Project Report on

Efficient Fine-tuning of LLM and Reinforcement Learning with Human Feedback

Submitted in partial fulfillment of the requirement for the degree of

B. Tech

in

CSE-AIE 2020-2024

21MAT311: MATHEMATICS FOR INTELLIGENT SYSTEMS 6

&

21AIE311: REINFORCEMENT LEARNING

by

TEAM-16

ABINESH S	**CB.EN.U4AIE20002**
JYOTHIS V. SANTHOSH	**CB.EN.U4AIE20025**
LOGESH KSR	**CB.EN.U4AIE20032**
MANESH KARUN R	**CB.EN.U4AIE20035**
PRANAV UNNIKRISHNAN	**CB.EN.U4AIE20053**

AMRITA
VISHWA VIDYAPEETHAM

Center for Computational Engineering and Networking (CEN)

Amrita Vishwa Vidyapeetham
Ettimadai, Coimbatore

2023

ABSTRACT

This project presents a comprehensive investigation into the application of parameter-efficient fine-tuning and reinforcement learning techniques for enhanced model adaptation. The study focuses on diverse aspects, including different floating point data points, 8-bit matrix multiplication, and low-rank adaptation, with the primary aim of fine-tuning the Llama model 7B on the Vicuna dataset, consisting of 20,000 samples. Two key techniques, namely Quantized Low-Rank Adaptation (QLoRA) and Parameter-Efficient Fine-Tuning (PEFT), are employed to maximize efficiency while ensuring high accuracy. Additionally, the paper explores the integration of Reinforcement Learning from Human Feedback (RLHF) and utilizes the TRL library (Transformer Reinforcement Learning) to facilitate the research. The results offer insights into the effectiveness of QLoRA, PEFT, and RLHF in enhancing model performance and adaptability.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

In today's fast-paced technological landscape, machine learning models play a pivotal role in various domains. To maximize their performance and adaptability, researchers continually strive to improve their efficiency and fine-tuning capabilities. In this project report, we delve into the realm of model adaptation, focusing on different floating point data points, 8-bit matrix multiplication, low-rank adaptation, and parameter-efficient fine-tuning. The primary objectives of this project encompass the utilization and evaluation of two key techniques: Quantized Low-Rank Adaptation (QLoRA) and Parameter-Efficient Fine-Tuning (PEFT). We aim to employ these techniques to fine-tune the Llama model 7B on the Vicuna dataset, consisting of 20,000 samples. By employing QLoRA and PEFT, we seek to enhance the model's adaptability, performance, and efficiency while maintaining a high level of accuracy. QLoRA, a novel approach introduced in recent research, allows for efficient model adaptation by leveraging quantization and low-rank approximation techniques. By quantizing the model's parameters and employing low-rank matrices, QLoRA significantly reduces the computational complexity while preserving the model's expressive power.nPEFT, on the other hand, focuses on parameter-efficient fine-tuning. Traditional fine-tuning methods often require extensive computational resources and time-consuming iterations. PEFT addresses this challenge by adopting strategies that maximize the utilization of existing knowledge while minimizing the number of model updates.

Additionally, our project explores the field of reinforcement learning, specifically Reinforcement Learning from Human Feedback (RLHF). RLHF is a paradigm that combines the power of reinforcement learning algorithms with human expertise. By leveraging the insights and feedback from human experts, RLHF aims to accelerate the learning process and improve the model's performance. To facilitate our research, we employ the TRL library (Transformer Reinforcement Learning), a comprehensive framework that provides a suite of tools and algorithms for reinforcement learning tasks with transformer-based models. The TRL library offers an array of functionalities to support our exploration of RLHF and reinforce our investigation into QLoRA and PEFT. Through our project, we aim to contribute to the growing body of knowledge on efficient model adaptation and reinforcement learning techniques. By fine-tuning the Llama model 7B on the Vicuna dataset using QLoRA and conducting extensive research on reinforcement learning, we anticipate gaining valuable insights into improving the performance and adaptability of machine learning models.

CHAPTER 2
PROJECT CHAPTERS

2.1 PEFT (PARAMETER-EFFICIENT FINE-TUNING)

PEFT, a library developed by Hugging Face, is designed to facilitate the creation and fine-tuning of adapter layers on large language models (LLMs). It seamlessly integrates with the Hugging Face Accelerate ecosystem, enabling efficient training and inference on large-scale models using frameworks like DeepSpeed and Big Model Inference.

The PEFT library provides support for a wide range of state-of-the-art models and offers numerous examples showcasing its capabilities. Some of the tasks and applications it covers include causal language modeling, conditional generation, image classification, 8-bit int8 training, the low-rank adaption of Dreambooth models, semantic segmentation, sequence classification, and token classification.

With its extensive model support and comprehensive set of examples, PEFT empowers researchers and practitioners to leverage adapter layers for efficient fine-tuning of LLMs across various domains and tasks.

