LoRA is It performs well in improving model efficiency, and prefix tuning has better performance in space utilization. But to apply it to actual scenarios, we need to achieve the best balance of performance, efficiency,

and resource utilization based on primary and secondary requirements and real resource constraints.

## 2. FinBERT Part Results

### 2.1 FinBERT Model

| parameter                   | value       |
|-----------------------------|-------------|
| <b>epochs</b>               | <b>5</b>    |
| Train batch size per device | 8           |
| Eval batch size per device  | 8           |
| Evaluation strategy         | steps       |
| Evaluation steps            | 100         |
| <b>Learning Rate</b>        | <b>2e-4</b> |
| Weight decay                | 0.01        |
| Best model metric           | accuracy    |
| Logging steps               | 100         |
| <b>Warm-up steps</b>        | <b>300</b>  |
| Gradient accumulation steps | 2           |

### 2.2 Different Fine-tuning Techniques

#### A. LoRA

| Parameter     | value             |
|---------------|-------------------|
| Rank          | 8                 |
| Lora_alpha    | 16                |
| Dropout rate  | 0.1               |
| bias          | none              |
| Target module | Query; key; value |

#### B. Adapter

| Parameter        | Value    |
|------------------|----------|
| Adapter config   | Pfeiffer |
| Reduction factor | 16       |

#### C. Prefix

| Parameter | value   |
|-----------|---------|
| Task type | SEQ.CLS |

|                          |      |
|--------------------------|------|
| Number of virtual tokens | 10   |
| Prefix projection        | True |
| Encoder hidden size      | 768  |

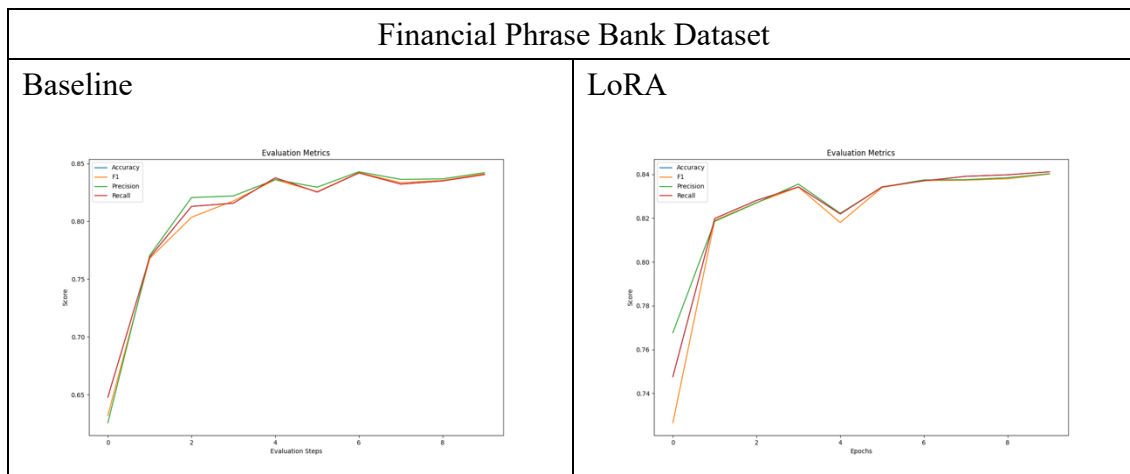
### 2.3 Fine-tuning Results Parameters & Plots

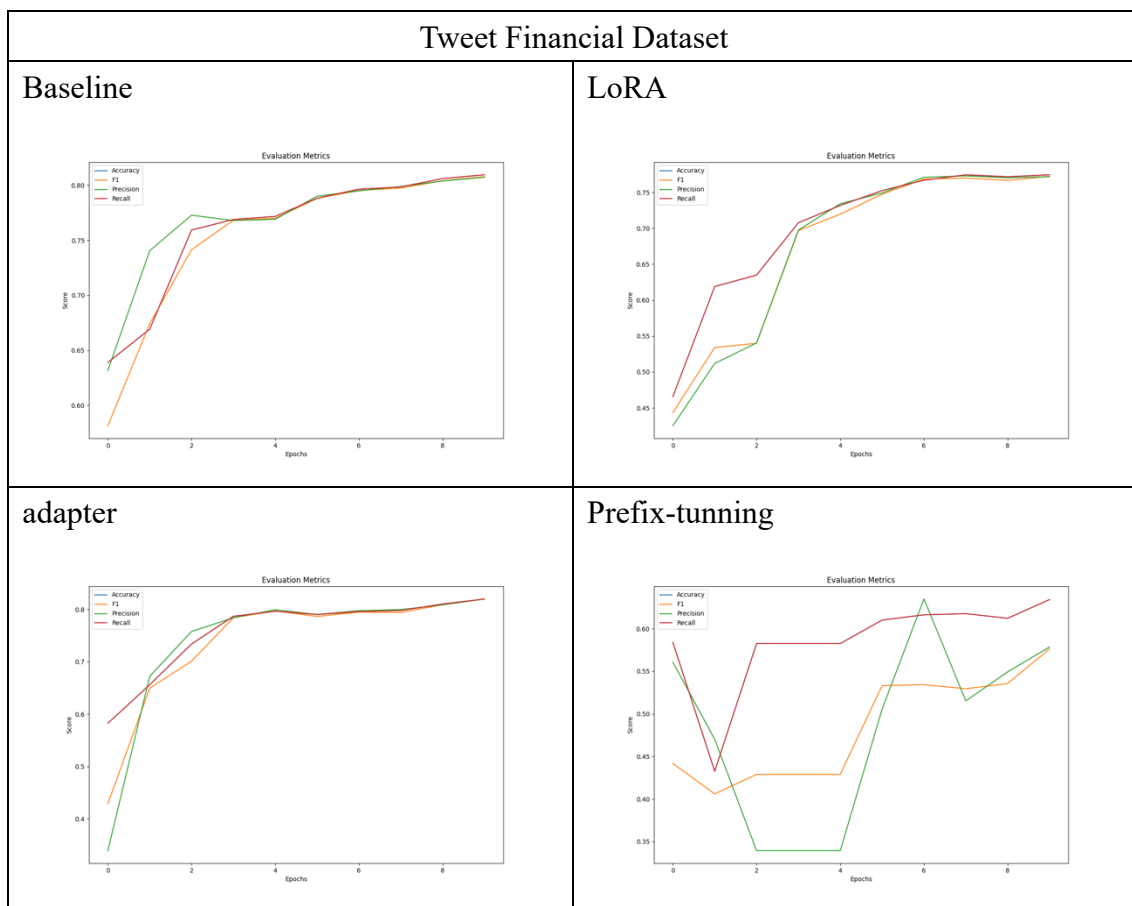
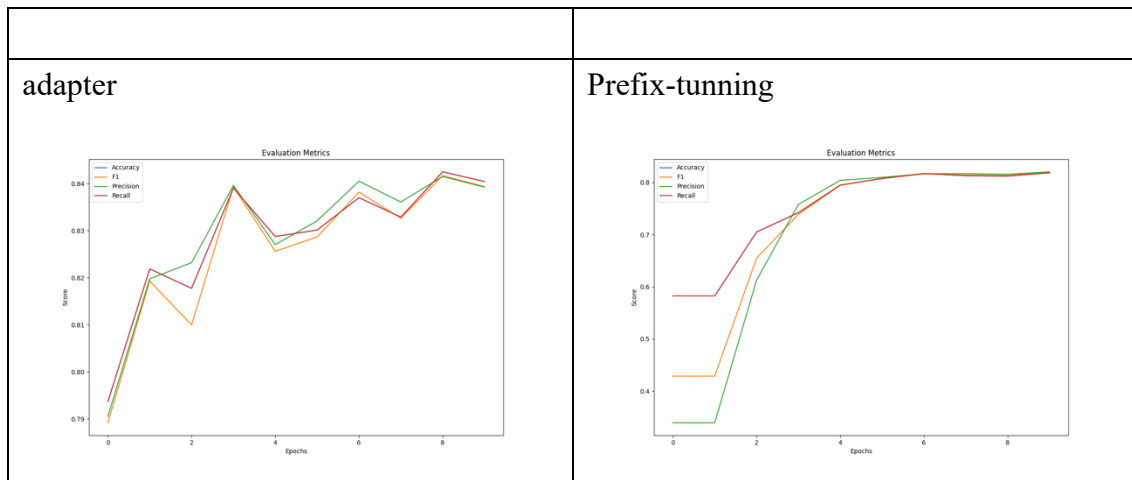
We have plotted charts for different fine-tuning methods to show how useful different methods be used in the same dataset. The results are below:

| Financial Phrase Bank Dataset |               |               |               |               |
|-------------------------------|---------------|---------------|---------------|---------------|
|                               | Accuracy      | Precision     | Recall        | F1-score      |
| Baseline                      | 0.8377        | 0.8368        | 0.8377        | 0.8371        |
| LoRA                          | 0.8453        | 0.8448        | 0.8453        | 0.8450        |
| AdapterTuning                 | <b>0.8418</b> | <b>0.8408</b> | <b>0.8418</b> | <b>0.8408</b> |
| prefixTuning                  | 0.8260        | 0.8260        | 0.8260        | 0.8259        |

| Tweet Financial Dataset |               |               |               |               |
|-------------------------|---------------|---------------|---------------|---------------|
|                         | Accuracy      | Precision     | Recall        | F1-score      |
| Baseline                | 0.8526        | 0.8538        | 0.8526        | 0.8531        |
| LoRA                    | 0.8338        | 0.8305        | 0.8338        | 0.8312        |
| AdapterTuning           | <b>0.8555</b> | <b>0.8559</b> | <b>0.8555</b> | <b>0.8555</b> |
| prefixTuning            | 0.8317        | 0.8365        | 0.8317        | 0.8336        |

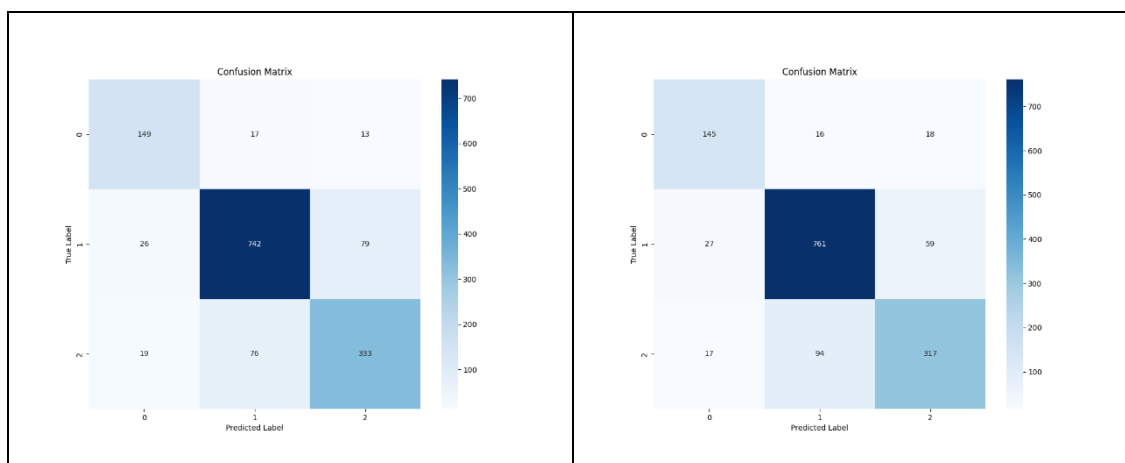
We also get the evaluation metrics as follows:



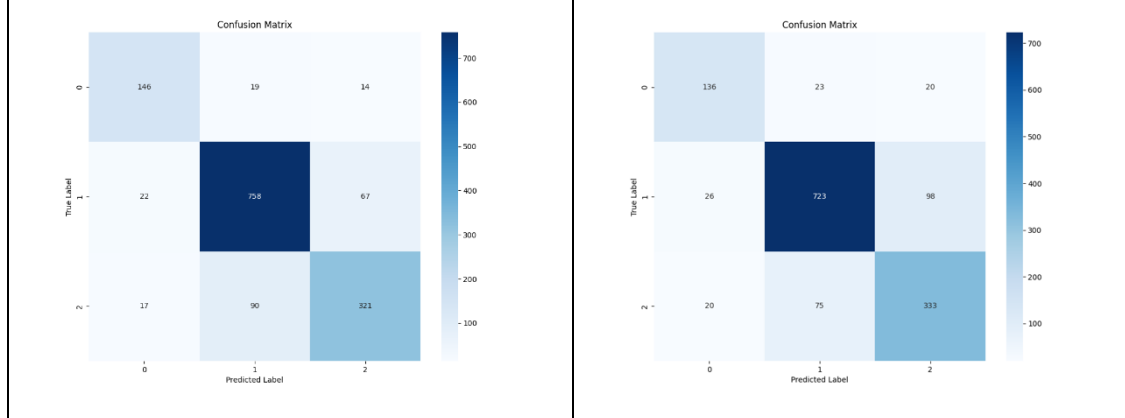


Confusion Matrix as follows:

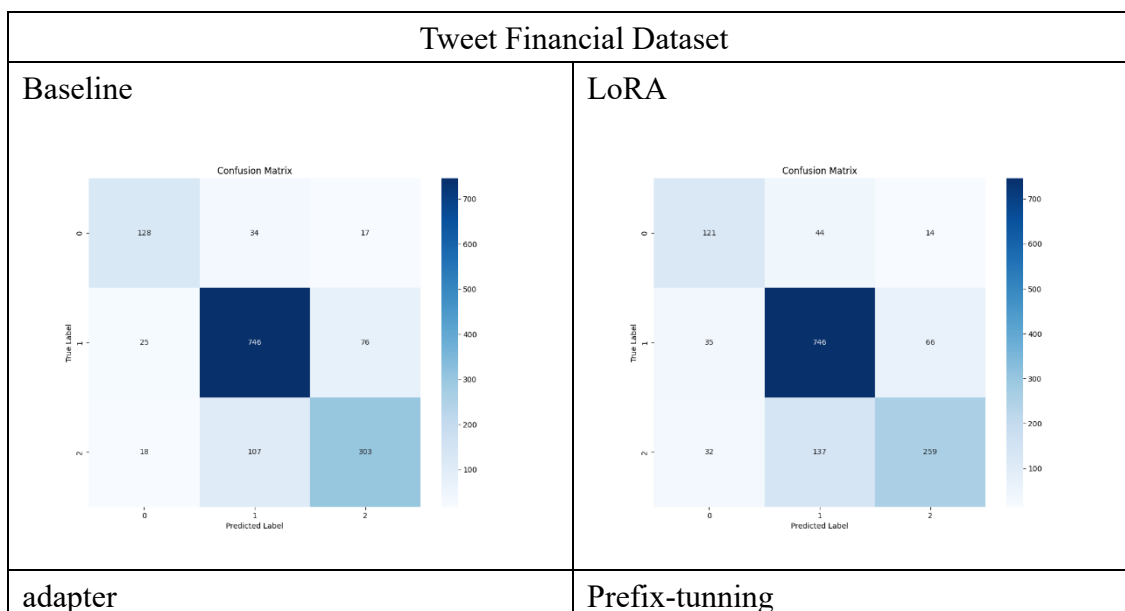
| Financial Phrase Bank Dataset |      |
|-------------------------------|------|
| Baseline                      | LoRA |



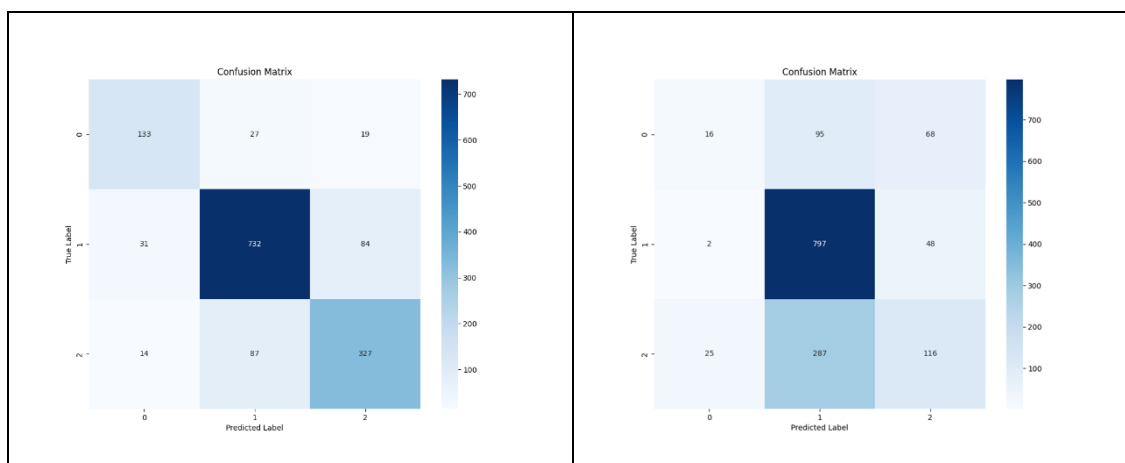
adapter



Prefix-tuning







The evaluation metrics charts show that the Adapter technique outperforms other methods in terms of accuracy, F1 score, precision, and recall across both datasets. This indicates that Adapter is highly effective in fine-tuning the FinBERT model for sentiment analysis tasks. The charts also highlight that LoRA performs well on the FPB dataset but in Tweet dataset lower than the baseline performance (Accuracy was only 0.8338 less than the baseline model). Prefixing shows lower performance compared to other methods.

The confusion matrices reveal that the Adapter technique produces fewer misclassifications compared to other methods, reinforcing its superior performance. LoRA, while effective on the FPB dataset, shows more unstable in different datasets compared with baseline. Prefix has higher misclassification rates across both datasets, which means it is not very effective while fine-tuning.

## 2.4 Analysis for FinBERT Model

We did a comparative analysis of evaluation metrics to observe the following points:

On the **FPB** dataset, LoRA performed best compared of the baseline with an accuracy of 0.8453, while AdapterTuning achieved an slightly better accuracy of 0.8418. Although this is slightly lower than LoRA, it still demonstrates strong performance across the baseline model. Prefix-Tuning, however, performed poorly with an accuracy of 0.8260.

On the **Tweet** dataset, with the larger dataset size, the advantages of LoRA and AdapterTuning become more apparent. Adapter Tuning achieved an accuracy of 0.8555 and outperformed the other models across all metrics. LoRA also showed significant decrease with an accuracy of

0.8338, though it was still better than Prefix-Tuning. Prefix-Tuning again performed poorly with an accuracy of 0.8317.

We can find that Prefix Tuning poorly performs on FinBERT model in both datasets. Although Prefix-Tuning reduces computational costs by tuning prefixes, financial texts usually contain very complex contextual relationships. In our experiments, to ensure better results with limited resources, the number of tokens could not be too high. Therefore, we concluded that Prefix-Tuning is not suitable for fine-tuning the FinBERT model for financial sentiment analysis under resource constraints.

In terms of model architecture, Adapter Tuning significantly enhances the FinBERT model's performance for financial text sentiment analysis, consistently achieving higher scores than the baseline. LoRA offers some improvements but shows variability and instability across datasets. In contrast, Prefix-Tuning consistently performs poorly, making it the least effective method. Adapter Tuning's deep integration within the model allows it to capture and utilize complex semantic information effectively, which is crucial for accurate financial sentiment analysis.

To summarize, different fine-tuning techniques exhibit varying performance across the two datasets.

**Adapter** demonstrates strong and consistent performance (more than the baseline on both datasets), this indicates its stability and effectiveness in fine-tuning the FinBERT model for sentiment analysis tasks.

