FINE TUNING LLMs FOR CONTEXT-AWARE DIALOGUE SUMMARIZATION AND HUMAN ALIGNED RESPONSES VIA RLHF

THESIS REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
ARTIFICIAL INTELLIGENCE

Submitted by

GAPESH KUMAR (23/AFI/18)

Under the supervision of

Prof. SHAILENDER KUMAR

DEPARTMENT OF COMPUTER SCIENCE
DELHI TECHNOLOGICAL UNIVERSITY

MAY, 2025

DEPARTMENT OF COMPUTER SCIENCE
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, **Gapesh Kumar**, Roll No – **23/AFI/18** student of M.Tech (**Artificial Intelligence**),hereby declare that the project Dissertation titled "**Fine Tuning LLMs for Context-Aware Dialogue Summarization and Human Aligned Responses via RLHF**" which is submitted by me to the **Department of Computer Science**, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi **Candidate's Signature**

This is to certify that the student has incorporated all the correction suggested by the examiners in the thesis and the statement made by the candidate is corrected to the best of our knowledge.

Signature of Supervisor **Signature of External Examiner**

DEPARTMENT OF COMPUTER SCIENCE

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled "**Fine Tuning LLMs for Context-Aware Dialogue Summarization and Human Aligned Responses via RLHF**" which is submitted by **Gapesh Kumar**, Roll No's – **23/AFI/18**, **Artificial Intelligence(AFI)** ,Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge, this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi **Prof. Shailender Kumar**

Date: 31.05.2025 **SUPERVISOR**

DEPARTMENT OF COMPUTER SCIENCE

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

We wish to express our sincerest gratitude to **Prof. Shailender Kumar** for his continuous guidance and mentorship that he provided us during the project. He showed us the path to achieve our targets by explaining all the tasks to be done and explained to us the importance of this project as well as its industrial relevance. He was always ready to help us and clear our doubts regarding any hurdles in this project. Without his constant support and motivation, this project would not have been successful.

Place: Delhi Gapesh Kumar

Date: 31.05.2025

Abstract

Large language model (LLM) optimization on a task is based on tuning it, which saves costs in resources. Training models or instruction tuning on pairs of instructions and completions makes them follow human directions in the right manner. Complete tuning remains computationally expensive, however. There have been a number of recent parameter-efficient fine-tuning (PEFT) methods that assist in resolving this problem. It remains very hard to align model outputs with human preferences, however.

In this current work, we tried instruction fine-tuning and PEFT techniques such as Low-Rank Adaptation (LoRA) to fine-tune pre-trained LLMs on a given task using structured training data and efficient tuning of a portion of model parameters. To ensure contextual appropriateness while improving response alignment with human expectation, we incorporated Reinforcement Learning from Human Feedback (RLHF) during fine-tuning.

Our results indicate that while PEFT approaches significantly minimize computational and memory expense without any loss in performance, instruction adaptation actually enhances model task conformity. RLHF also prevents the model from providing out-of-context responses thus ensuring that responses are uniform and human-aligned.

Observations of this work show that highly specialized and resource-effective LLMs may be built by combining PEFT, instruction tuning, and RLHF. These methods offer a rational and scalable way to fine-tune, thus enhancing the usefulness and flexibility of LLMs for a wide range of other applications.

Contents

List of Tables

List of Figures

List of Symbols/Abbreviation

PEFT	Parameter-Efficient Fine-Tuning
LoRA	Low-Rank Adaptation
PPO	Proximal Policy Optimization
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
RLHF	Reinforcement Learning from Human Feedback
KL Divergence	Kullback-Leibler Divergence, a measure of distribution difference
LLM	Large Language Model
$V_\theta(s)$	Estimated value of state s under parameters θ
$\hat{R}_t$	Estimated return at time t
$L^{\mathbf{value}}(\theta)$	Value loss function with parameters θ
$L^{\mathbf{policy}}(\theta)$	Policy loss function with parameters θ
$\pi_\theta(a_t\|s_t)$	Policy function: probability of action a_t in state s_t under parameters θ

Chapter 1

INTRODUCTION

1.1 Overview

Its goal is to introduce the various techniques of PEFT and instruction tuning in large language models, particularly comparing how they improve both the efficiency and effectiveness of the method used to train the models. The process of instruction tuning is special among fine-tuning approaches. Training these language models involves using smaller, focused databases made for the explicit instruction following task, so no general instruction following is needed. While LLMs come with much existing expertise, they still need to be adjusted to perform well in challenging areas. Information developed during pre-training is used to help an expert dog act according to its trainer's expectations. Adapting a model helps it work well in the situations it is designed for. By adapting their way of interpreting and performing lessons, they are able to face many complicated needs in different tasks.

PEFT techniques, like LoRA., Adapters and Prefix-Tuning, could lead to much more efficient instruction tuning compared with other alternatives. Such tools save a lot of computer resources and memory for the purpose of fine-tuning. The main difference is that engineers used to fine-tune every component of the model, but with this technique, only some parts are moved, the rest remains unchanged. Low-rank decomposition which is how LoRA works, tunes a select number of layers. On the other hand, Adapters are very simple and can be optimized inside the model. Still, prefix-tuning changes the input sentences to manage the model without touching the weights that matter the most. That is to reach the outcomes you want. Using these techniques let users make their models

harder while ensuring they remain relevant and useable within machine learning systems with fewer resources.

At present, Reinforcement Learning from Human Feedback is a leading technique for fixing large language models to human values, tastes and expectations. RLHF lets people improve LLMs by constantly providing feedback for increasing how helpful, safe and useful each model becomes.

More and more, reinforcement learning (RL) is being used as a solution that complements PEFT and individual teaching. If given feedback from interaction, large language models can learn without using outside datasets or commands. It is made possible because of reinforcement learning methods.

According to the work, the application of reinforcement learning might transform a series of PEFT methods and education practices. Because of these ways, LLMs can work in more complex industrial cases, making AI use easier by making LLMs adaptable, efficient and scalable. As a result, LLMs are able to handle difficult situations.

1.1.1 Motivation

It is necessary for large language models to offer enough training to each person and for enough memory and computation to be provided. Their capabilities must be applied and cannot be ignored. All the model parameters are regularly updated throughout fine-tuning which starts at the training stage. Training is the first part of this operation. For example, caching model weights, optimizer states, gradients, forward activations and temporary memory uses up quite a lot of computing space and memory. All of these show ways that memory can be affected. A temporary memory is also necessary as well. Having a memory in this environment is clearly better when it is temporary. When it comes to consumer electronics, this approach does not work for products that are needed for hundreds of gigabytes of data.

Chapter 2

LITERATURE REVIEW

2.1 Introduction

Much improvement has been made in large language models or LLMs, with increasing usage and variety in different applications. Many more applications are now being created with LLMs. The way methodologies have grown over the years is why the situation appears as it does now. I cover what has been found from big research studies that have gone before. The assessment focuses on how evaluations are carried out, what outcomes are found and what weaknesses are noted within the field.

2.2 Background and Research Gaps

Tuning such models for a specific need is now central to improving NLP processes. This happens because more people are using pre-trained language models, like BERT, GPT, RoBERTa and T5. It is very challenging for finetuning to be done in a realistic way because of the high resources it requires for processing and memory. Such a challenge has led to the invention of Parameter-Efficient Fine-Tuning techniques. These include Adapters, LoRA, BitFit, Prompt Tuning and Delta-Tuning which combine several prior methods. They let us improve a specific part of the model's parameters and it results in exactly the same performance as a full-fine-tuning process but takes less effort to train. Experts in medicine have looked at PEFT for different uses, including normal language processing, learning causes, code writing and bringing together individual cognitive signals. According to these studies, PEFT is both flexible and relatively inexpensive.

Promising studies are underway, but there are real barriers in PEFT research at this time. The lack of standard tools to measure achievement in most disciplines greatly hinders the effort to measure progress in the same way. Even so, our view of how PEFT systems converge has only started to develop. While studies have shown PEFT techniques work well, how they can be used for continued learning and multitasking is not yet well known. Lastly, relatively few studies focus on finding ways to fine-tune models just for creating code. In addition, although external guidance such as using gaze as a reference or bringing in causal inference is possible, it is still non-standard and has only a small impact right now. Besides, although Delta-Tuning and Quantum-PEFT are promising developments, we need to test them under real conditions to make sure they help.