2.2 8-BIT MATRIX MULTIPLICATION

Efficient 8-bit matrix multiplication is a technique introduced in the paper LLM.int8() to address performance degradation when quantizing large-scale models. This method focuses on optimizing matrix multiplications in Linear layers, which are commonly used in neural networks.

The approach involves breaking down the matrix multiplications into two stages: the outlier hidden states part and the ”non-outlier” part. The outlier hidden states are processed using float16 precision, while the non-outlier part is computed using int8 precision.

By employing 8-bit matrix multiplication, the size of a full-precision model can be reduced by a factor of 4. Similarly, for half-precision models, the reduction is by a factor of 2. This reduction in model size can lead to improved efficiency and resource utilization in memory-constrained environments.

2.3 LOW-RANK ADAPTATION AND PEFT

In 2021, a research paper titled ”LoRA: Low-Rank Adaption of Large Language Models” introduced a method for fine-tuning large language models by freezing the pre-trained weights and utilizing low-rank versions of the attention matrices in the query and value layers. This approach significantly reduces the number of parameters in the model, resulting in lower memory requirements during fine-tuning.

The authors of the paper demonstrated that fine-tuning with low-rank adapters achieved comparable results to fine-tuning the entire pre-trained model. This technique enables fine-tuning of language models while using significantly less GPU memory.

However, there are trade-offs to consider. The forward and backward passes become slower by approximately a factor of two due to the additional matrix multiplications involved in the adapter layers. Despite this drawback, the memory efficiency gained through low-rank adaption makes it a valuable approach for fine-tuning large language models with limited computational resources.

2.4 QLORA (QUANTIZED LOW-RANK ADAPTATION)

QLoRA is a technique that reduces the memory requirements for fine-tuning language models without sacrificing performance compared to standard 16-bit model fine-tuning. It achieves this by compressing a pre-trained language model using 4-bit quantization and adding a small number of trainable parameters known as Low-Rank Adapters. During fine-tuning, only the Low-Rank Adapters are updated, while the frozen quantized pre-trained model is used to propagate gradients. QLoRA utilizes a storage data type for the base model weights and a computation data type for performing computations. It dequantizes the weights when necessary, keeping memory usage low during training and inference.

Multiple experiments have demonstrated that QLoRA fine-tuning matches the performance of 16-bit methods. The Guanaco models, which employ QLoRA fine-tuning for LLaMA models on the OpenAssistant dataset, are state-of-the-art chatbot systems and perform competitively with ChatGPT on the Vicuna benchmark. This highlights the effectiveness of QLoRA as a tuning technique.

(i) WORKING

1. Load a model using 4-bit precision. There are different variants of 4-bit quantization that you can experiment with, such as NF4 (normalized float 4) or pure FP4 quantization. The recommended option, based on both theoretical considerations

and empirical results from the paper, is to use NF4 quantization, as it tends to offer better performance.

2. Flexibility to choose any combination, such as float16, bfloat16, or float32. Using a 16-bit compute data type can lead to faster matrix multiplication and training.

3. It is possible to modify the compute data type of the quantized model. This means that we can choose the desired data type for performing computations within the quantized model.

4. We can enable nested quantization, this feature allows for a second round of quantization after the initial one, resulting in an additional saving of 0.4 bits per parameter. It is worth noting that this feature is also utilized in the training Google Colab notebook, highlighting its practical application and usefulness.

5. It's important to note that all the components discussed are composable, meaning we can combine them to find the optimal configuration for our specific use case. Here are some general guidelines:

   (a) If memory usage is a concern, you can consider using double quantization to save additional memory.

   (b) For higher precision, NF4 quantization is recommended.

   (c) If faster fine-tuning is desired, using a 16-bit compute data type can be beneficial.

(ii) **What are the supported models?**

This includes popular architectures such as Llama, OPT, GPT-Neo, GPT-NeoX for text models, Blip2 for multimodal models, and more. These models can be quantized using the 4-bit approach with the support of the accelerated library.

2.4.1 Can we train 4bit/8bit models?

Pure 4-bit training is not possible for these models. However, it is feasible to train them using parameter-efficient fine-tuning methods (PEFT) by incorporating adapters on top of the models. This approach, which we implemented in this paper, is officially supported by the PEFT library from Hugging Face.

2.5 RLHF (REINFORCEMENT LEARNING FROM HUMAN FEEDBACK

It is a complex concept that involves multiple stages of training and deployment. In this paper, we will outline the training process into three fundamental steps:

1. **Pretraining a language model (LM):** The initial step is to train a language model using a large corpus of text data. This pre-trained LM forms the foundation for subsequent steps.

2. **Gathering data and training a reward model:** Data is collected by obtaining human preferences or feedback on different model-generated outputs. This data is then used to train a reward model, which learns to assign scores or rewards to model-generated samples based on their quality or desirability.