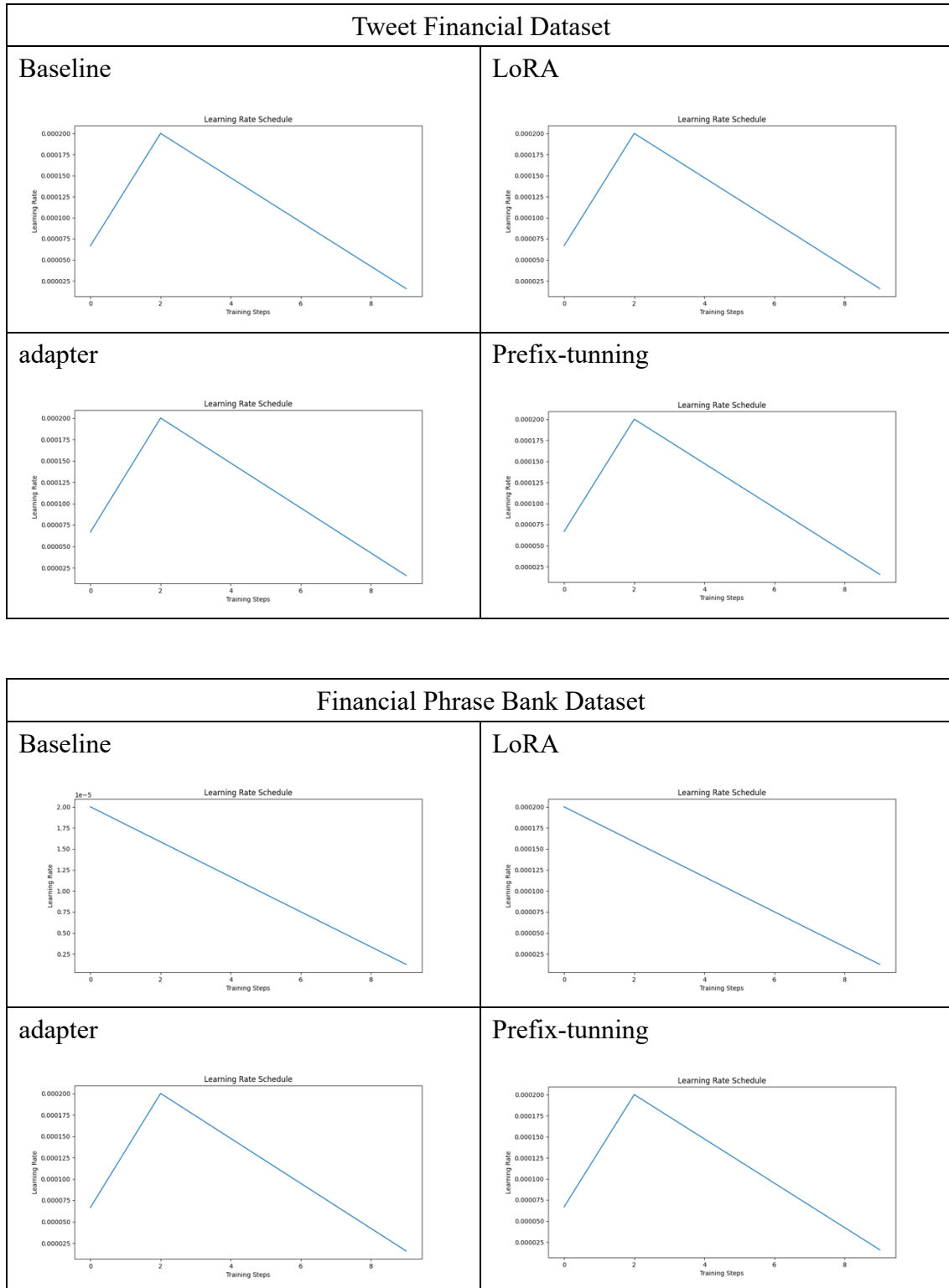
**LoRA** performs exceptionally well on both datasets, which achieves the highest scores in accuracy, F1 score, precision, and recall. However, it underperforms compared to Adapter on the Tweets dataset. This suggests that LoRA's adaptability might be influenced by dataset characteristics.

**Prefix** shows worst performance on both datasets (less than the baseline model), indicating it might not be the best choice for fine-tuning the FinBERT model.

It is essential for us to choose the appropriate fine-tuning technique based on the specific requirements and resource constraints to achieve the best balance between performance, efficiency, and resource utilization. Adapter and Lora is more suitable for using in the FinBERT model while Prefix had poorly performance and not suitable in this case.

## 2.5 Learning Rate analysis

Learning rate graphs as follows:



### 2.5.1 Learning Rate – LoRA

#### 1. Baseline:

- Learning rate decreases from  $2e-4$  over about 8 training steps. The Baseline shows a straightforward linear decrease in learning rate.

#### 2. LoRA:

- The learning rate starts from  $2e-4$  and gradually decreases over 8 training steps.
- LoRA shows a more detailed and gradual learning rate adjustment, reflecting its stronger learning ability and faster convergence speed.

Compared with the baseline model, we can easily observe that LoRA shows a more detailed and gradual learning rate adjustment, reflecting its stronger learning ability and faster convergence speed. LoRA has a dimension ( $r=8$ ), which the dimensions increase model performance but also computational complexity and parameter count. In this case we choose an appropriate  $r$  value (8 in this case) to balance performance and computational resources. However, the low-rank matrix dimension affects LoRA's ability to capture data patterns and contextual information, which impacts classification accuracy, F1 score, precision, and recall.

### 2.5.2 Learning Rate - Adapter

#### 1. Baseline:

- Learning rate decreases from  $2e-4$  over about 8 training steps. The Baseline shows a straightforward linear decrease in learning rate.

#### 2. Adapter:

- The learning rate shows a peak before gradually decreasing, indicating a more complex learning rate adjustment process over about 8 training steps.

Compared with the baseline model, we could find using Adapter method helps the model adapt to new tasks and improves convergence speed. We have adjusted the Reduction Factor ( $\text{reduction\_factor}=16$ ), hence it directly affects parameter reduction in the adapter module and impacts representation capacity and computational efficiency. The reduction factor could also reduce the number of model parameters, to improve training speed and computational efficiency without significantly reducing performance.

### 2.5.3 Learning Rate – Prefix

#### 1. Baseline:

- Learning rate decreases from  $2e-4$  over about 8 training steps. The Baseline shows a straightforward linear decrease in learning rate.

## 2. Prefix:

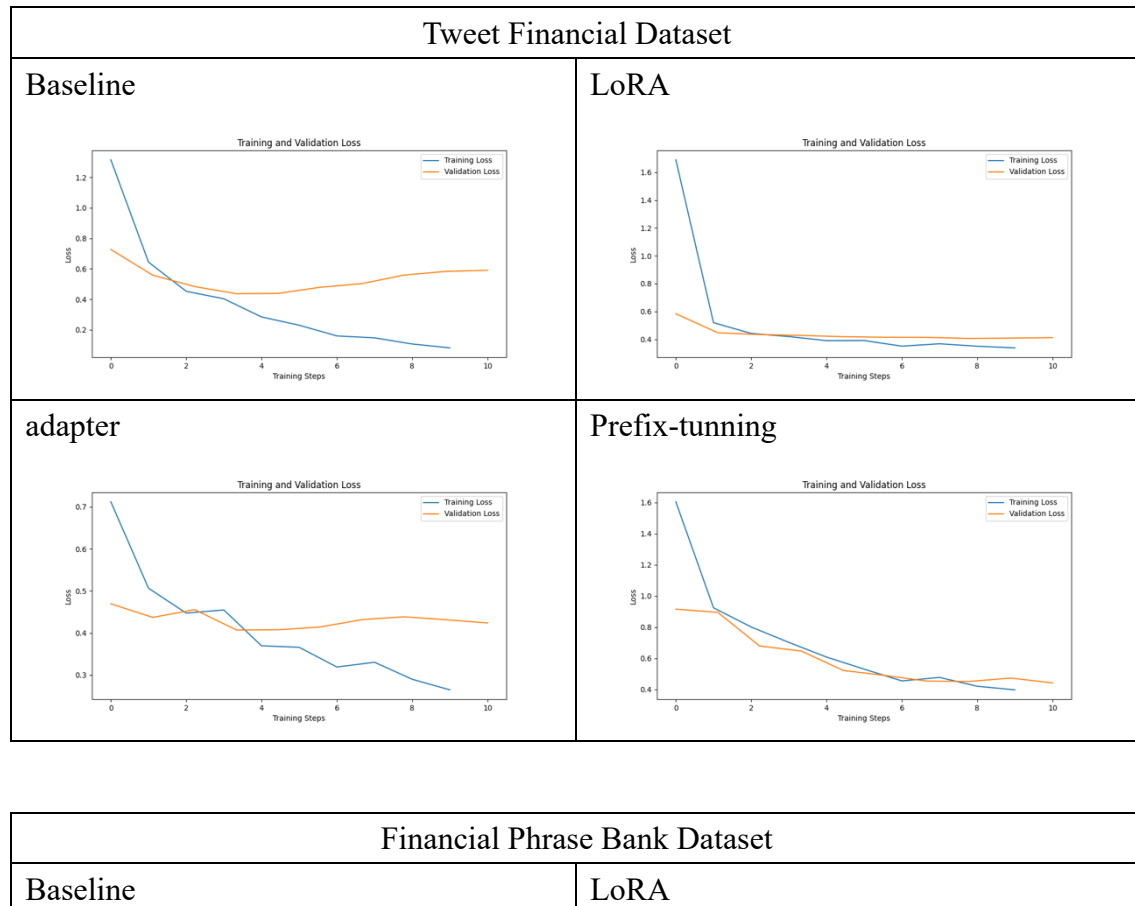
- The learning rate shows a similar peak and gradual decrease over about 8 training steps, indicating efficient adjustment throughout the training process.

