Improving the Language Understanding Capabilities of Large Language Models Using Reinforcement Learning

Sai Ashish Somayajula[*] **Bokai Hu**[*] **Qi Cao** **Xin Pan** **Pengtao Xie**
UC San Diego
{ssomayaj, boh001, q9cao, p1xie}@ucsd.edu

Abstract

Instruction-fine-tuned large language models (LLMs) under 14B parameters continue to underperform on natural language understanding (NLU) tasks, often trailing smaller models like BERT-base on benchmarks such as GLUE and SuperGLUE. Motivated by the success of reinforcement learning in reasoning tasks (e.g., DeepSeek), we explore Proximal Policy Optimization (PPO) as a framework to improve the NLU capabilities of LLMs. We frame NLU as a reinforcement learning environment, treating token generation as a sequence of actions and optimizing for reward signals based on alignment with ground-truth labels. PPO consistently outperforms supervised fine-tuning, yielding an average improvement of 6.3 points on GLUE, and surpasses zero-shot and few-shot prompting by 38.7 and 26.1 points, respectively. Notably, PPO-tuned models outperform GPT-4o by over 4% on average across sentiment and natural language inference tasks, including gains of 7.3% on the Mental Health dataset and 10.9% on SIGA-nli. This work highlights a promising direction for adapting LLMs to new tasks by reframing them as reinforcement learning problems, enabling learning through simple end-task rewards rather than extensive data curation. Our code is available at https://github.com/coder-qicao/RL4GLUE.

1 Introduction

Large language models (LLMs) (Radford et al., 2019; Brown, 2020; Touvron et al., 2023b) have revolutionized natural language processing (NLP) with their powerful text generation capabilities (Radford, 2018). Pretrained on large-scale unlabeled corpora, LLMs can generate coherent and contextually relevant content. Using prompt-based strategies like zero-shot and few-shot prompting (Brown, 2020), these models can address a wide range of downstream tasks without task-specific fine-tuning. However, when applied to instruction-fine-tuned LLMs under 14B parameters [1]—such as LLAMA2-7B-chat-hf—these methods often underperform on natural language understanding (NLU) tasks compared to encoder-only models like BERT (Devlin, 2018), which consistently excel on benchmarks such as GLUE (Wang et al., 2019) and SuperGLUE (Wang et al., 2020). For instance, our evaluation of LLAMA2-7B-chat-hf shows that zero-shot prompting with task-specific prompts yields an average performance of 46.1 across all GLUE datasets, while few-shot prompting improves performance to 58.7—both significantly trailing BERT-base's 79.6, as shown in Table 1.

To enhance NLU capabilities of LLMs, we investigate reinforcement learning (RL)-based fine-tuning approaches. Motivated by recent work such as DeepSeek (Liu et al., 2024), which demonstrates the utility of reward-driven optimization for improving reasoning abilities, we explore the use of Proximal Policy Optimization (PPO) (Schulman et al., 2017a) to align model outputs with task-specific objectives.

While standard fine-tuning (SFT) is commonly used to adapt LLMs to downstream tasks, we find it insufficient for NLU—often underperforming even smaller encoder-only models like BERT-base. In contrast, we use PPO to enhance LLM performance by framing NLU as a reinforcement learning problem, as shown in Fig 1. The sequence of input tokens up to timestep $t-1$ represents the state s_t, and the token generated at timestep t is treated as the action a_t. After generating the full response, a heuristic extracts the predicted answer, which is compared to the ground-truth label to assign a scalar reward R. PPO then updates the model to maximize

[*]Equal Contribution

[1]The 14B cutoff reflects our computational constraints; LLAMA-2-13B-chat-hf is the largest model we were able to evaluate.