3. **Fine-tuning the LM with reinforcement learning:** The pre-trained LM is fine-tuned using reinforcement learning techniques. The reward model guides the training process by providing feedback signals, and the LM is updated iteratively to optimize its outputs based on the learned rewards.

(i) **Training process**

Pretraining language models

To begin with, RL from Human Feedback (RLHF) typically utilizes a language model that has undergone pretraining using classical pretraining objectives. The choice of the initial language model can vary. For instance, OpenAI employed a smaller version of GPT-3 called InstructGPT for their popular RLHF model. Anthropic used transformer models ranging from 10 million to 52 billion parameters, and DeepMind employed their 280 billion parameter model named Gopher.

While the initial model can be further fine-tuned on additional text or conditions, it is not a mandatory step. For example, OpenAI fine-tuned their model using human-generated text deemed ”preferable,” while Anthropic distilled an original language model based on context clues to meet their criteria of being ”helpful, honest, and harmless.” These approaches involve using expensive and augmented data sources, but they are not obligatory for understanding RLHF.

Determining the best starting model for RLHF does not have a definitive answer. The choice of the initial model is an open-ended exploration in the design space of RLHF training.

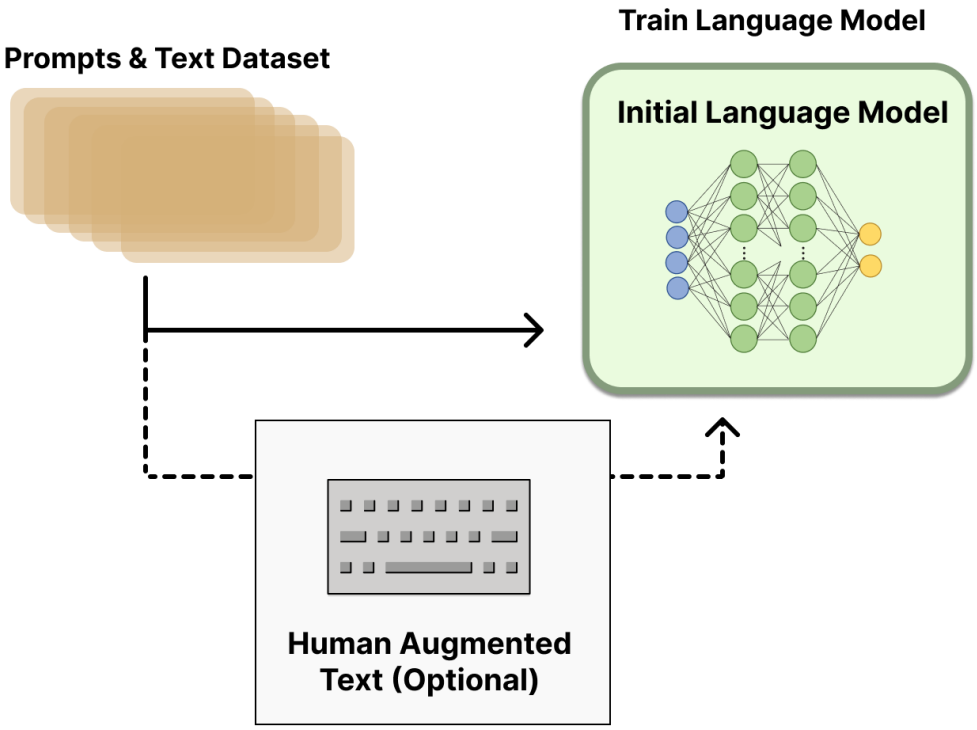

Figure 2.1: Pre-training

The next step involves generating data to train a reward model, which integrates human preferences into the RLHF system.

Reward model training The process of generating a reward model (RM) or preference model calibrated with human preferences is a crucial aspect of RL from Human Feedback (RLHF). The objective is to develop a system that takes a sequence of text as input and returns a scalar reward, representing human preference. This system can be an end-to-end language model (LM) or a modular system where the output is a reward, such as ranking the outputs and converting the ranking into a reward value. The scalar reward output is important for seamless integration with existing RL algorithms in the subsequent stages of RLHF.

LMs for reward modeling can either be another fine-tuned LM or an LM trained from scratch on preference data. For example, Anthropic utilizes a specialized fine-tuning method called preference model pretraining (PMP) after pretraining, as it has been found to be more sample-efficient than regular fine-tuning. However, there is currently no consensus on the best variation of reward modeling, and research in this area is ongoing.

The training dataset for the RM consists of prompt-generation pairs, which are gen-

erated by sampling prompts from a predefined dataset. Anthropic's data, is primarily generated using a chat tool on Amazon Mechanical Turk. OpenAI, on the other hand, used prompts submitted by Users to the GPT API. These prompts are passed through the initial LM to generate new text.