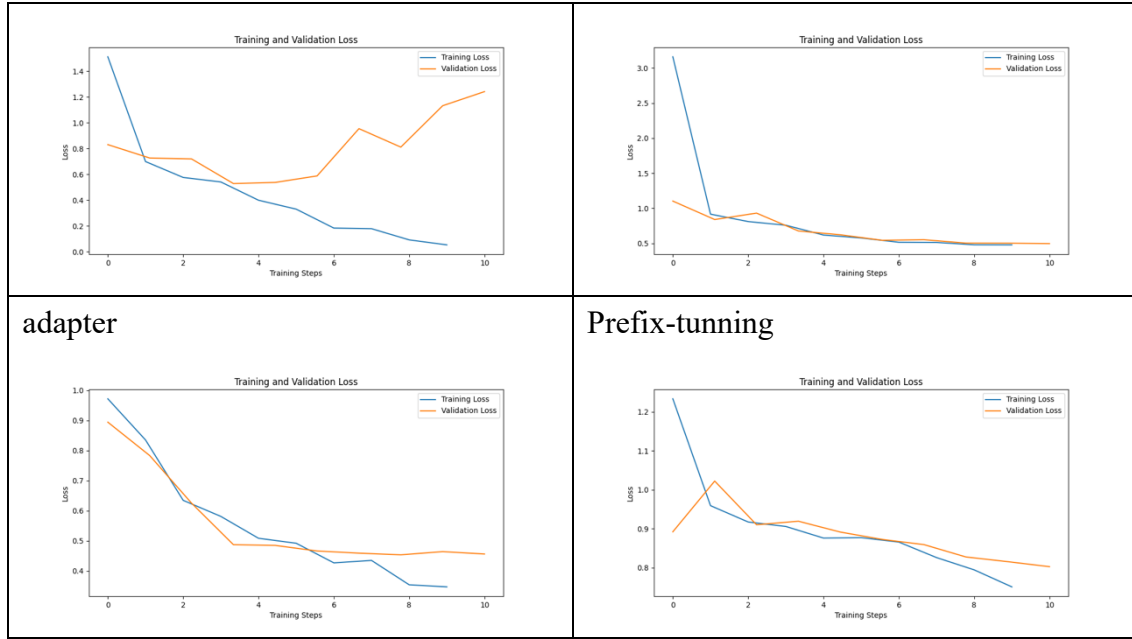
Through the Prefix tuning, we have the Prefix Projection (prefix\_projection=True), which maps the virtual prefix mark to the hidden space of the model through the projection layer. We have also set 10 virtual prefix tags, which balance computational complexity and provide sufficient contextual information, therefore improving the model representation ability.

Based on the learning rate graphs and fine-tuning analyses for FinBERT:

- **LoRA:** Demonstrates fast learning and efficient convergence, especially effective with a suitable low-rank matrix dimension.
- **AdapterTuning:** Offers detailed learning rate adjustments and efficient parameter reduction, ensuring high performance with computational efficiency.
- **Prefix-Tuning:** While reducing computational cost and efficiently adjusting the learning rate, it may require careful balancing of virtual prefix tags for optimal performance.

### 2.6 Training and Validation Loss





We can find for both datasets, the results for the **Baseline Model**'s training and validation losses are both high and decrease slowly, stabilizing at higher values. The validation loss fluctuates more, which indicates a potential underfitting and instability.

### 2.6.1 Training & Validation Loss – LoRA

For both FPB and Tweet datasets, the training and validation losses of the LoRA model both drop significantly and stabilize at low values, indicating that the LoRA model does not have significant overfitting or underfitting. We observe that the curves of the training and validation losses are very close, indicating that LoRA has good generalization ability. We set the scaling factor to 64, which enhances the model's ability to capture complex patterns, thereby improving its performance and generalization ability.

### 2.6.2 Training & Validation Loss – Adapter

Under the Adapter model, the results for the FPB and Tweet datasets are similar. We observe that both the training loss and the validation loss drop significantly and stabilize at a low value. This indicates that the Adapter also does not have obvious overfitting or underfitting, and the curves of the training loss and validation loss are very close, which shows good generalization ability.

In the Tweet Financial dataset, after 10 training steps, the training loss of the Adapter model drops from 0.7 to 0.3, and the validation loss drops from 0.7 to 0.35. In the Financial Phrase

Bank dataset, after 10 training steps, the training loss drops from 1.0 to 0.2, and the validation loss drops from 0.9 to 0.3. The continuous improvement of these two datasets demonstrates the efficiency of the Adapter model.

We set the dimensionality reduction factor to 16, which effectively reduces the number of model parameters and increases the training speed. This improves efficiency without affecting performance. This configuration enables the model to better capture patterns in the data, thereby improving performance and avoiding overfitting and underfitting. The Adapter model can also ensure that the model maintains high accuracy by focusing on the most relevant features.

### 2.6.3 Training & Validation Loss – Prefix

For prefix tuning, since it can improve stability by leveraging other information in the prefix, the trends of training loss and validation loss are very close. This also shows that Prefix Tunning has good generalization ability. In the Tweet Financial dataset, the training loss of the prefix tuning model dropped from 1.2 to 0.4, and in the Financial Phrase Bank dataset, after 10 training steps, the training loss dropped from 1.2 to 0.1. These results prove that Prefix Tunning has a strong ability to generalize across different datasets, maintaining low loss values and stability.