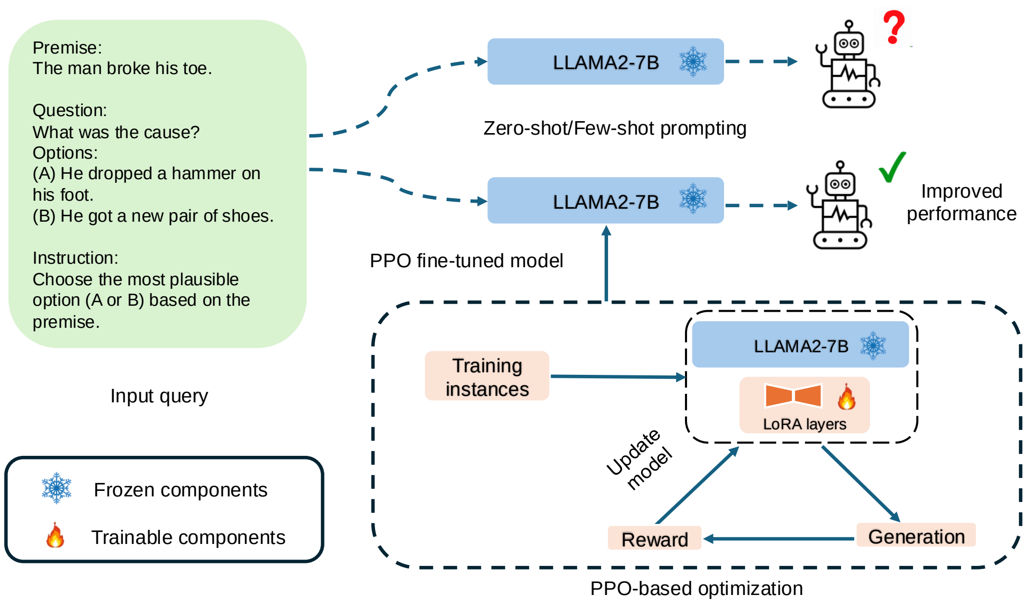

Figure 1: Despite extensive pre-training and instruction tuning, zero-shot and few-shot prompting of models like LLAMA2-7B-chat-hf and LLAMA2-13B-chat-hf continues to underperform on NLU tasks, often falling short of smaller encoder-based models like BERT. To address this performance gap, we recast the task of NLU adaptation as an RL problem, fine-tuning the model using PPO. The input consists of a prompt and a query. In the standard setting (top path), the base model, with zero-shot/few-shot prompting, struggles to generate correct answers. Our approach selectively updates lightweight LoRA layers, using PPO to optimize a reward signal based on task correctness. This optimization encourages performance gains while constraining deviation from the base policy. We find that PPO fine-tuning substantially outperforms standard SFT.

this reward, enabling direct optimization for task-specific objectives. Empirically, PPO-based fine-tuning of LLAMA2-7B-chat-hf improves GLUE performance by 6.3 points over SFT, surpasses zero and few-shot prompting by 38.7 and 26.1 points, respectively, and even outperforms GPT-4o by over 4% on sentiment and inference tasks—achieving gains of 7.3% on the Mental Health dataset and 10.8% on SIGA-nli. These results demonstrate the effectiveness of reinforcement learning in aligning LLMs under 14B parameters with NLU objectives.

Pre-trained LLMs possess broad linguistic knowledge, spanning syntactic and semantic structures, acquired from large-scale text corpora. We show that reinforcement learning, specifically PPO, can refine this general understanding to better align with task-specific NLU objectives. Similar patterns are observed in DeepSeek, where chain-of-thought pre-training enhances reasoning capabilities, and subsequent RL-based fine-tuning further improves performance. Building on this insight, our findings suggest a promising direction: adapting LLMs to new tasks by formulating them as reinforcement learning problems. When models are sufficiently pre-trained, task alignment may be achieved without additional labeled data—requiring only a well-defined reward function over the outputs. PPO can then optimize the model toward high-reward behaviors. This approach offers a scalable, label-efficient alternative to conventional supervised fine-tuning through reward-driven adaptation.

2 Method

To enhance the performance of LLMs on NLU tasks, we adopt two distinct fine-tuning methods. The first approach involves supervised fine-tuning, where the input consists of a concatenation of the task-specific prompt, query and the ground truth answer, with the model optimized using the next-token prediction objective. The second approach utilizes PPO, framing response generation as a reinforcement learning problem. In this setup, the sequence of input tokens until timestep $t-1$ represents the state s_t, and each token generated at timestep t is treated as an action a_t. After generating the entire sequence, a heuristic-based process extracts the final answer from this generated sequence, and is compared to the ground truth. PPO is then employed to optimize the model by maximizing the cumulative reward derived from this comparison. To reduce computational complexity, we fine-tune LoRA layers instead of the full model. We refer the readers to Appendix A and B for preliminaries on PPO.

2.1 Task-Specific Prompt Design

We detail the construction of task-specific prompts used to query the LLM for NLU tasks. Each prompt begins with a clear task description, outlining the necessary background information to guide the model in solving the task. Following this, we specify strict requirements for the output format, ensuring that the response is encapsulated within a predefined structure, specifically between '<Judgement></Judgement>' tags. This structure ensures consistency in the model's responses, facilitating easier extraction and evaluation of the results.