2.3 Key Insights from the Literature

The papers explain that by using fewer adjustments, PEFT techniques can reach 90-95% of the results a larger-scale approach would have, lowering the need to change many model parameters. It leads to much less operation of computers and other needed machines. For example, Delta-Tuning[3] excels on over a hundred NLP tasks while using fewer parameters.

LoRA and Adapters[4] which are examples of PEFT, have been demonstrated to preserve model performance but require much less time and memory to fine-tune. Furthermore, including additional reasoning or cognitive inputs has demonstrated that systems perform better and show fewer problems in situations where they are not trained for or when they encounter different data.

The importance of PEFT has been confirmed by research in code generation and the studies suggest that PEFT methods need to be more refined for particular uses. In general, PEFT provides a straightforward way to adjust large language models that is most helpful in situations where resources are limited or the needs keep changing.

2.4 Evaluation Metrics and Findings

Table 2.1: Summary of Literature Review on Instruction Fine-Tuning and PEFT Methods

Research	**Model Used**	**Fine-Tuning Method**	**Evaluation Metrics**	**Metric Scores**	**Research Gap**
Parameter-efficient fine-tuning of large-scale pre-trained language models (2023)	Various large PLMs (e.g., BERT, GPT variants)	Delta-tuning (unified PEFT framework including Adapters, LoRA, Prompt Tuning)	Accuracy, F1, Task-specific benchmarks (100+ NLP tasks)	Comparable or better than full fine-tuning with <10% parameters updated	Lack of standardized benchmarks across diverse domains; need for better theoretical understanding of convergence in PEFT
Parameter-Efficient Transfer Learning for NLP (2022)	Adapters	Adapters	Accuracy, BLEU	Adapter tuning achieves within 0.4% accuracy of full fine-tuning on GLUE - Adapter parameters add 3.6%	Full fine-tuning is resource-intensive; feature-based methods underperform. Adapters offer near state-of-the-art results with minimal parameters, enabling scalable multitask learning.
Fine-Tuning Pre-trained Language Models for Robust Causal Representation Learning (2024)	Pre-trained LMs (BERT, RoBERTa)	Standard fine-tuning with causal front-door adjustment	Domain generalization accuracy, OOD robustness	Improved OOD accuracy by 5–10% over baseline fine-tuning	Integration of causal inference with PEFT remains underexplored
Fine-Tuning Pre-Trained Language Models with Gaze Supervision (2024)	Transformer-based LMs	Fine-tuning with auxiliary gaze supervision signals	GLUE benchmark scores, accuracy, F1	2–3% improvement on GLUE over baseline fine-tuning	Scalability of cognitive signal integration to larger models and datasets
A Comparative Study of PEFT Methods for Python Code Generation (2023)	Code generation models (e.g., Codex, CodeT5)	PEFT methods (Adapters, LoRA, BitFit) vs Full Fine-Tuning	BLEU, Code correctness, GPU memory usage	PEFT methods achieve 90–95% of full fine-tuning performance with 5–10% parameters trained	Need for better PEFT methods tailored for code generation tasks

Chapter 3

METHODOLOGY

3.1 Objective

This report mainly examines how we execute the fine-tuning process, since that is our main goal with the use of Flan-T5. In other words, we try to accomplish this summarization by carrying out two different methods; Complete (Full) fine-tuning and Parameter-Efficient Fine-Tuning (PEFT). This project has the goal of boosting the model's ability to do summarization by fine-tuning it and then measuring how such techniques improve the model's performance using the ROUGE score to compare before and after. Moreover, the experiment uses two conversation data corpora, SAMSum and DialogSum, to test and observe the model's efficiency with real dialogs and determine its suitability for practical use.

3.2 Methodology

3.2.1 Dataset Preparation

The datasets used for fine-tuning the Flan-T5 model are SAMSum and DialogSum.

SAMSum Dataset

The data contains more than 16,000 examples of summary-based dialogues. Because the dialogues include normal language as well as more formal expressions, it becomes possible to understand a wide variety of subjects that appear in everyday life. Messages have emoticons, informal expressions and now and then even misspelled words meant to resemble real conversations. A short summary of the most important points in the

discussions is included in the third-person-written abstracts. The information contained in the rich content dataset can be accessed non-commercially under a particular license and was chosen and annotated by linguists specializing in the English language.

DialogSum Dataset

In whole, the DialogSum dataset offers a large set of dialogues that are all summarized. 1,000 of the 13,460 dialogues have been kept aside for testing. All of the discussions in the dataset are separated into specialized sets that help with testing, validation and training. In the example we looked at earlier, we saw a doctor and patient planning an outpatient visit and advising the patient on ways to quit smoking. Considering how extensive this dataset is, it includes both formal and informal dialogue to keep things interesting. Also, to be ready for fine-tuning, data is tokenized and pre-processed with the Transformers library, even if some of this work is done beforehand.

Preparing the Dataset for Reinforcement Learning from Human Feedback

At this specific point in the process, a large and significant list of prompt texts is intentionally designed, with every one of these prompts individually focused on the same task. A specific large language model, better known by the name LLM, then undertakes the task of producing numerous different completions for every single prompt in our large set of prompts. These different completions produced by the LLM then become the main point of reference for determining what necessarily amounts to human feedback in this given context.

3.2.2 Model Selection and Setup

Pre-training Flan-T5 was the way Google did the summarizing task. Autopilot utilizes the AutoModelForSeq2Seq class from the Transformers package to load the tokenizer and model. We utilize the original model as the baseline to compare against other versions. There are two images presented here, figure 3.1 and figure 3.2.

The Flan-T5 model has a staggering number of more than 250 million parameters, which represents a significant amount of its ability and complexity. During the study, the impact of training a fraction of a model's parameters, referred to as Parameter-Efficient

Fine-Tuning (PEFT), will be contrasted and evaluated against the impact of training all the parameters via full fine-tuning. The intricate and complex information regarding the two readings can be seen depicted in figure 3.3.

Large Language Models

Large Language Models (LLMs) are advanced machine learning techniques that ingest, generate, and process human language in bulk. They serve as the foundation for natural language processing (NLP) functionality such as text generation, translation, summarization, and question answering.

Architecture and Components of LLMs

- **Transformer Architecture:** Self-attention transformers were a replacement for the then-used sequential models, e.g., RNNs. This was substituted with the sequential models. Because this is not anything out of the norm, the transformers could very well fill these functions. Simultaneous input sequence processing and the storage of long-range correlations are now in the realm of probable possibilities at this point.
- **Key Components of a Transformer-based LLM:**
  - **Embedding Layer:** The approach taken here is to map input tokens, potentially words or subwords, to high-dimensional dense vectors that preserve semantic and grammatical information. We refer to this operation as tokenization.
  - **Self-Attention Mechanism:** Through dynamic weights to every token in relation to the rest of the sequence, the model is able to attend to input elements pertinent to the issue at hand. This enables the model to attend to the elements actually required by the immediate challenge.
  - **Feedforward Neural Networks:** At the end of the day, the sequence of attracting attention outputs finally results in abstraction extraction at higher levels throughout the process.
  - **Encoder and Decoder:**

* **Encoder Only:** The Transformer encoder stack is the only component in encoder-alone versions. The encoder uses bidirectional self-attention to process the full input sequence, attending to all tokens before and following it. The model can extract extensive contextual information from the complete sequence.

  Models are trained using Masked Language Modeling, which masks certain input tokens and predicts them based on context.

  Encoder-only models excel in text classification, named entity identification, sentiment analysis, and other discriminative tasks that require deep text knowledge and representation. **Example: BERT, RoBERTa, ELECTRA**. Encoder-only models produce embeddings or predictions for fixed input sequences, not coherent token-by-token sequences for autoregressive text production.

* **Decoder-only:** Decoder-only models employ the Transformer decoder stack. Masked self-attention restricts every token to observe only left-to-right tokens seen so far. This is an autoregressive model, which predicts a token once it has observed the preceding tokens.