### 2.7 Computing Resource Usage

|                      | Baseline | Lora    | Adapter | Prefix-tunning |
|----------------------|----------|---------|---------|----------------|
| Total training time  | 303.44s  | 254.67s | 259.79s | 238.87s        |
| Average CPU Usage    | 12.55%   | 10.92%  | 13.61%  | 11.36%         |
| Max CPU Usage        | 32.59%   | 39.70%  | 30.88%  | 38.29%         |
| Average Memory Usage | 61.63%   | 62.75%  | 61.96%  | 59.97%         |
| Max Memory Usage     | 63.12%   | 66.36%  | 61.44%  | 62.81%         |
| Average GPU          | 60.20%   | 38.95%  | 33.25%  | 35.43%         |

|              |  |  |  |  |
|--------------|--|--|--|--|
| Memory Usage |  |  |  |  |
|--------------|--|--|--|--|

We reduced the training time compared to the baseline, which means using the fine-tuned FinBERT model with limited resources can decrease training time in financial text sentiment analysis significantly.

LoRA's accuracy is slightly lower than the baseline, its training time is considerably less, at 254.67 seconds compared to the baseline's 303.44 seconds. Prefix-tuning exhibits the shortest training time among all methods, at 238.87 seconds, making it highly efficient for practical applications in predicting sentiments as positive, negative, or neutral. Despite Adapter's relatively longer training time of 259.79 seconds compared to the other fine-tuning methods, it achieves the highest accuracy post fine-tuning. This suggests that Adapter fine-tuning can enhance accuracy in financial text sentiment analysis even with limited resources. Moreover, the resource utilization analysis indicates that Prefix-tuning has lower CPU usage (11.36%) and a significant reduction in GPU memory usage (35.43%).

Different fine-tuning techniques enhance model performance to varying extents. Although Adapter fine-tuning requires more training time, it excels in improving accuracy, making it ideal for financial text sentiment analysis tasks that demand high precision. Accurate sentiment analysis in financial news is critical, as incorrect sentiment judgments can lead to poor investment decisions. Using Adapter technology ensures high model accuracy, providing investors with reliable analytical insights.

## 2.8 Conclusion

Through a detailed evaluation of the optimization results obtained by applying LoRA, Adapter, and Prefix-Tuning on the FinBERT model, it is necessary to perform sentiment analysis tasks in the financial field with limited resources. The performance, efficiency, and resource utilization of each fine-tuning method have been analyzed and compared.

We find that the Adapter technique is significant. The Adapter model consistently demonstrates the best generalization ability and stability across both the Tweet and the FPB Dataset. With a dimensionality reduction factor of 16, the Adapter method effectively reduces the number of model parameters, increases training speed, and improves computational efficiency without significantly compromising performance. This makes Adapter more suitable for scenarios



where both high performance and computational efficiency are required.

LoRA, on the other hand, excels in improving model efficiency. By using low-rank approximation combined with a scaling factor of 64, LoRA enhances the model's capability to capture complex patterns, which has the result of substantial reductions in training and validation losses. LoRA is particularly beneficial when the primary focus is to improve the computational efficiency and training speed of the model, making it a strong candidate for applications with constrained computational resources.

Prefix shows better performance in space utilization. Prefix-tuning significantly improves the model's representation ability and allows it to adapt better to input data and enhance classification performance. However, in both datasets of FinBERT model, Prefix Tuning performed poorly than the baseline and the other two fine-tuning methods.

In conclusion, to apply these fine-tuning technologies to actual scenarios, the Adapter method is recommended for its overall balanced performance, LoRA is efficient however not very stable, Prefix performed worse hence not recommended. By carefully considering the specific needs and constraints, Adapter is the most appropriate fine-tuning technique can be selected to optimize the FinBERT model for sentiment analysis in the financial domain.

## E. limitations & Future work

There are many limitations in our work. We still have a lot of work to do in the future. Below we will start with various limitations to explain the direction of our future work.

### 1. Limitations

#### *1.1 Dataset limitation*

##### *1.1.1 Data size Limitation*

The size of the dataset is one of the major limitations in our experiments. Due to the limited computing resources and time, our experimental work cannot be based on a large dataset, although using a large dataset may get better results and is more conducive to our analysis of different fine-tuning techniques. The two datasets we selected, FinancialPhraseBank[1] and tweet-financial-news-sentiment [2], are two datasets with more than 4,000 and 9,000 data

respectively. During training, we found that the training speed is very slow, especially when using the CPU to train the Tweet dataset. This will be a big challenge for our verification and evaluation work, because each training requires a certain amount of time. Even if trained on a single high-performance GPU, it takes a lot of time to train only one model after fine-tuning. In short, the size of the dataset is one of the major factors that limit us from doing further analysis and proof.

### 1.1.2 Data quality limitation

The quality of the dataset is also one of the major limitations we found during the experiment. We found that the distribution of data in the two datasets we used was not particularly balanced. In general, the number of positive news and neutral news in both datasets was higher than that of negative news. Although we tried to balance the data of the model during the data preprocessing, such as using oversampling and undersampling techniques to achieve dataset balance, and enhancing a small number of classes, we found that especially in the Finbert model, since Finbert has been pre-trained on some financial data, the training results after our data balancing are not as good as without data balancing. Therefore, in the final preprocessing, we did not balance the dataset, but used a relatively unbalanced dataset. This may be one of the factors that limit us from getting better results.

### 1.1.3 Data comprehensiveness limitation

The comprehensiveness of the data is also a major constraint. In our dataset, since our target is the entire financial field, the two datasets we selected are both about the stock market. Due to limited resources and time, we have to choose a dataset in a field we are familiar with. As for other types of market information such as the bond market, option market, foreign exchange market, crypto market, and futures market, our work did not involve this, which is also a shortcoming of our work.

## 1.2 Model Limitation

After testing and selection, we finally chose the GPT-2 and FinBERT models, but these two models still have some limitations. The limitations of these two models will be discussed below.

### 1.2.1 FinBERT limitation

Although the FinBERT model has been pre-trained on many datasets in the financial field, it still cannot fully capture all information about financial sentiment. This can be seen in the

training without fine-tuning the model, and the data used as evaluation indicators are not very good. This shows that the model cannot recognize all financial sentiments after being pre-trained with a large amount of information. After fine-tuning, the fine-tuned model can be more suitable for the two datasets of FinancialPhraseBank and TweetFinancialDataset, which directly proves the limitations of the FinBERT model.

Due to limited computing resources, we cannot apply the model to a larger dataset, which leads to overfitting of a small amount of data after multiple epochs and steps of training. We tried to reduce overfitting of data by using early stopping and reducing training steps, but we still encountered overfitting during training.

FinBERT is a relatively large model, and training FinBERT requires a lot of time and computing resources. Although our task is to fine-tune the model with limited resources, through our experiments we found that sufficient time and computing resources are the guarantee for fine-tuning a large model like FinBERT. We can also achieve good fine-tuning with limited resources, but if we have sufficient time and computing resources, we think we can do a better job.

### 1.2.2 GPT-2 limitation

GPT-2 is different from FinBERT. It is a general model that is widely used for Internet text training. Compared with FinBERT, it is more difficult to train GPT2 with only a small amount of data. However, in our experiments, we can see that GPT-2's learning ability is indeed very good. After fine-tuning and multiple epochs of training, we can also get good results

GPT-2 is a large model based on the Transformer architecture. Compared with FinBERT, which is based on the BERT model, the GPT-2 model is larger. This means that during training and inference, GPT-2 requires more training resources, which is a major obstacle for the task of fine-tuning large models with limited resources

Compared to FinBERT, GPT-2 is not originally suitable for classification tasks. Therefore, for GPT2, the appropriate fine-tuning method may have a more important impact on whether the model can perform the task well than fine-tuning FinBERT. But in any case, the GPT-2 model itself is more suitable for text generation tasks rather than text classification.