For example, in the CoLA task, which assesses grammatical acceptability, the prompt is structured as follows:

```
System_prompt:
    You are an assistant to analyze the
    linguistic properties
    of a sentence. The task is to decide
     the linguistic acceptability
    of a sentence. If the sentence is
    linguistically correct then it
    is acceptable, else it is not.
The result you give should have the
    following form:
    <Judgement> {Insert only "Yes" or "
    No" here} </Judgement>
Prompt:
    Now judge if the sentence "{sentence
    }" is linguistically acceptable.
Assistant:
    <Judgement>
```

The prompt starts with background information about CoLA, specifies restrictions on the output (such as labeling a sentence as acceptable or unacceptable), and concludes with a special start token, <Judgement>, to initiate the model's response generation.

2.2 Supervised Fine-tuning of LLM on NLU Tasks

Given an NLU training dataset, $\mathcal{D}^{(tr)} = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i represents the input text and y_i the ground truth label, we fine-tune the LLM on a sequence consisting of the task-specific prompt p (described in section 2.1) concatenated with the input x_i and the ground truth answer y_i. The model is trained using the next-token prediction objective, where it predicts the next token in the sequence by conditioning on all preceding tokens. This objective trains the model to learn to predict the correct answer for the NLU task conditioned on the task-specific prompt and input.

2.3 Proximal Policy Optimization for LLM Fine-tuning on NLU Tasks

We utilize PPO to fine-tune the LLM on NLU tasks, following the training protocol outlined in Appendix B. The reward function is specifically designed for each NLU task. In this work, we use a simple reward function, where a reward is assigned at *the end of the generation* based on alignment with the ground truth labels. We use regular expression matching to extract answers from the LLMs outputs by first locating the text within the '<Judgement></Judgement>' tags. Depending on the task, we then search for task-specific keywords (such as "yes", "no", "acceptable", or "not acceptable") to identify the answer. These extracted answers are compared with the ground truth to determine the appropriate rewards.

For instance, CoLA, a classification task, answers are categorized as *acceptable*, *unacceptable*, or *exceptional* (incorrect format). For STS-B, a regression task, the extracted answer is a floating-point number between 0 and 5. Reward per generation for classification tasks is given by $R = \mathbb{1}(\hat{y} == y_i)$, where $\hat{y}$ is model's prediction and y is ground truth. For STS-B, a regression task, the reward for each generation is defined as $R = 2.5 - |\hat{y}_i - y_i|$, where $\hat{y}_i$ is the predicted score and y_i is the ground truth. Since both scores lie in $[0, 5]$, their absolute difference ranges from 0 to 5. Subtracting this from 2.5 centers the reward around zero and bounds it within $[-2.5,\ 2.5]$, with the maximum reward achieved when $\hat{y}_i = y_i$. *Incorrectly formatted responses* are penalized with a value of -1 for classification tasks and -2.5 for regression tasks, representing the largest penalties applied in each case.

2.4 Low-Rank Adaptation

To mitigate the computational cost of full-model fine-tuning, we employ LoRA (Hu et al., 2021) during both the supervised fine-tuning and PPO stages. Instead of updating the entire model, we restrict the updates to LoRA layers, which significantly reduces the number of trainable parameters by decomposing the weight matrices into low-rank matrices.

3 Experiments

3.1 Experimental Setup

We trained and evaluated our models on the GLUE (Wang et al., 2019) and SuperGLUE (Wang et al., 2020) benchmarks. All experiments were conducted using instruction-tuned LLAMA2-7B models (Touvron et al., 2023a)[2]. We perform both single task and multi-task fine-tuning: 1) *Single-task fine-tuning*: For each subtask within GLUE and SuperGLUE, a separate task-specific LoRA module was trained independently. 2) *Multi-task fine-tuning*: In the multi-task setting, datasets from different subtasks within each benchmark were combined, and a single LoRA module was trained to handle all tasks simultaneously. Please refer to Appendix C for detailed hyperparameter settings.

3.2 Baselines

We evaluated the performance of our approach against three baselines:

- **Encoder-only models**: We compare our results with encoder-only transformer models, specifically BERT-base (110M parameters) and BERT-large (340M parameters) (Devlin et al., 2019).
- **Zero-shot prompting:** The model is provided with task-specific prompts, as outlined in section 2.1, along with the input query. The model is required to generate predictions solely based on these prompts and the input query, without any additional task-specific fine-tuning.
- **Few-shot prompting:** In this setting, the model is provided with both the task-specific prompt and one to five labeled examples (which ever gave the best performance) from the training dataset as demonstrations. These examples are provided as reference to guide the model in generating more accurate responses for the input query. Similarly, no task-specific fine-tuning is performed.