  The model is trained with Causal Language Modeling to forecast the next token given prior tokens. Decoder-only models are used to leverage language modeling, conversational systems, code generation, and creative writing. Text is generated incrementally, one token at a time. **Example:** GPT family (GPT-2, GPT-3, GPT-4), LLaMA, and BLOOM. Decoder-only models can't be applied for tasks requiring bidirectional context or full understanding of the entire input sequence as they possess unidirectional attention.

* **Encoder-Decoder:**Encoder-decoder models are hybrids of Transformer encoder and decoder stacks. Encoding starts with bidirectional processing of the input sequence in an effort to produce a contextual representation. The decoder generates the output sequence autoregressively using tokens produced previously and encoder output.

  Sequence-to-sequence training models such as denoising or span corruption, the model produces a target sequence from corrupted or altered in-

put.

These are well-suited for machine translation, question answering, and abstractive summarization where input and output sequences differ in structure or length. Decoupling the decoder and the encoder, the model is capable of fully understanding input prior to generating a structured output.**Example:** MarianMT, BART, FLAN-T5.

Because they are composed of two components, encoder-decoder models need more computational power while training and predicting but handle difficult sequence-to-sequence tasks better.

Training Process

- LLMs are first trained on huge corpora constructed from books, web pages, and code bases. Huge corpora of billions of words are taken from a great variety of sources. Few corpora are employed for instructing LLMs. Huge corpora are employed for training. LLMs must begin with this training before performing further complicated tasks.

- **Self-Supervised Learning Objectives:**

  - **Masked Language Modeling (MLM):**Masked Language Modeling (MLM) is one of the self-supervised pretraining techniques employed to pretrain large language models, notably Transformer encoder-based models such as BERT and RoBERTa. MLM strives to learn word-contextual relationships by filling in the missing or masked words in sentences.

    Some of the tokens at random positions (usually 15%) are substituted with a special [MASK] token during MLM pretraining. The model is asked to predict the masked tokens contextually from both sides of the word. The model develops an in-depth sense of linguistic context, semantics, and syntax through bidirectional attention.

    Example: *"The cat sat on the [MASK]"* Context word assists the model to predict "mat" as masked. MLM bidirectional technique enables all of a sequence's words to be processed at once, with better handling of subtlety of meaning and interdependence compared to regular left-to-right language

models.

Cross-entropy loss between the predicted and masked tokens is the training objective which seeks to minimize it. Surprisingly, only masked tokens compute loss, with unmasked tokens acting as context.

- **Causal Language Modeling (CLM):** Causal Language Modeling (CLM) or autoregressive language modeling is another well-known training plan of decoder-only Transformer models like GPT models.

  Next token generation in a sequence is done by CLM model from the previous tokens in strict left-to-right order. The one-way model predicts the distribution of the next word given the previous words and produces text one token at a time with contextual directions.

  Example: *"The cat sat on the"* There are past tokens with only words that are used to predict "mat" in the model. The objective is to maximize next token probability to be correct for all the training corpus.

  As it generates text sequentially, CLM models can be used in story composition, chatbots, and code writing. Unidirectional models will not be able to use future context during training, which may affect understanding of some examples of language.

- **Fine-Tuning:** After the first pretraining, when an LLM is trained on common patterns from huge databases, occasionally the model must be fine-tuned to specialize well in a specific application or class. Fine-tuning pre-trained parameters on labeled data for the target use enables more accurate results, usability, and performance.

Scale and Capabilities

- As there are more parameters in an LLM, it can learn and identify very complex patterns and language nuances much more effectively. The scale enables multiple breakthroughs in natural language generation and understanding:.

- **Larger models exhibit:**

  - More advanced models have a better sense of world knowledge, semantics, and syntax and are capable of responding to ambiguous or unclear questions.

With this better understanding, they create grammatically correct, contextually meaningful, and semantically rich content.

- **Zero-shot learning:** Most of the LLMs possess the zero-shot learning ability, which is quite remarkable. The model can accomplish new tasks through task-specific training and fine-tuning. A big LLM can provide answers, translate, or summarize text if it is supplied with only its pretraining data and patterns. Task-specific model development and labeled data are greatly reduced through the use of this ability.
- **Few-shot learning:**Apart from zero-shot capability, large LLMs are also astounding when it comes to few-shot learning, learning to perform new tasks after seeing only a few examples in the prompt. In-context learning enables the model to pick up task constraints and provide correct responses without gradient updates or retraining. Few-shot learning adds even greater flexibility and runtime prototyping, making users able to modify model behavior at runtime with low data.
- Large models produce more coherent text across longer passages with thematic coherence and logical coherence. They are able to create original and delicate content, such as creative writing, heavy reasoning, and poetry, that smaller models cannot do. This is because creativity emerges since the model has heavy exposure to varied linguistic styles and knowledge in training.

Text-to-Text Summarization Using RNNs architecture

Recurrent Neural Networks (RNNs) have successfully been go-to architecture for sequence modeling problems such as text summarization. RNNs read input sequences in order one at a time, having a hidden state that takes in knowledge from past tokens to guide processing for the current token. Sequential dependency allows RNNs to perceive temporal and contextual dependencies in text, which is important while creating well-coherent summaries.

Encoder-decoder approach is quite prevalent in text-to-text summarization. Encoder RNN processes the input sentence word for word and transforms it into a fixed-size context vector that contains the whole input sequence. Decoder RNN produces the summary by generating words one at a time step-by-step from the context vector and previously

generated words from itself.

While helpful, there are some limitations of the traditional RNNs, e.g., no capacity to learn long-term relations with exploding or vanishing gradients, and thus cannot be applied to abstract long or intricate documents. This is addressed by algorithms like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) that overcome some of the limitations by using gate units for maintaining longer memory for a sequence of time.

The parallelization and scalability problems of the RNN-based summary models are caused by sequential computation, therefore leading to slow training and wastage of hardware.

Text-to-Text Summarization Using Transformer architecture

Transformer design is a natural language transformation, especially text-to-text applications like summarization. As opposed to RNNs, Transformers don't process the input sequentially but instead employ a self-attention approach where the model has the ability to give weight to all words of the input sequence at the same time. Parallel processing ability enables the model to capture intricate long-range relationships easily and effectively.

In a standard Transformer-based summarization model, the model itself is a combination of an encoder and a decoder. The encoder converts the input text into a sequence of unbroken representations by feeding a sequence of a number of layers of self-attention and feed-forward networks. The decoder produces the summary using the previously created tokens and the output of the encoder in an attempt to produce contextually relevant and coherent summaries.

Perhaps most notably, the Transformer's multi-head self-attention mechanism, which allows numerous attention "heads" to look at numerous features of the input text in parallel, enables the model to learn abstract linguistic representations such as syntactic structure, semantic meaning, and discourse relations on which good summarization relies.

Most remarkable of Transformer's achievements is multi-head self-attention, where it learns multiple attention "heads" in parallel that pay attention to different parts of the input text. This enables the model to learn very diverse types of linguistic features like syntactic structure, semantic meaning, and discourse relations that are needed for high-quality summarization.

Positional encoding is applied to input embeddings to preserve word order information as the self-attention mechanism is position-agnostic. The model can thus utilize the parallelism of the Transformer while preserving sequence relationship of language in this manner.

Transformer models performed better than RNN-based models in summarization in terms of more processing of long text, higher coherence, and less training time because they are parallelizable. Common Transformer-based models used in summarization are BART, T5, and PEGASUS that achieved novel state-of-the-art performance in abstractive summarization.

Parameter-Efficient Fine-Tuning (PEFT)

Flan-T5 was pre-trained by Google for the job of creating headlines. The AutoModelForSeq2Seq from Transformers is used by Autopilot to load the tokenizer and the model. We look at the first version of the model as a starting reference against other versions. Two illustrations are included here, figure 3.1 and figure 3.2.

The model is very advanced because it has over 250 million parameters. Both methods for converting pre-trained models will be compared in a study: PEFT, where only partial parameters are updated and full fine-tuning, where all parameters are updated. The facts and explanations of both readings are shown in detail in figures 3.3.