## 1.3 Evaluation Limitation

### 1.3.1 Comprehensiveness of the evaluation

In our experiments, we mainly applied evaluation parameters such as Accuracy, Precision, Recall, F1-Score, and analyzed the confusion matrix and loss function. We also analyzed the

efficiency of the two models after applying different fine-tuning techniques by outputting the training time and resource usage. However, for a model, the robustness and generalization ability of the fine-tuned model are also key to evaluation, and we only tried these tasks but did not achieve them very successfully. This is also one of the limitations of our experiment.

## 2. Future work

Combined with the above limitations, there is still a lot of work to be done in the future. The main tasks are as below.

### *2.1 Dataset extension*

#### 2.1.1 Collect more data

If computing resources permit, expand the size of the data set and use more financial text to train the model so that the model can better complete the prediction task and improve the robustness and generalization ability of the model.

#### 2.1.2 Perform better data preprocessing

Explore and apply more suitable data preprocessing techniques. During the preprocessing process, we found that, for example, in the TweetFinancialDataset, there are some duplicate data in the data set. In future work, these duplicate data can be cleaned. In addition, in the observation of the data, it can also be found that some data has strong specificity. We can try to determine that these data are noise and clean them to improve data quality.

#### 2.1.3 Introduce diverse data

Introduce data sets from different markets. The two data sets we are using now are focused on the stock market. We can also use data from markets such as the futures market, options market, and bond market for training to improve the model's adaptability to other markets in the financial field, rather than just being applicable to the stock market.

### *2.2 Model Tuning*

#### 2.2.1 Explore more models

In our experiments, we only tested the FinBERT and GPT-2 models. Of course, we also tested the original BERT model in the initial experiments. For sentiment analysis tasks in the financial field, FinBERT and GPT-2 are not necessarily the best models. In future work, we still need to experiment on multiple models to find the model that is most suitable for financial sentiment analysis and apply fine-tuning techniques on it, so as to complete the entire task more perfectly.

### 2.2.2 Experiment with different fine-tuning techniques for a specific model

In our experiments, we experimented with several fine-tuning techniques, such as LoRA, Prefix Tuning, and Adapter Tuning. In fact, there are more fine-tuning techniques that can be put into practice, such as Prompt Tuning. And we can also consider trying fusion technology. In our opinion, the fusion of technology may greatly improve the training effect. For example, Prefix Tuning with LoRA combines LoRA with prefix fine-tuning to adjust the model by adding a trainable prefix before the input sequence. These can be the direction of our future work.

### 2.2.3 Adjust model hyperparameters to adapt to different datasets and explore the best training combination

In our experiments, since we are more focused on implementing advanced fine-tuning methods, our focus is not on adjusting hyperparameters. During the experiment, we also unified the hyperparameters to ensure that the final results are not affected by changes in hyperparameters. However, in the actual fine-tuning process, it is very necessary to adaptively adjust the hyperparameters. Later, when implementing advanced fine-tuning techniques, we will also try to adjust the hyperparameters and find the best parameter combination.

## 2.3 *Efficiency Improvement*

### 2.3.1 Simplify model

Since one of our research goals is to efficiently implement fine-tuning technology on large models with limited resources, it is necessary to simplify the model. This has a very significant effect on improving training efficiency. For tasks in a specific domain, simplifying the model is beneficial to improving the adaptability of the model to tasks in a specific domain and improving training efficiency.

### 2.3.2 Apply distributed training

In future work, we will continue to apply distributed training. We can use distributed training to improve computing efficiency, model performance, fault tolerance, stability, flexibility and scalability, and maximize the use of limited resources. Using distributed training, we can use larger data sets to train faster, which will greatly improve work efficiency and obtain more constructive results.

### 2.3.3 Use more computing resources

In the training of LLMs, sufficient computing resources are the guarantee of training. It is also

the guarantee of fine-tuning effect. In future work, we will not be restricted by limited computing resources. We will use more computing resources to train and evaluate the model and improve the practicality of the fine-tuning model.

## *2.4 Evaluation Methods Enhancement*

### *2.4.1 Introduce more evaluation metrics*

In our experiment, since our work is to classify the sentiment of text in the financial field, the evaluation parameters used are only parameters related to such evaluation classification, such as Accuracy and Precision. In future work, we will introduce different data sets, such as candlestick charts, which requires us to introduce evaluation parameters specifically for images (k-line charts), such as BLEU, into our evaluation parameter set.

### *2.4.2 Improve the generalization ability of the model*

In future work, we still need to improve the generalization ability of the model. In the limitations of our current work, we mentioned that our model only performs sentiment analysis on text data in the stock market. In the financial field, first, there are not only stock markets, but also many other markets such as bond markets. Secondly, it is not only text data that needs to be predicted. For example, predicting the trend of K-line charts is very meaningful in the financial market. Even if predicting the trend sounds unrealistic, from the perspective of quantitative investment, predicting the stock trend is precisely the key to quantitative investment. Therefore, in future work, improving the generalization ability of the model and making the fine-tuned model more adaptable to various branches in the financial field is one of our future work goals.

### *2.4.3 Improve the robustness of the model*

Improving the robustness of the model and making it stable on different datasets is also one of our future goals. This will also help the fine-tuned model adapt to more types of datasets.

### *2.4.4 Experience more practical evaluation*

Use real-world data to test the model. With the development of society and the advancement of technology, financial texts are not static. For example, 20 years ago, the keywords representing positive performance in the US stock market might have been Coca-Cola and Cisco; ten years ago, the keywords became Apple, Google, and Amazon. Today, the keywords might be Nvidia, Tesla, and meta. Therefore, testing the model with up-to-date data can

improve the practicality of the model in real life, which is also the goal of all fine-tuning models in specific fields - application to real life.

## *2.5 Domain Adaptability Improvement*

### *2.5.1 Cross domain adaptation*

After fine-tuning the financial field relatively perfectly, we can expand it to other specific fields. For example, in the medical field. We know that for the diagnosis of diseases, we can also divide them into three categories: positive, negative, and neutral, which is very consistent with text sentiment analysis in the financial field. So, we can adapt the model to diagnostic text analysis by adjusting some parameters and implementing some fine-tuning techniques, thereby contributing to the development of the medical industry.

### *2.5.2 Adaptive learning*

Explore adaptive learning methods to enable the model to dynamically adapt to various changes in financial sentiment.

## *3. Conclusion*

In this project, we successfully implemented three advanced fine-tuning methods: LoRA, PrefixTuning, and AdapterTuning on GPT2 and FinBERT models. Except for prefixTuning, which is not applicable to the GPT2 model, other fine-tuning technologies have achieved significant performance. promote. This is very important for the development of the field of financial text sentiment analysis. This proves that by fine-tuning large models using advanced methods, the usability and effectiveness of the model in the field of financial text analysis can be effectively improved. This research demonstrates the potential of FinBERT and GPT2 in the financial field, and also demonstrates the importance of fine-tuning technology in specific fields, and also lays the foundation for continued research and development in the future.