Human annotators are employed to rank the generated text outputs from the LM. Instead of assigning scalar scores directly to each piece of text, which can be challenging due to varying human values and resulting in uncalibrated and noisy scores, rankings are used to compare the outputs of multiple models and create a more reliable and regularized dataset.

There are multiple methods for ranking the text, with one successful approach being to have users compare the generated text from two language models conditioned on the same prompt. By comparing model outputs in head-to-head matchups, an Elo system can be employed to generate a ranking of the models and their outputs relative to each other. These different ranking methods are then normalized into a scalar reward signal for training.

Interestingly, successful RLHF systems to date have utilized reward language models of varying sizes relative to text generation. For example, OpenAI employed a 175B LM with a 6B reward model, Anthropic used LM and reward models ranging from 10B to 52B, and DeepMind employed 70B Chinchilla models for both the LM and reward model. It is intuitive that preference models need to have a similar capacity to comprehend the text as a model generating the text would require.

At this stage of the RLHF system, there is an initial language model capable of generating text and a preference model that assigns a score to any given text based on human perception. The next step involves using reinforcement learning (RL) to optimize the original language model with respect to the reward model.

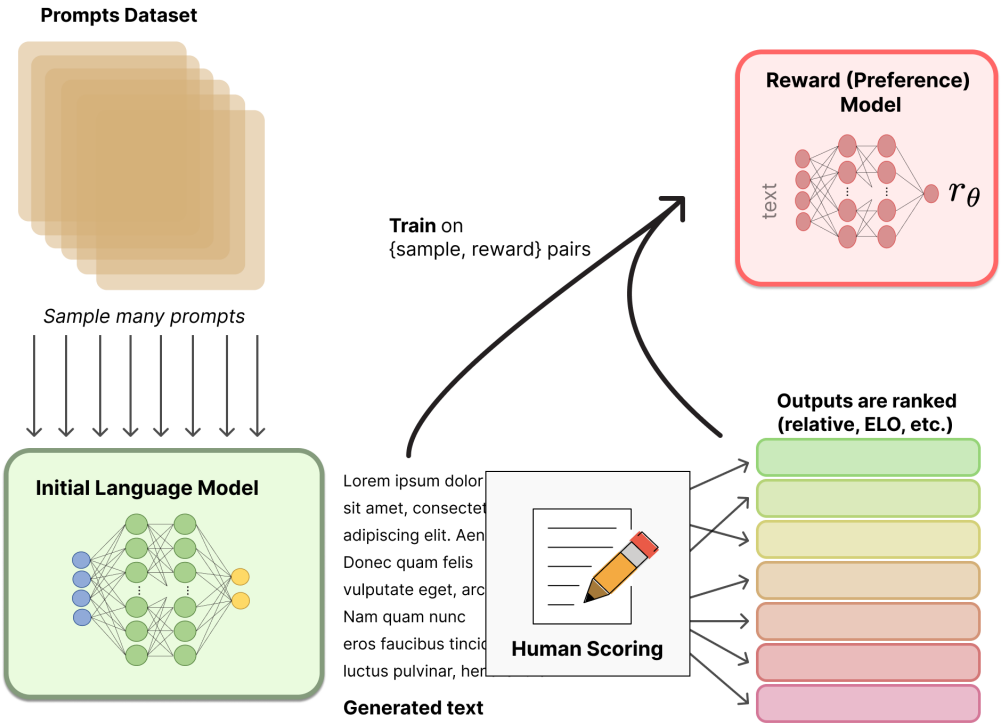

Figure 2.2: Reward model

Fine-tuning with RL For a long time, training a language model (LM) with reinforcement learning (RL) was considered impractical due to engineering and algorithmic challenges. However, multiple organizations have achieved success by fine-tuning some or all of the parameters of a copy of the initial LM using the Proximal Policy Optimization (PPO) algorithm, a policy-gradient RL algorithm. Fine-tuning the entire LM with billions or even trillions of parameters is prohibitively expensive, so parameters are frozen, and only a subset is updated during RL training. PPO, a well-established RL algorithm, was a favorable choice for scaling RL training for RL from Human Feedback (RLHF) due to its relative maturity.

To formulate fine-tuning as an RL problem, the policy is the LM that takes a prompt as input and generates a sequence of text or probability distributions over text. The action space consists of all tokens in the LM's vocabulary, and the observation space is the distribution of possible input token sequences. The reward function combines the preference model and a constraint on policy shift. Given a prompt, the fine-tuned policy generates text, which is concatenated with the original prompt. The preference model assigns a scalar reward, representing "preferability," to the combined text-prompt input. Additionally, per-token probability distributions from the RL policy are compared to those from the initial model to compute a penalty on the difference between them. This

penalty, often based on the Kullback-Leibler (KL) divergence, prevents the RL policy from deviating significantly from the initial pre-trained model. It helps ensure that the generated text remains coherent and prevents the optimization process from generating nonsensical text that still receives a high reward from the preference model. The KL divergence is approximated via sampling from both distributions, and the final reward sent to the RL update rule is a combination of the reward term and the KL penalty term.