Training Process

- If you want the model's weights to stay frozen, your training should only focus on just a few components, leaving the weights fixed in the process. Going after this objective is a legitimate activity.
- The training process is limited to the components that have been used most recently, and as few additional parameters or layers as feasible are introduced into the system.

PEFT Methods Type

- **Selctive Method:** Some specific parameters within a model such as particular layers, can be tuned to improve its accuracy. It may happen to ensure that the

information is accurate enough. Specific jobs find certain techniques useful, although they may not perform as fast or use as much computer memory as others.

- **Reparameterization Methods:**Within these strategies, there is Low-Rank Adaptation (LoRA), considered a major system improvement method. Further techniques that fit this category are covered here as well. Loop Recurrent Attention Revisited (LoRA) helps to improve the accuracy of Loop Recurrent Attention (LRA) by reducing some requirements that must be adjusted. Small models are possible because fewer parameters are used in their creation.

- **Additive Methods:**One approach is to use extra trainable components in the model and not touch the weights included earlier. Consumers may pick from the following two major categories:

  - **Adapter Method:**When we say adapter methods, we are talking about adding more elements to the encoder and decoder of the model. This may also hear these layers called "adapter methods."

  - **Soft Prompt Method:**They are classified as soft prompt techniques; one of them is playing around with the input prompt. These approaches are regularly called soft prompt techniques. In most cases, it includes inserting trainable parameters into the computer prompt embeddings.

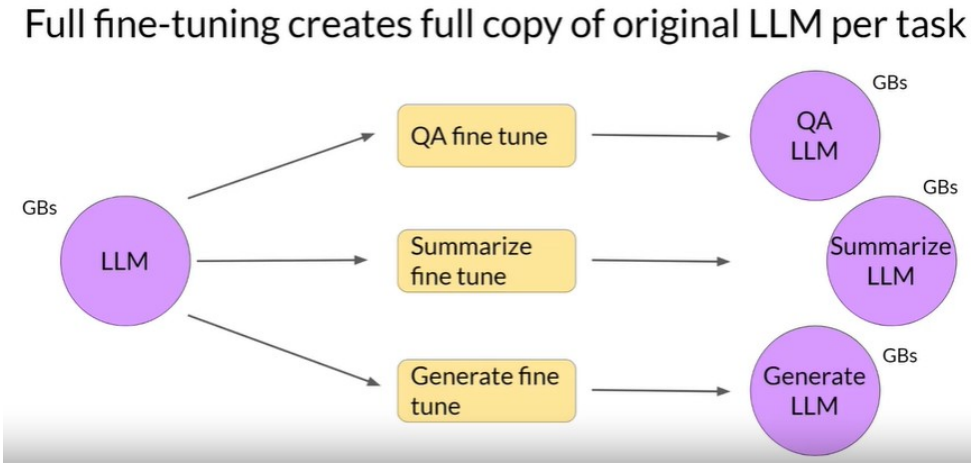

Figure 3.1: Full fine tuning, Available: https://medium.com/@kanikaadik07/peft-parameter-efficient-fine-tuning-55e32c60c799 [1]

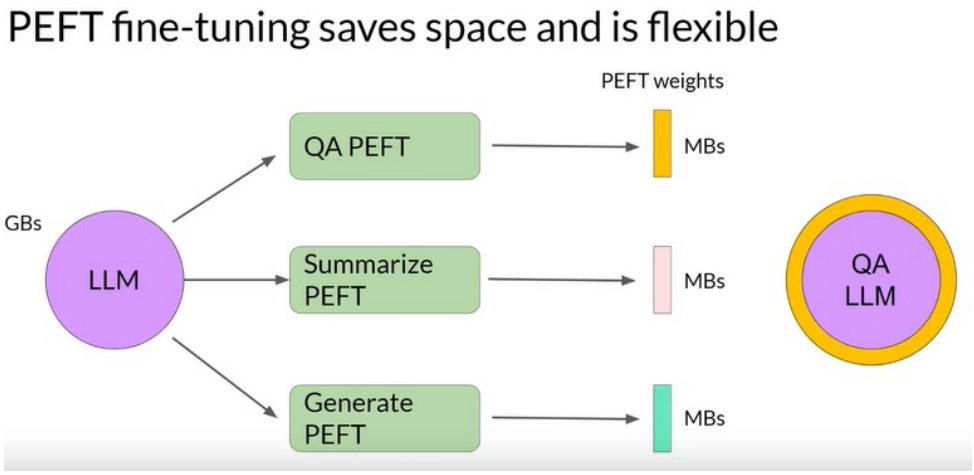

Figure 3.2: PEFT fine tuning, Available: https://medium.com/@kanikaadik07/peft-parameter-efficient-fine-tuning-55e32c60c799 [1]

3.2.3 LoRA (Low-Rank Adaptation)

Applying LoRA on large infrequently updated models leads to a major increase in how well they are fine-tuned. The potential is raised when low-rank matrices are incorporated into the analysis. Variations are made on chosen design variables, with the goal that these alterations will not affect the entire set of parameters. To reach this objective, the organization is applying this tactic. The performance of the model has stayed exactly the same, but fine-tuning it led to a decrease in the number of trainable parameters and, thus, a decrease in the overall count of parameters. **Strategies and Methodologies at Work**

- **Freeze Original Model Weights:** The next step requires training the basic weights first and then locking them. Moving forward to the next step requires you to successfully complete this requirement. For the whole process of fine-grading, the starting weights do not change to guarantee equal results.
- **Introduce Low-rank Decomposition Matrices:**
  - Because we want to consider every type of interest, suggestion is made to add two smaller matrices, A and B. Such layers are widely considered the key to fixing the problem..
  - So that the sizes of their product and weight matrices are compatible, the sizes have been ensured. For this reason, matrices are certain to perform well.

Its accomplishment is made with the aim of having the greatest influence. At present, the assessment is done to confirm that the desired result can be reached.

- It is likely that the original weight matrix holds information that the decomposition matrices do not at first capture, despite their lower rank. Hundreds of thousands of parameters make up the first weight matrix. The rank of all these decomposition matrices is either at URL or below and this is significant.

- **Train Low-rank Matrices:**

  - Low-rank matrices are trained with the help of supervised learning, a type of instruction. In such instruction, teachers use information that matches the work to be learned. During the entire period, the initial model's weights do not change from when they were first explored.
  - Even though only a small part of the matrices is updated, accuracy can be maintained with as few as 10 percent of the usual number of trainable parameters. Lowest values are the only ones updated in the matrices.

- **Inference Process:**

  - During inference, the low-rank matrices multiply to produce a matrix that shares same dimensions as the weight matrix included in the original model. After inference is finished, the observer employs the matrix to correctly interpret the data. When the model is ready, the matrix is called on to predict what the results will be.
  - You then unite the new matrix with the first model's fixed weights so you can perform inference during the present jo

3.2.4 Full Fine-Tuning

When the whole model has been trained using SAMSum and DialogSum datasets, a great deal of fine-tuning needs to be done to achieve proper results. By sending it simple guidelines for summarizing, the model can quickly be improved. Both datasets include these ideas as part of their analysis. The purpose of this is to achieve the preferred results. The

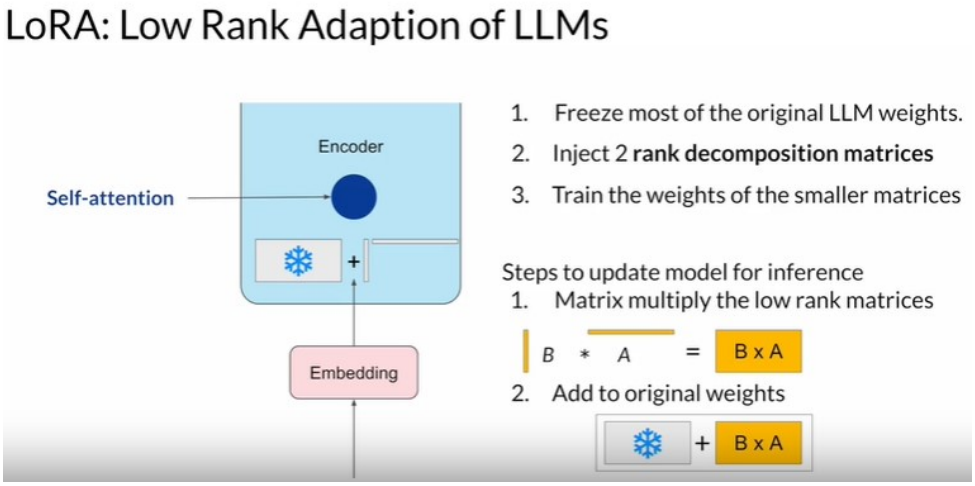

Figure 3.3: LoRA: Low Rank Adaptation of LLMs, Available: https://medium.com/@yash9439/introduction-to-llms-and-the-generative-ai-part-4-parameter-efficient-fine-tuning-peft-83c6dd039787[2]

goals are achieved by preparing and assigning summary tasks. Because of this, the model might create brief versions of what others will say in the presentation.