## *F. Engineering Practices*

In this section, we detail the engineering practices followed to perform the experiments.

Here are the introduction for the experimental process in engineering practice

### *1.Experimental Environment*

In our experiments, we used Jupyter Notebook's Python to write tests, visualize results, and have built-in documentation functions to record detailed experimental data, and perform necessary result interpretations based on the visual results. Our entire experimental process can finally be

reproduced in Jupyter Notebook. Because we performed LoRA, Adapter tuning, and Prefix tuning on Finbert and GPT-2, which required powerful computing power, and we did not have enough funds to purchase equipment, we used Google Colab's free T4 GPU access to assist experiments, which helped us Transformer model training.

Maintenance of our code base is done on Github. Github's private function ensures the security of our code. Team members conduct internal access and code library management through collaboration and version control systems. Experimental data and results can be shared with every member. This greatly helps our team collaborate and ensure code consistency.

We also use Jira for project management. Work tasks are often adjusted based on experiments. We use the Jira system to track the progress and errors of test tasks. Each member can receive their own work tasks immediately and check the project progress regularly. Jira's high transparency and task tracking capabilities improve our team's coordination efficiency and overall project progress management.

## 2. Libraries and Tools

We used a hugging face Transformers library as our main tool. The Transformers library provides a large number of pre-trained models and natural language processing tools, which have saved us a lot of time when fine-tuning our models. We also use the PEFT library for more complex fine-tuning of LoRA, Adapter, and Prefixing. Huggingface's Datasets library was used for data processing and Pytorch was used for model training. sklearn's accuracy score, precision\_recall\_fscore\_support and confusion -- matrix were adopted for model evaluation to help us evaluate the performance of the model in various aspects. pandas and numpy were used for data manipulation and analysis, and experimental results were visualized by matplotlib and seaborn to show the training process and experimental results.

## 3. Data Preparation

In this experiment, we experiment on two data sets:

**Tweet Financial Dataset:** This dataset contains financial tweets where we rate financial events as negative, neutral, or positive based on public sentiment towards financial times. Because the tweets are sent on the social media platform, where the language is informal and abbreviated, we preprocessed them by using datasets library at Hugging Face, marking the text as a sequence, filling it to a uniform length, and converting the text labels to values, facilitating the training of our model.

**Financial Phrase Bank Dataset:** The dataset consists of financial phrases extracted from various formal news reports, sorted by mood. Formal news reporting is more standard and



structured. We also used the datasets library for pre-processing, and the format matches the transformer model. The Financial Phrase Bank database presents more systematic financial phrases, providing a new perspective for sentiment analysis. Compared with the tweets financial dataset, this dataset focuses more on formal language and rich reported content.

### Baseline Models

Two baseline models were implemented for comparative analysis:

1. **FinBERT**: We modified the BERT model based on Tweet Financial Datasets
2. **GPT-2**: The pre-trained GPT-2 model is fine-tuned by the Financial Phrase Bank dataset for classification.

### Fine-Tuning Methods

1. **Baseline Fine-Tuning**: Both the FinBERT and GPT-2 models were fine-tuned using conventional practices. Tokenization was done using AutoTokenizer for FinBERT model and GPT2 Tokenizer for GPT-2 model respectively, with padding and truncation up to a maximum length of 128 tokens. The training configurations encompassed batch sizes of 8, a total of five epochs and learning rates of  $2e-5$  for FinBERT and  $2e-4$  for GPT-2. AdamW optimizer with weight decay set to 0.01 was employed.
2. **LoRA Fine-Tuning**: Low-Rank Adaptation (LoRA) involves the addition of trainable low-rank matrices to transformer layers. In this case, rank=32, alpha=64, dropout rate=0.1 were used in FinBERT configuration. Similar setup was repeated in attention layers targeting low-rank adaptation in GPT-2.
3. **Adapter Fine-Tuning**: This approach entails the inclusion of lightweight bottleneck adapters on transformer layers. The "pfeiffer" config where reduction factor is equal to 16 was taken as an instance. The adapter layers were trained independently from the main model weights.
4. **Prefix Tuning**: Prefix tuning includes learnable prefix tokens inserted into the input sequence. The setting had ten virtual tokens with enabled prefix projection. All virtual tokens got optimized during fine-tuning process.

## 4. Evaluation Metrics

We selected Accuracy, F1 score, Precision and recall to evaluate the model.

**Accuracy**: The ratio of correctly predicted instances to the total number of instances provides a measure of the overall correctness of the model. But in sentiment analysis, for imbalanced data sets, accuracy can be misleading because one sentiment category may dominate.

Precision: Ratio of true positive predictions to total positive predictions, a measure of how reliable the model is at predicting an emotion, thereby minimizing false positives.

Recall is called sensitivity. The ratio between actual positive instances and true positive predictions represents the recall rate for each model.

F1 score is harmonic average of accuracy and recall rates, providing a balanced indicator, is particularly suitable for unbalanced datasets, ensuring that both false positives and false negatives are considered.

In conclusion, these indicators ensure that the model not only accurately predicts the dominant category, but also effectively captures the nuances of a few categories. By using these metrics, the performance of the sentiment analysis model can be comprehensively evaluated, ensuring that it accurately identifies emotions and minimizes misclassification.

## 5. Resource Monitoring

To ensure that computing resources are used efficiently, we monitor resource usage during training. We use the psutil library to record metrics such as CPU usage, memory usage, and GPU memory usage. With this monitoring, we are able to optimize the training process and detect and resolve any potential bottlenecks in a timely manner.

## References

- Araci, D., 2019. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*.
- Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Floridi, L. and Chiriatti, M., 2020. GPT-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30, pp.681-694.
- He, R., Liu, L., Ye, H., Tan, Q., Ding, B., Cheng, L., Low, J.W., Bing, L. and Si, L., 2021. On the effectiveness of adapter-based tuning for pretrained language model adaptation. *arXiv preprint arXiv:2106.03164*.
- Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L. and Chen, W., 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.
- Lagler, K., Schindelegger, M., Böhm, J., Krásná, H. and Nilsson, T., 2013. GPT2: Empirical slant delay model for radio space geodetic techniques. *Geophysical research letters*, 40(6), pp.1069-1073.
- Lester, B., Al-Rfou, R. and Constant, N., 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.
- Li, X.L. and Liang, P., 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.
- Pfeiffer, J., Rücklé, A., Poth, C., Kamath, A., Vulić, I., Ruder, S., Cho, K. and Gurevych, I., 2020. Adapterhub: A framework for adapting transformers. *arXiv preprint arXiv:2007.07779*.
- Financial\_Phrasebank dataset: [https://huggingface.co/datasets/takala/financial\\_phrasebank](https://huggingface.co/datasets/takala/financial_phrasebank)
- Twitter financial: <https://huggingface.co/datasets/zeroshot/twitter-financial-news-sentiment>