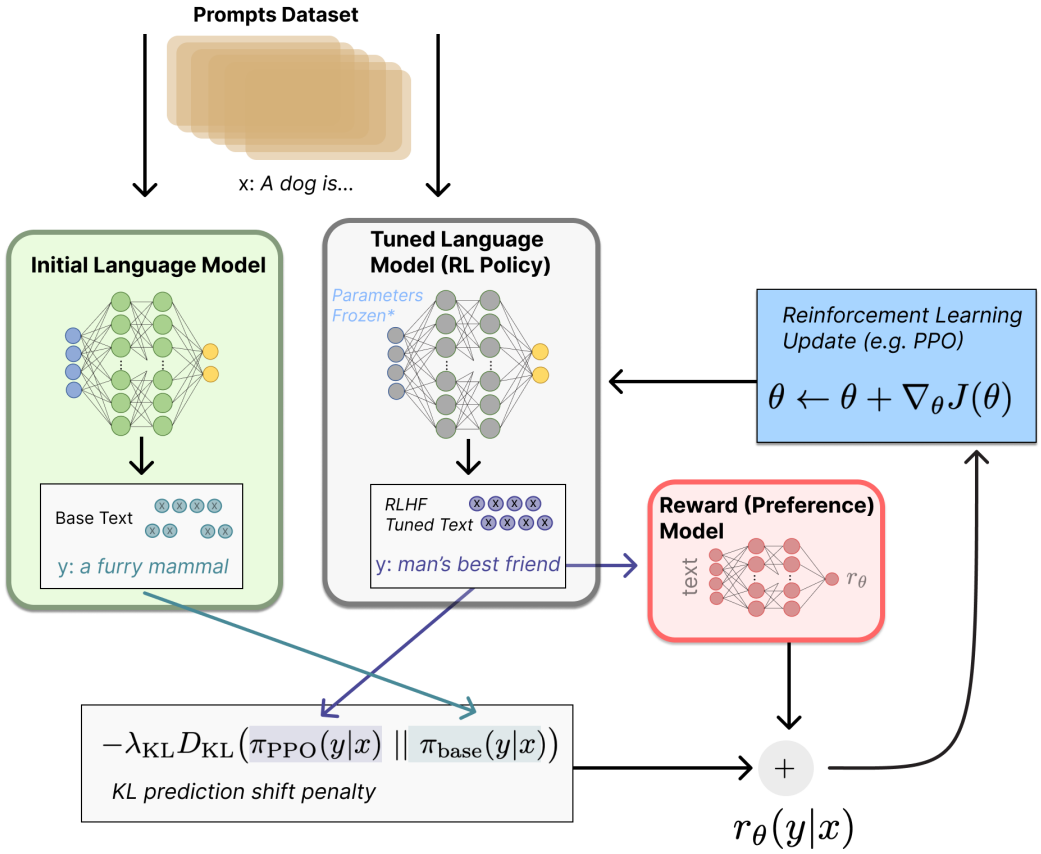

Figure 2.3: RLHF

Some RLHF systems may incorporate additional terms into the reward function. For instance, OpenAI successfully experimented with mixing additional pre-training gradients, derived from human annotation sets, into the update rule for PPO in their InstructGPT model. As research in RLHF progresses, the formulation of the reward function is expected to continue evolving.

Finally, the update rule involves applying the PPO algorithm to maximize the reward metrics using the current batch of data. PPO is an on-policy optimization algorithm that ensures the parameter updates are constrained within a trust region to avoid destabilizing the learning process.

2.6 TRL LIBRARY (TRANSFORMER REINFORCEMENT LEARNING

The trl library simplifies and enhances the RL step of fine-tuning language models, allowing users to apply RL on their custom datasets and training setups. It offers flexibility and ease of use for a wide range of applications, such as fine-tuning models for generating positive movie reviews, controlled generation, or reducing toxicity.

With trl, we can utilize the popular Deep RL algorithm, Proximal Policy Optimization (PPO), either in a distributed manner or on a single device. The library leverages accelerate a component of the Hugging Face ecosystem, to enable scaling up experiments to a significant scale, making it accessible to any user.