The model is trained at training time with few CPU resources (a low epoch value) and a partial set of the training dataset (125 samples, for instance). Good performance can only be reached through regular warm-up. The model can be taught because of the way these two characteristics interact.

The TrainingArguments class is utilized to establish and establish the training environment. Training is facilitated in Transformers thanks to the Trainer class within the library. Making sure people receive the help they need is the duty of this class.

When the model has been calibrated, ROUGE is employed to evaluate its capability to summarize treated materials. Once the activity is done, the period when the model is being adjusted begins which includes this assessment. The ROUGE score measures how well prepared summaries function, especially when compared to those that have been produced by common readers.

3.2.5 Instruction Fine Tuning

A qualitative review of the refined model includes comparing summaries made from both the humans and the inceptive Flan-T5 model (zero-shot). Its aim is to help the model gain better knowledge of its own outcome. At this time, this comparison aims to gather information. After looking at both documents, you might notice that the improvement comes from the better written summary.

For vicenary analysis, we depended on ROUGE-1, ROUGE-2, ROUGE-L and the ROUGE-Lsum scores. They are measured to judge how efficiently the model works. Generally, results from the fine-tuned model show that improvements over the original model are quite significant.

3.2.6 Parameter Efficient Fine-Tuning (PEFT) with LoRA

Parameter-efficient fine-tuning (PEFT) technique meets with low-rank adaptation (LoRA) method, offering an alternative approach to fine-tuning. PEFT, or "parameter-efficient fine-tuning," says it all. LoRA is valuable as it allows training a few parameters, thus reducing the computational cost and memory needed. This benefit speaks to one of the numerous advantages of using LoRA. In fact, this is just one of the numerous advantages that come with its application. A method such as this one is especially fitting when resources are limited. There are times when the resources available become limited.

To be put into perspective, the PEFT method considers learning precisely 1.4% of the model parameters. The use of LoRA adapters makes this choice a very viable option. Compared to the adapters, which occupy around 10 megabytes, the fully fine-tuned device is roughly one gigabyte.

One of the most significant aspects of the PEFT approach is the integration of LoRA adapters within the base Flan-T5 model. As a result, the model is fine-tuned by using fewer steps and epochs, which means that the resources used are considerably less. This goal is achieved by utilizing two separate datasets, SAMSum and DialogSum. After PEFT, the model is measured based on ROUGE scores, and the results are compared against the baseline model and the adapted model to the instruction. The two models

are compared. A comparison of the overall performance is done in the correct manner. In essence, this study is to better understand the compromise between computer efficiency and real performance. This particular effort is being done with the view of achieving this.

3.2.7 Reinforcement Learning from Human Feedback

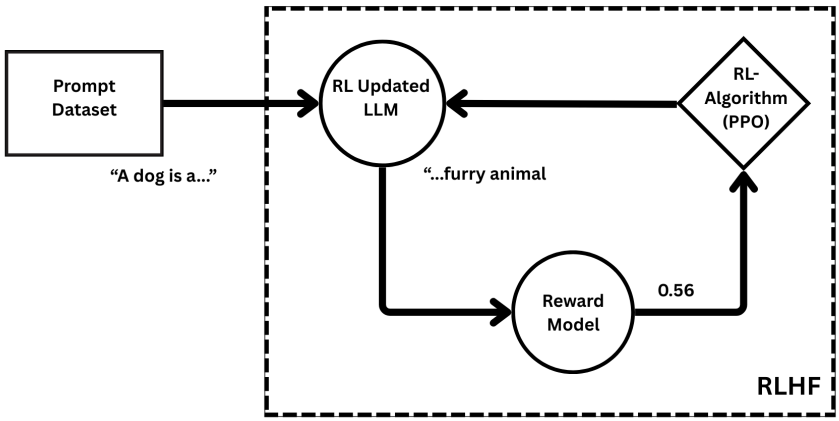

Figure 3.4: RLHF: Reinforcement Learning from Human Feedback

Reward Model

The Reward Model itself is a small language model that has been trained with care to act as a replacement for human judgment. Its main purpose is to assist in aligning large language models (LLMs) with human taste through the provision of autonomous, scalable feedback instead of requiring explicit human evaluation.

The reward model training begins with the generation of a dataset that is rich in prompts, each followed by a number of completions generated by the model itself. Human annotators enter in this step to evaluate and rank these completions on a range of alignment measures, such as helpfulness, harmlessness, honesty, and correctness. These annotator rankings are the supervised training labels that the RM needs. For each prompt with N completions, this gives $\binom{N}{2}$ distinct pairs. Each pair consists of one preferred completion and one less preferred completion; the preferred completion is the first one listed and is given a reward score of 1, and the less preferred completion is given a score of 0.

This organized format is attractive to the reward model's input demands, enabling it to learn from relative preferences.[5].

Having these ranked completions identified, the reward model is subsequently trained on supervised learning to predict which completions humans like. This converts subjective human preference into a quantitative measure, allowing the reward model to give scalar reward scores to fresh outputs produced by the model.

Reinforcement Learning for Fine-Tuning Language Models

Once the Reward Model (RM) has been trained, it is then employed to fine-tune a base language model through reinforcement learning techniques. The most commonly applied algorithm in this procedure is Proximal Policy Optimization (PPO), a technique that is well known for its stability and efficacy in policy gradient techniques. The aim of this process is to enhance the behavior of the base model so that it produces completions optimized to maximize rewards specified by the RM—taking the model closer to meeting human preferences.

Reinforcement Learning Process

The reinforcement learning process proceeds in iterative cycles:

- **Prompt Sampling:** A prompt is sampled from a dataset.
- **Completion Generation:** The current version of the language model generates a response (completion) for the prompt.
- **Reward Assignment:** The generated response is passed through the reward model, which assigns a scalar reward based on alignment with human preferences (e.g., helpfulness or harmlessness).
- **Model Update with PPO:** Using the PPO algorithm, the model's weights are updated to increase the likelihood of generating completions with higher rewards. PPO constrains how much the model's policy can change in a single update, thus ensuring stable and reliable training.

Proximal Policy Optimization

Proximal Proximal Policy Optimization (PPO) is one of the strongest reinforcement learning algorithms, which is heavily used to train large language models (LLMs) for alignment purposes, such as Reinforcement Learning from Human Feedback (RLHF) and Constitutional AI. In each of these uses, the primary goal of PPO is to optimize the conduct of a language model to make its outputs more in line with human preferences. These preferences are generally quantified with a trained reward model (RM), which serves as an approximation of human judgment by providing scalar reward values to the generated responses by the model.

PPO is part of the policy gradient approaches, designed to adjust the parameters of a policy to maximize the expected rewards by adjusting the probability distribution of possible actions. Among the features of PPO is its excellent training stability, which is ensured by the use of a "trust region." This concept restricts the magnitude of policy updates, so every iteration contributes small changes. Specifically, PPO uses a clipping mechanism that restricts how much the probability of taking some actions can vary, which in turn prevents huge, destabilizing updates and provides a smooth and stable learning process.

Loss Functions in Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) uses a compound loss function with three dominant elements: the loss of the value function, policy loss, and entropy loss. Each of them has a significant role to play in facilitating stable and effective training of reinforcement learning models.