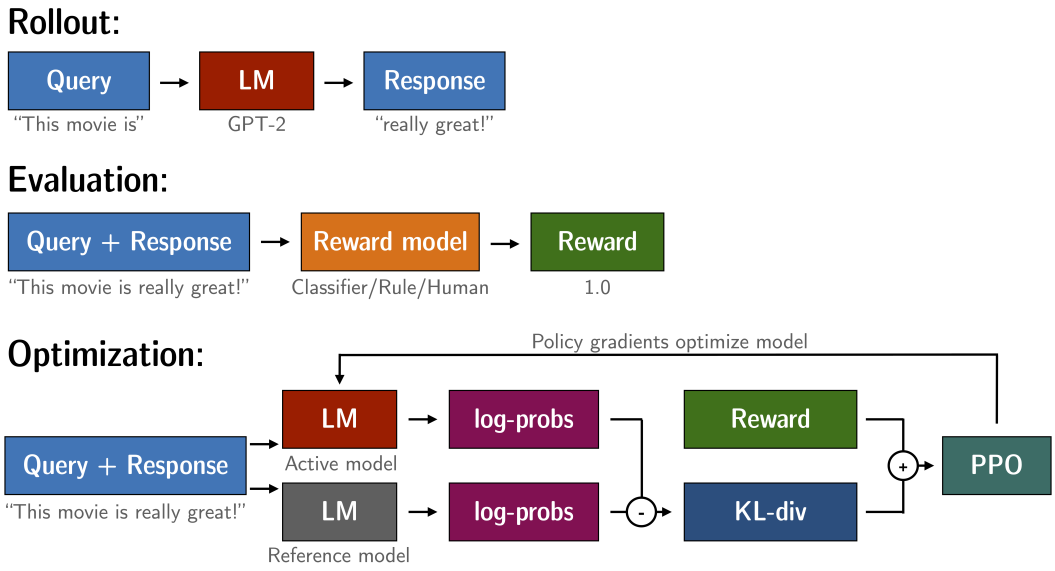

Figure 2.4: TRL overview

The process of fine-tuning a language model with RL using trl follows a general protocol. It involves working with two copies of the original model. To ensure that the active model doesn't deviate too much from its original behavior or distribution, the logits (outputs) of the reference model are computed at each optimization step. This constraint adds complexity to the optimization process, as it requires maintaining at least two copies of the model per GPU device. As the model size increases, fitting the setup onto a single GPU becomes more challenging.

(i) WORKING

Training language models at scale can present various challenges. One significant challenge is ensuring that the model and its optimizer states can fit into the available GPU

memory. The memory requirements depend on the precision or data type of the model parameters, such as float32 (32-bit), float16 (16-bit), bfloat16 (16-bit), and even more exotic precisions like int8 (8-bit). As a rule of thumb, loading a model with one billion parameters typically consumes 4GB of memory in float32 precision, 2GB in float16 precision, and 1GB in int8 precision.

To address these challenges at scale, several techniques have been adopted. The most common paradigms are Pipeline Parallelism, Tensor Parallelism, and Data Parallelism.

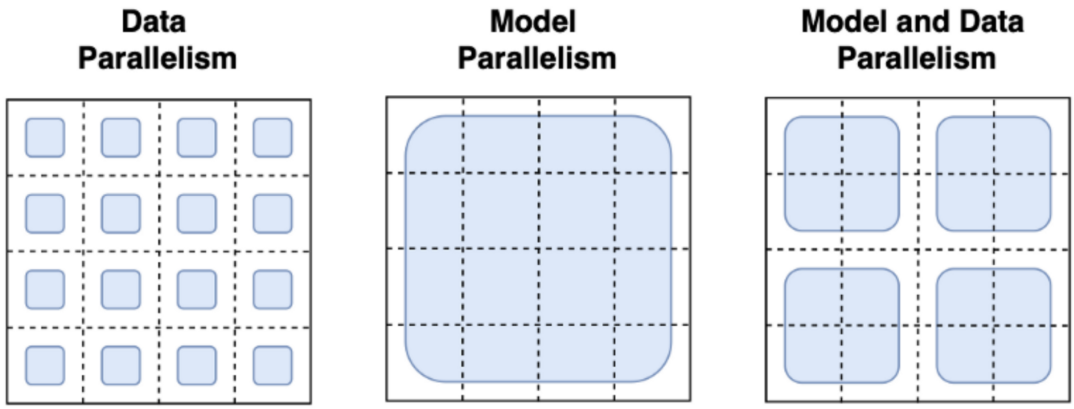

Figure 2.5: Model Parallelism

Data Parallelism involves hosting multiple instances of the same model in parallel on different machines, with each instance being fed a different data batch. This strategy replicates the single-GPU case and is already supported by trl.

In Pipeline Parallelism, the model is distributed across machines by splitting it layer-wise. This approach requires defining a communication protocol for exchanging activations and gradients across processes. Implementing this strategy is non-trivial and may involve frameworks like Megatron-DeepSpeed or Nemo.

Tensor Parallelism, on the other hand, splits tensor operations across GPUs, such as matrix multiplications. It also requires sharding the model weights across multiple devices and defining communication protocols for activations and gradients.

Implementing Model Parallelism strategies like Pipeline and Tensor Parallelism often involves adopting specialized frameworks and tools such as Megatron-DeepSpeed or Nemo. These frameworks facilitate the distribution and communication of model components across devices.

Additionally, there are other essential tools for scaling language model training, including Adaptive Activation Checkpointing and fused kernels, which help optimize

memory usage and computational efficiency.

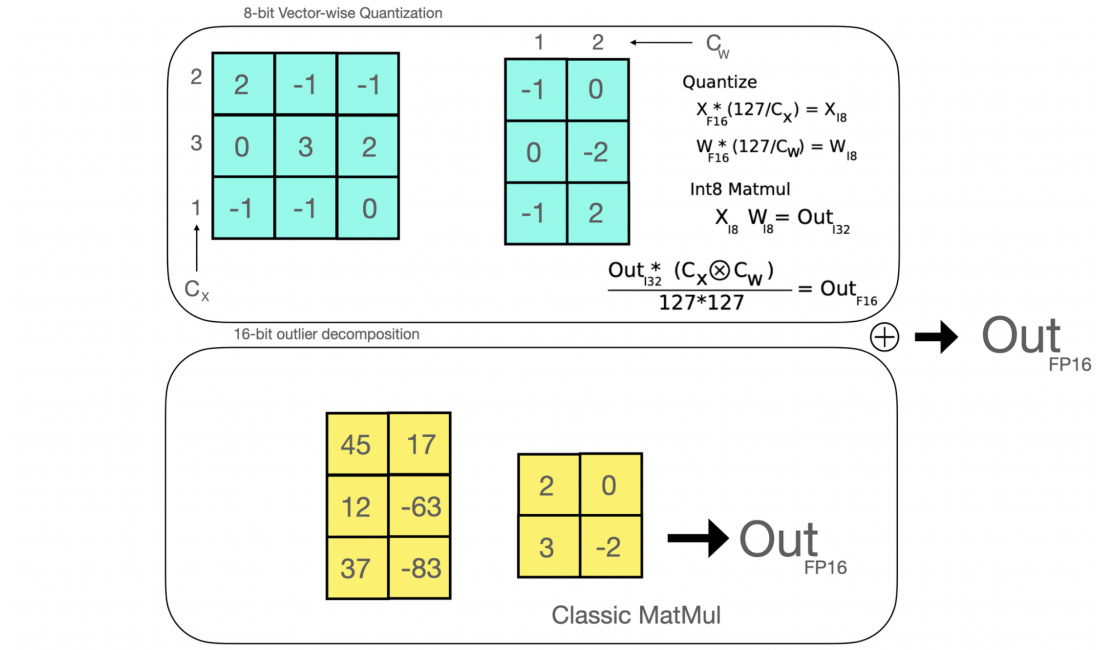

Figure 2.6: 8bit matrix multiplication

CHAPTER 3
RESULTS

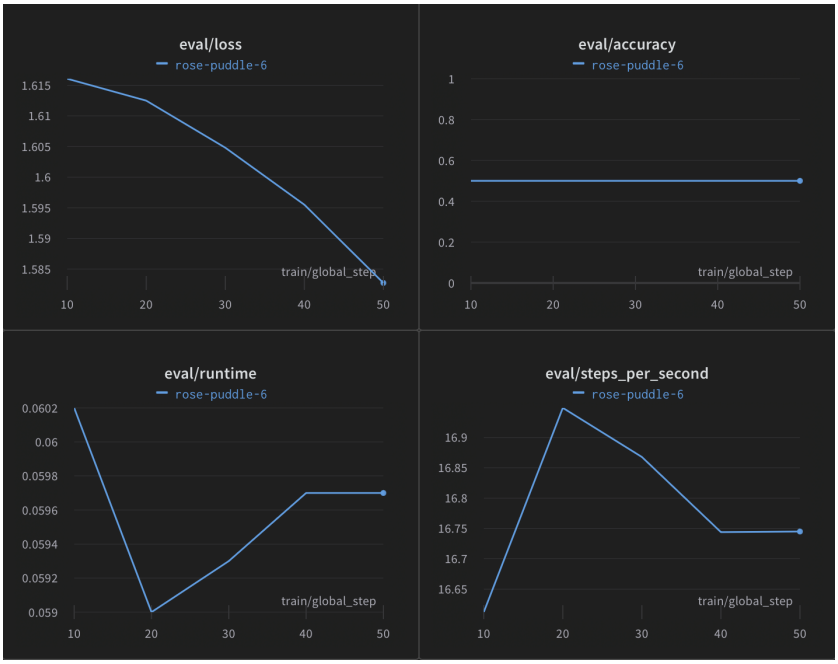

Figure 3.1: Training Results

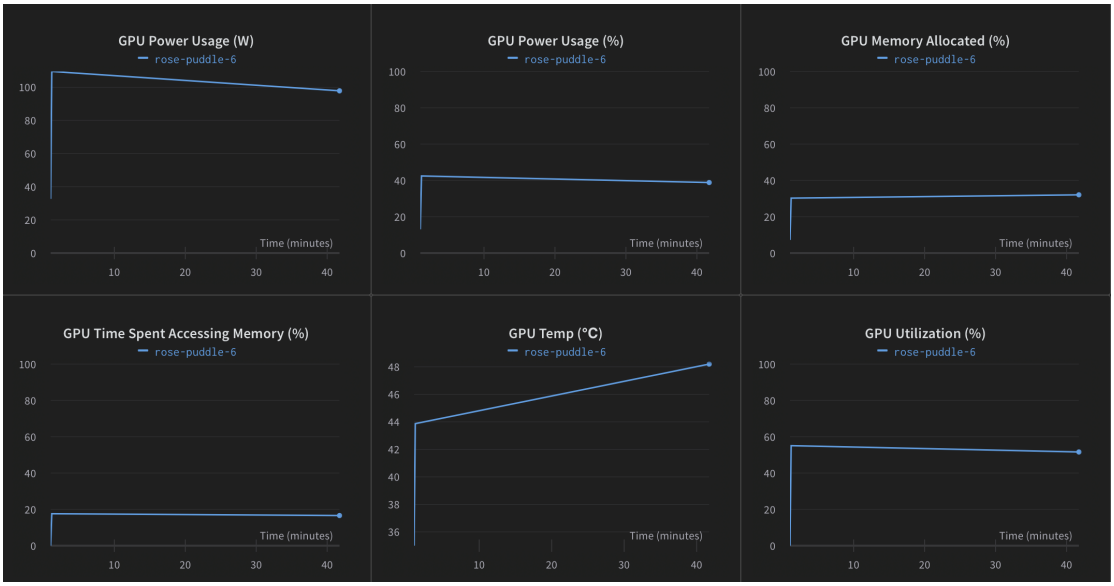

Figure 3.2: System Results

CHAPTER 4
CONCLUSION

This project has focused on enhancing the performance of the Llama Uncensored model through the application of Qlora and the generation of a policy. Despite limitations regarding training time and GPU constraints, we successfully implemented the policy on the GPT-2 model, which yielded promising results in terms of generating text that exhibited both creativity and informativeness. Furthermore, we presented the models with a user interface, providing a tangible demonstration of their capabilities.

Looking ahead, there are several avenues for future exploration and improvement. Foremost, we intend to train the 7B model on the RL policy, as we anticipate this will yield further enhancements in performance. This step has the potential to unlock even greater levels of creativity and informativeness in the generated text.