- **Value Function Loss:** This part predicts the accuracy of the value function $V_\theta(s)$ to estimate expected rewards. It employs loss estimated as the squared error between approximated value $V_\theta(s_t)$ and empirical return $\hat{R}_t$, as demonstrated in the equation:

$$L^{\text{value}}(\theta) = \left(V_\theta(s_t) - \hat{R}_t\right)^2 \tag{3.1}$$

  Here, $V_\theta(s_t)$ represents the predicted value of the state s_t, and $\hat{R}_t$ denotes the empirical return or advantage estimate.

- **Policy Loss (Clipped Objective): Policy Loss:** Policy loss is due to enhancing the action choice of the model to optimize the learning expectation. PPO employs a clipped surrogate objective to support stable learning. The loss is as follows:

$$L^{\text{policy}}(\theta) = \min\left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right) \tag{3.2}$$

  where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between the new and old policies, and $\hat{A}_t$ is the estimated advantage. The clipping prevents large policy updates, ensuring the training remains within a stable range.

- **Entropy Loss: Entropy Loss:** To promote discovery and avoid early convergence to poor policies, PPO receives an entropy bonus. This penalty discourages reliance on the action distribution of the policy and favors more diverse action choices. The loss in entropy is expressed as:

$$L^{\text{entropy}}(\theta) = \left[-\sum_a \pi_\theta(a \mid s_t) \log \pi_\theta(a \mid s_t)\right] \tag{3.3}$$

  This term is typically weighted and added to the total loss during optimization.

Combined PPO Loss Function:

The overall PPO objective function combines the above components as follows:

$$L^{\text{PPO}}(\theta) = L^{\text{policy}}(\theta) + c_1 L^{\text{value}}(\theta) + c_2 L^{\text{entropy}}(\theta)$$

where c_1 and c_2 are coefficients used to weight the contributions of the value function and entropy losses, respectively.

3.2.8 Comparison of Full Fine-Tuning and PEFT

Fine-tuning tends to perform better in generating higher ROUGE scores than PEFT, yet PEFT possesses humongous strengths in the fields of computer and memory efficiency. PEFT is highly valuable to use with large models on computers that have scarce computers. Despite some minor loss of quality (e.g., 1-2% lower ROUGE scores), PEFT is a pragmatic and scalable solution for effective deployment. Full fine-tuning will provide improved ROUGE score than PEFT, but PEFT has huge memory and compute speed

benefits.

The PEFT process is very convenient to work with large models on cost-effective compute hardware. Although slight small performance degradation (e.g., 1-2% reduction in ROUGE scores) may occur, PEFT offers an efficient and scalable solution for effective deployment.
Overall, end-to-end fine-tuning yields superior ROUGE performance compared to PEFT, but PEFT provides phenomenal benefits of processing and memory. Because of its flexibility, the PEFT model is a good fit for the deployment of large models with restricted processing powers. There may be little performance compromise (say, one to two percent decrease in ROUGE scores), but PEFT provides a flexible and best solution for deployment efficiency.

PEFT model accuracy is compared with base case model based on ROUGE score, computational use, and model size. Fine-tuning gives better ROUGE result primarily but, PEFT gives excellent memory saving and processing efficiency.

In consideration of the elasticity or the flexibility of the PEFT strategy, using large models is the most appropriate whenever computing resources are limited. Even when there are potential opportunities for performance loss (e.g., loss of one to two percent in ROUGE scores), PEFT offers an effective and scalable method towards effective adoption. It is so because PEFT is specifically intended for facilitating effective adoption.

3.3 ROUGE Metric

This specific group of standards, sometimes called the "gold standard," is being applied in a try to measure the reports which are causing. There is comparison drawn between those oversights which were prepared that were utilized as models and the used oversights. The students are completing their way through the "Recall-Oriented Understudy for Gisting Evaluation" test which they are taking currently. The children are currently taking the test. This finally leads to a cautious examination of input and reference totals overlap

between word pairs, word sequences, and n-grams. Owing to the fact that the facts have been already established, this description is being presented to you for reflective consideration. The graphical representation of a mathematical study is given below.

3.3.1 Evaluation Metric ROUGE

ROUGE-n is the program that determines the intersection of n-grams between the reference summary and the produced summary G. It must be an R.

$$\text{ROUGE-n} = \frac{\sum_{S \in R} \sum_{\text{n-gram} \in S} \text{C}_{\text{match}}(\text{n-gram})}{\sum_{S \in R} \sum_{\text{n-gram} \in S} \text{C}(\text{n-gram})} \tag{3.4}$$

Where:

- n: Gives the size of n-grams (for example, ROUGE-1 for unigrams and ROUGE-2 for bigrams).
- C_{match}: Gives the size of n-grams in G that matches n-grams in R.
- C: The total number of n-grams in R.

Precision, Recall, and F1-Score

When examining ROUGE measures, one technique is to examine their accuracy, recall, and F1-scores. Examples of this strategy are:

- **Precision:** Fraction of overlapping n-grams in G that appear in R:

$$\text{Precision} = \frac{|R \cap G|}{|G|} \tag{3.5}$$

- **Recall:** Fraction of overlapping n-grams in R that appear in G:

$$\text{Recall} = \frac{|R \cap G|}{|R|} \tag{3.6}$$

- **F1-Score:**Finding the harmonic mean of recall and precision:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.7}$$

3.3.2 Evaluation Metric for RLHF

Toxicity assessment plays a key role in Reinforcement Learning with Human Feedback (RLHF) in determining and minimizing toxic or offending answers among model outputs. It would ideally utilize a pre-trained detector like a RoBERTa-based hate speech detector to provide a toxicity probability score on answers. Non-toxic score is utilized as a reward signal for training. Success in detoxification is determined via quantitative and qualitative measurement through the use of parameters like mean toxicity score, standard deviation, and side-by-side comparison.

$$\text{Mean Improvement} = \frac{\mu_{\text{before\_detoxification}} - \mu_{\text{after\_detoxification}}}{\mu_{\text{before\_detoxification}}} \tag{3.8}$$

KL divergence[6], however, captures how much a fine-tuned model deviates from its original state. Scaled as penalty in PPO, it blocks the model from straying too much from its baseline action, thus providing fluency and coherence. The reward is penalized by subtracting a KL-scaled penalty, prompting safer output without sacrificing quality. As a pair, toxicity evaluation and KL divergence guarantee safety and stability in training language models.

3.3.3 Final Evaluation

The last test utilizes a sample so much greater than the sample of the previous test. That is to say, it is only after both models have been calibrated. You should keep this in mind as you examine the different ways the models can be utilized in discussions that otherwise have not been discussed.
To find out which fine-tuning technique does better when used on actual conversational data, the results are compared. To facilitate simpler comparison of the results between the studies, initial values are used.

A study of the compute and memory consumed vs. computation rate vs. ROUGE score trade-offs is underway. What the outcomes of all these many fine-tuning methods have been are being examined to determine the merits and demerits of each of them.

The aim of this study is to illustrate that PEFT is well-balanced between speed and

efficiency. Because of this, it would be a great candidate for deployment to be applied in applications utilized by the real world.

Toxicity was reduced in a fine-tuned instruction language model with Reinforcement Learning with Human Feedback (RLHF) by maximizing output by means of a RoBERTa-based hate speech classifier that shifted away from toxic output. Despite the scarcity of toxic samples, the method reduced mean toxicity scores. KL divergence was implemented to synchronize PPO model with reference model without coherence loss and overfitting. Steady KL values 27–29 reflected balance in learning, highlighting the significance of striking a balance between safety and linguistic coherence in RLHF-trained models.

Chapter 4

RESULTS and DISCUSSION

4.1 Result Analysis

4.1.1 Results for Fine Tuning

Flan-T5 was fine-tuned using two distinct methods, and the goal of this research work was to compare and examine the performance of both these methods. It is the work that has been carried out as research and on which this assertion is based. As per the findings, complete fine-tuning and PEFT with LoRA were found to be the best methods. This was determined based on the result of the experiment. Following the introduction of the totally rewritten model with instruction-based prompts, there was an improvement in the rate of production of the summary. Beyond any doubt, this was a great advantage. Of particular interest here is the remarkable progress in each and every one of the ROUGE scores three classes (ROUGE-1, ROUGE-2, ROUGE-L). The PEFT model also kept the performance level equal to the other models even though it was much less efficient than the one that had received full fine-tuning. ROUGE scores fell slightly, but still remained at 1.7%.

Table 4.1: Result Table for ROUGE Metric

Metric	**Original Model**	**Instruct Model**	**PEFT Model**	**PEFT vs Original**	**PEFT vs Instruct**
ROUGE-1	0.2334	0.4216	0.4081	+17.47%	-1.35%
ROUGE-2	0.0760	0.1804	0.1633	+8.73%	-1.70%
ROUGE-L	0.2015	0.3384	0.3251	+12.36%	-1.34%
ROUGE-Lsum	0.2015	0.3384	0.3249	+12.34%	-1.35%

4.2 Observations

- **PEFT vs Original Model:**In every region of ROUGE, PEFT surpasses the baseline model by over 17.47%. Compared to the baseline model, the PEFT model significantly satisfies all ROUGE requirements.

- **PEFT vs Instruct Model:** The PEFT model should be approximately 1.35% worse than the teach model. This is because the teach model is extremely highly finely-tuned. You can see this if you compare the PEFT model and the teach model.

PEFT is the best solution when both capacity for processing and memory are at a limit since it consumes so little of either. This work can be done with very little increase in time in an attempt to advance. PEFT is therefore the best solution in this respect. After conducting extensive research, it was established that PEFT is the best option.

Index	Reward Before	Reward After	Reward Difference
1	2.351077	2.948620	0.597543
2	1.883278	2.461895	0.578617
3	1.587816	1.953090	0.365274
4	1.611620	1.907171	0.295550
5	0.772964	1.032556	0.259592
6	2.434810	2.642623	0.207814
7	1.360727	1.429471	0.068744
8	2.301333	2.369421	0.068088
9	1.165916	1.167107	0.001191
10	3.151282	3.142738	-0.008543
11	1.537484	1.517594	-0.019890
12	1.284117	1.249279	-0.034839
13	1.457126	1.392192	-0.064933
14	2.251136	2.094307	-0.156829
15	2.855694	2.664426	-0.191268
16	2.217947	2.009304	-0.208643
17	1.776725	1.562407	-0.214318
Mean	1.8241	1.9691	0.0908

Table 4.2: Comparison of Reward Before and Reward After for different context with an improvement of -9.84%

4.2.1 Results for RLHF

To assess the effectiveness of Reinforcement Learning with Human Feedback (RLHF) based on Proximal Policy Optimization (PPO), we tested the performance of the model

on 16 varied conversations comparing reward scores prior to and after fine-tuning and KL divergence. The outcomes revealed the consistent increase in reward scores after PPO, confirming that the model delivered more aligned and preferred answers with fewer divergences from the original behavior.But even though there is an improvement of -9.84% over its toxicity. To quantify safety, we employed Facebook's RoBERTa hate speech detector to estimate toxic levels. Fine-tuning with PPO resulted in mean toxicity scores significantly lower with less variance, indicating more uniform and detoxed output. Word comparisons indicated that post-PPO outputs retained the underlying message but with more neutral and respectful language, where even slight wording modifications greatly improved toxicity scores.

Also, reward scores—representative of the probability of non-toxic answers—increased enormously on samples, affirming that PPO assisted the model to follow more closely ethical communication guidelines. To better tune the model, we used Low-Rank Adaptation (LoRA), adapting only 1.4% of parameters. The parameter-efficient method accelerated training and lowered resource use without performance loss or even improving it. Overall, incorporating PPO and PEFT through LoRA successfully enhanced response quality and safety without substantial computation overhead.

Chapter 5

CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

Based on the results of the present research, the process of fine-tuning thoroughly can significantly improve the capacity of the Flan-T5 model to summarize. Based on the results of the study thoroughly, the conclusion was made. It is worth noting here to note that this addition will lead to a general increase in the amount of dollars spent on computers. The other side is observed here. It is a possible way that consumes less but produces equal results that can be obtained. PEFT and LoRA, for instance, have this available for their customers.

The benefit lies in the fact that this is so. This kind of alternative can be done in the future. Considering this specific reaction, the PEFT should be taken into account. This is just one of the numerous reasons that it should be taken into account. Although there is a slight deceleration of rate as the procedure is being performed, the PEFT method is an adaptable remedy. This is especially true in the case of situations and models that require little by way of resources but are of huge size. The reality of the case is that that is exactly what has occurred, albeit with a lesser level of success. In situations where inadequate tools are available, it is especially critical to keep this in mind.

It is most likely that the forthcoming work will examine a number of varied optimisation processses with a goal to improve the performance of PEFT and reduce the gap down by optimizing it up to the end. Further, the PEFT techniques can be demonstrated with a high performance for bulk applications if they are rendered complex or they are

combined with a variety of other memory-efficient methods. Since the processes of PEFT are extremely simple to use, this is achievable. Because of this, the methods of PEFT are far more effective. It can be anticipated that executing PEFT processes in such a way will lead to an enhancement in their performance as a whole. This issue needs to be taken into account. The extraordinary circumstances have made the environment such that there are chances of realizing the objectives that were established. It may be possible that this will be done in trying to make the PEFT procedures more effective; however, doing it would be putting salt to injury. If one makes the research wider so that more data are included and how other components of PEFT, e.g., other ranks, affect it is investigated, one can gain further insight into how the system works and how this can be further extended. One can choose to do this. One can also choose to learn more about the performance of the method. So, it means that there is a chance that you can learn something new because of this. It would be nice to do this in an effort to make big strides toward the goals that were initially set. Enlarging the scale of the outcomes of the research is one method that is present in an attempt to achieve this aim. Maybe, researchers who are investigating the relationship of partial least squares training and reinforcement learning can develop fine-tuning systems that are scalable, efficient, and adaptive. Maybe, these systems can be deployed on large scale. There is a chance that this will occur. This is due to the fact that the model will have to learn automatically something new from new information and concepts which it is not too familiar with. This is going to occur due to the fact that the model is being supplied with new information.

Human Feedback Reinforcement Learning (RLHF) with Proximal Policy Optimization (PPO) and a hate speech reward model to decrease model response toxicity. The PPO-trained model was able to preserve semantic accuracy but provide answers that were significantly less toxic. It preserved fluency and relevance to ensure output quality and coherence during fine-tuning.

Both quantitative measurements and qualitative examination offered definite improvements that attested to the reality that RLHF can make language model action more aligned with human values of safe and respectful communication. Additional training or exposure to more varied samples of toxicity may even achieve greater toxicity

decrease,emphasizing the promise of RLHF in guiding language models towards more human-sensitive and responsible outputs.

5.2 Future Scope

It is quite probable that the subsequent study will investigate a number of different optimization procedures with the intention of improving the performance of PEFT and lowering the gap by fine-tuning it all the way down. Further, the PEFT methods may have a superior performance for large-scale applications if they are made more complex or if they are coupled with other memory-efficient ways. Because PEFT processes are rather easy to understand, this is something that can be accomplished. As a consequence of this, the PEFT techniques are much more efficient. There is a possibility that performing PEFT processes in this manner will result in an improvement in their overall performance. Consideration ought to be given to this matter. The unique conditions have created an environment in which there is a possibility of accomplishing the goals that have been set. It is possible that this might be done in order to enhance the efficiency of the PEFT procedures; nevertheless, doing so would be like adding insult to injury. If the study is expanded to include larger datasets and the implications of more PEFT components, such as different ranks, are studied, it is possible to learn more about how the system operates and how it may be scaled up. This is something that is achievable. Additionally, there is the option of acquiring extra information about the efficiency with which the method operates. Consequently, this suggests that there is a possibility that you may acquire new knowledge as a result of this. It would be advantageous to do so in order to make considerable progress toward the goals that were initially established. Increasing the breadth of the results of the research is one approach that may be used to accomplish this objective.It is possible that researchers who are investigating the connection between partial least squares training and reinforcement learning may be able to construct fine-tuning systems that are very effective, scalable, and adaptive. It is possible that these systems will be implemented on a vast scale. There exists a possibility that this will take place. This is as a result of the fact that the model is required to automatically learn new things from new information and ideas that it is not very acquainted with. This

is going to take place as a consequence of the model being provided with new information.