This project has achieved noteworthy outcomes. We have successfully finetuned the Llama Uncensored model using Qlora, generated a policy for it, and demonstrated its capabilities through a user interface. Moving forward, we plan to refine the models by training the 7B variant on the RL policy. We firmly believe that these models hold immense potential for diverse applications, including but not limited to generating creative content, answering questions, and facilitating language translation. By further refining and expanding upon these models, we can unlock their true potential and contribute to the advancement of natural language processing .

REFERENCES

1. QLoRA: Efficient Finetuning of Quantized LLMs.
   Link: https://arxiv.org/abs/2305.14314

2. Fine-tuning a masked language model.
   Link: https://huggingface.co/course/chapter7/3?fw=pt

3. "Learning from human preferences" by OpenAI (2018).

4. "Reinforcement learning from human feedback: A survey" by Liu et al. (2022).

5. "Incorporating human feedback into reinforcement learning" by Veness et al. (2017).

6. "Reinforcement learning with human feedback: Learning dynamic choices via pessimism" by Hu et al. (2023).

7. "Illustrating Reinforcement Learning from Human Feedback (RLHF)" by Hugging Face (2023).

8. LLaMA: Open and Efficient Foundation Language Models.
   Link: https://arxiv.org/abs/2302.13971

9. A Survey of Large Language Models.
   Link: https://arxiv.org/abs/2303.18223

10. A Bibliometric Review of Large Language Models Research from 2017 to 2023.
    Link: https://arxiv.org/abs/2304.02020

11. Introducing LLaMA: A foundational, 65-billion-parameter large language model.
    Link: https://ai.facebook.com/blog/large-language-model-llama-meta

12. Vicuna: A Large-Scale Dataset of Instructions for Fine-Tuning LLMs.
    Link: https://arxiv.org/abs/2203.13647

13. Fine-tuning LLMs for Question Answering with the Vicuna Dataset.
    Link: https://arxiv.org/abs/2203.14400

14. Fine-tuning LLMs for Natural Language Inference with the Vicuna Dataset.
    Link: https://arxiv.org/abs/2203.14401