Future work could focus on merging Reinforcement Learning with Human Feedback (RLHF) using Proximal Policy Optimization (PPO) with advanced Parameter-Efficient Fine-Tuning (PEFT) techniques such as QLoRA. This integration would enable more efficient and scalable training by significantly reducing computational resources and memory requirements while maintaining or improving model performance. Leveraging QLoRA's quantization-aware fine-tuning alongside RLHF could accelerate the training process and allow for fine-grained control over the model's behavior, leading to better alignment with human values and reduced toxicity.

Bibliography

[1] K. Adik, "Peft — parameter efficient fine tuning," *Medium*, 2023, accessed: 2025-05-24. [Online]. Available: https://medium.com/@kanikaadik07/peft-parameter-efficient-fine-tuning-55e32c60c799

[2] Y. Patel, "Introduction to llms and the generative ai: Part 4 — parameter efficient fine-tuning (peft)," *Medium*, 2024. [Online]. Available: https://medium.com/@yash9439/introduction-to-llms-and-the-generative-ai-part-4-parameter-efficient-fine-tuning-peft-83c6dd03978

[3] N. Ding, "Parameter-efficient fine-tuning of large-scale pre-trained language models," *Nature Machine Intelligence*, 2023. [Online]. Available: https://www.nature.com/articles/s42256-023-00626-4

[4] N. Houlsby, "Parameter-efficient transfer learning for nlp," *arXiv preprint*, 2022. [Online]. Available: https://arxiv.org/abs/1902.00751

[5] R. Kirk, I. Mediratta, C. Nalmpantis, J. Luketina, E. Hambro, E. Grefenstette, and R. Raileanu, "Understanding the effects of rlhf on llm generalisation and diversity," *Transactions on Machine Learning Research*, sep 2024. [Online]. Available: https://arxiv.org/abs/2405.07863

[6] T. Kwa, D. Thomas, and A. Garriga-Alonso, "Catastrophic goodhart: regularizing rlhf with kl divergence does not mitigate heavy-tailed reward misspecification," *arXiv preprint*, 2024, presented at Mechanistic Interpretability Workshop, ICML 2024; poster at NeurIPS 2024. [Online]. Available: https://arxiv.org/abs/2407.14503

References

1. C. Raffel *et al.*, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020. [Online]. Available: https://arxiv.org/abs/1910.10683

2. E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models," in *NeurIPS*, 2021. [Online]. Available: https://arxiv.org/abs/2106.09685

3. E. M. Bender, T. Gebru *et al.*, "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?" in *FAccT*, 2021. [Online]. Available: https://dl.acm.org/doi/10.1145/3442188.3445922

4. H. *et al.*, "Adapters: Efficient Fine-Tuning for Natural Language Understanding and Generation," in *ACL*, 2022. [Online]. Available: https://arxiv.org/abs/1902.00751

5. V. Sanh *et al.*, "Multitask Prompted Training Enables Zero-Shot Task Generalization," in *ICLR*, 2021. [Online]. Available: https://arxiv.org/abs/2110.08207

6. Z. Shi and A. Lipani, "DePT: Deep Parameter-Efficient Tuning for Language Models," University College London, United Kingdom, 2024. [Online]. Available: https://arxiv.org/abs/2309.05173

7. P. Lan *et al.*, "Efficient Prompt Tuning by Multi-Space Projection and Prompt Fusion," *Northeastern University, China*, 2024. [Online]. Available: https://arxiv.org/abs/2405.11464

8. L. Wang *et al.*, "Parameter-Efficient Fine-Tuning in Large Models: A Survey of Methodologies," *Zhejiang Laboratory*, 2024. [Online]. Available: https://arxiv.org/abs/2410.19878

9. G. Pu, A. Jain, J. Yin, and R. Kaplan, "Empirical Analysis of the Strengths and Weaknesses of PEFT Techniques for LLMs," *Scale AI*, 2024. [Online]. Available: https://arxiv.org/pdf/2304.14999

10. B. Runwal, T. Pedapati, and P.-Y. Chen, "From PEFT to DEFT: Parameter Efficient Finetuning for Reducing Activation Density in Transformers," *Research Paper*, 2024. [Online]. Available: https://arxiv.org/pdf/2402.01911

11. C. Clarke, Y. Heng, L. Tang, and J. Mars, "PEFT-U: Parameter-efficient fine-tuning for user personalization," *Computer Science & Engineering, University of Michigan, Ann Arbor, MI*, 2024. [Online]. Available: https://arxiv.org/abs/2407.18078

12. R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128-135, 1999. [Online]. Available: https://doi.org/10.1016/S1364-6613(99)01294-2

13. G. M. van de Ven, N. Soures, and D. Kudithipudi, "Continual learning and catastrophic forgetting," *arXiv*, 2024. [Online]. Available: https://arxiv.org/abs/2403.05175

14. Z. Alammar, L. Alzubaidi, J. Zhang, Y. Li, A. Gupta, and Y. Gu, "Generalisable deep learning framework to overcome catastrophic forgetting," *Patterns*, vol. 5, no. 12, p. 100678, 2024. [Online]. Available: https://doi.org/10.1016/j.patter.2024.100678

15. Y. Zhai, S. Tong, X. Li, M. Cai, Q. Qu, Y. J. Lee, and Y. Ma, "Investigating the catastrophic forgetting in multimodal large language model fine-tuning," *Conference on Parsimony and Learning, Proceedings of Machine Learning Research*, vol. 234, 2024. [Online]. Available: https://proceedings.mlr.press/v234/zhai24a.html

16. M. Elsayed and A. R. Mahmood, "Addressing loss of plasticity and catastrophic forgetting in continual learning," *arXiv*, 2024. [Online]. Available: https://arxiv.org/abs/2404.00781

17. A. A. Citarella, M. Barbella, M. G. Ciobanu, F. De Marco, L. Di Biasi, and G. Tortora, "ROUGE metric evaluation for text summarization techniques," *SSRN*

Electronic Journal, 2024. [Online]. Available: https://doi.org/10.2139/ssrn.4753220

18. A. A. Citarella, M. Barbella, M. G. Ciobanu, F. De Marco, L. Di Biasi, and G. Tortora, "ROUGE metric evaluation for text summarization techniques," *SSRN Electronic Journal*, 2024. [Online]. Available: https://doi.org/10.2139/ssrn.4753220

19. A. A. Citarella, M. Barbella, M. G. Ciobanu, F. De Marco, L. Di Biasi, and G. Tortora, "ROUGE metric evaluation for text summarization techniques," *SSRN Electronic Journal*, 2024. [Online]. Available: https://doi.org/10.2139/ssrn.4753220

20. R. Kirk, I. Mediratta, C. Nalmpantis, J. Luketina, E. Hambro, E. Grefenstette, and R. Raileanu, "Understanding the effects of RLHF on LLM generalisation and diversity," *Transactions on Machine Learning Research*, Sep. 2024. [Online]. Available: https://arxiv.org/abs/2405.07863

21. T. Kwa, D. Thomas, and A. Garriga-Alonso, "Catastrophic Goodhart: regularizing RLHF with KL divergence does not mitigate heavy-tailed reward misspecification," presented at the Mechanistic Interpretability Workshop, ICML 2024; poster at NeurIPS 2024. [Online]. Available: https://arxiv.org/abs/2407.14503

22. K. Adik, "PEFT — Parameter Efficient Fine Tuning," *Medium*, 2023. [Online]. Available: https://medium.com/@kanikaadik07/peft-parameter-efficient-fine-tuning-55e32c60c799

23. Y. Rajput, "Introduction to LLMs and the generative AI: Part 4 — Parameter efficient fine-tuning (PEFT)," *Medium*, 2023. [Online]. Available: https://medium.com/@yash9439/introduction-to-llms-and-the-generative-ai-part-4-parameter-efficient-fine-tuning-peft-83c6dd039787