After generating a response, we applied regular expression matching to extract the relevant answer from the model's output. We directly matched task-specific keywords (like "yes" or "no") in the generated text to identify the answer. This extracted answer was then compared to the ground truth label to evaluate the model's performance.

[2]https://huggingface.co/daryl149/llama-2-7b-chat-hf

3.3 Results on GLUE Benchmark

In this section, we present our experiments on the GLUE benchmark, comparing the results with encoder-only models such as BERT (Devlin et al., 2019). We use the LLAMA2-7B-chat-hf model as the LLM for our evaluations. The baselines include zero-shot prompting and few-shot prompting. For fine-tuning methods, we compare both supervised fine-tuning and PPO across single-task and multi-task settings. The results are summarized in Table 1. From the results, we make the following observations.

First, we observed that zero-shot prompting of the LLAMA2-7B-chat-hf model with task-specific prompts consistently underperformed compared to the smaller BERT-base model. LLAMA2-7B-chat-hf struggled notably on simpler tasks like SST-2, which only required classifying sentiment as positive or negative. This underscores the model's weak language understanding capabilities, with zero-shot prompting proving inadequate compared to BERT-base. **Second**, few-shot prompting showed improvements over the zero-shot baseline, achieving an average score of 58.7 compared to 46.1, but it still lagged significantly behind the BERT-base model's score of 79.6. **Third**, supervised fine-tuning (SFT) using LoRA modules for each task further boosted performance, bringing it closer to BERT's level with an average score of 78.5, though still slightly behind BERT-base's 79.6. **Fourth**, fine-tuning with PPO delivered the best results, achieving an average score of 84.6, surpassing even BERT-large's 82.1. Moreover, zero-shot and few-shot prompting of LLAMA2-7B-chat-hf displayed a noticeable output imbalance, with a tendency to favor certain classes or values. In contrast, models fine-tuned with PPO showed no significant bias. **Fifth**, the total computational time for PPO is approximately *1.32 times* that of SFT, indicating only a marginal increase in computational costs.

Additionally, we compared the results with multi-task training, where a *single LoRA module* was trained across all datasets using both SFT and PPO to reduce time complexity. We found that SFT on individual tasks outperformed its multi-task fine-tuning counterpart. However, while PPO on multi-task training did not perform as well as PPO on single-task training, it still outperformed BERT-large in average performance, achieving a score

Models	**MNLI-m**	**MNLI-mm**	**QQP**	**QNLI**	**SST-2**	**CoLA**
BERT-base	84.6	83.4	<u>71.2</u>	90.5	93.5	52.1
BERT-large	86.7	85.9	**72.1**	92.7	<u>94.9</u>	**60.5**
LLAMA2-7B-chat-hf						
Zero-shot prompting	38.3	39.7	31.3	58.5	75.7	18.6
Few-shot prompting	62.4	61.7	30.9	60.7	84.2	29.0
PPO-ST	**88.8**	<u>88.2</u>	70.5	<u>93.2</u>	**96.4**	<u>59.9</u>
SFT-ST	87.0	86.5	63.8	**93.6**	73.8	50.7
PPO-MT	<u>88.7</u>	**88.3**	67.3	90.2	94.6	47.7
SFT-MT	84.9	84.5	62.9	86.0	72.0	41.4

Models	**STS-B**	**MRPC**	**RTE**	**WNLI**	**AX**	**Average**
BERT-base	85.8	88.9	66.4	/	/	79.6
BERT-large	86.5	<u>89.3</u>	70.1	/	/	82.1
LLAMA2-7B-chat-hf						
Zero-shot prompting	27.5	66.3	59.3	44.5	9.2	46.1
Few-shot prompting	45.5	80.8	72.9	51.4	9.2	58.7
PPO-ST	<u>92.6</u>	**89.4**	84.3	<u>74.7</u>	**52.7**	**84.8**
SFT-ST	84.7	85.8	80.4	63.7	<u>45.1</u>	78.5
PPO-MT	**94.7**	86.7	**86.9**	66.4	43.4	<u>82.9</u>
SFT-MT	85.5	82.6	<u>86.2</u>	**76.0**	41.2	76.22

Table 1: GLUE test results are scored by the evaluation server (GLUE benchmark). Average column indicates the averaged performance across all the datasets excluding the WNLI and AX datasets. F1 scores are reported for QQP and MRPC, Spearman correlations for STS-B, Matthew's correlations for CoLA, and accuracy scores for the other tasks. *Zero-shot prompting* refers to prompting with task-specific prompts and an input query, while *Few-shot prompting* refers to prompting with task-specific prompts, 1-5 demonstrations (chosen based on the best performance), and an input query. *PPO* stands for proximal policy optimization, and *SFT* refers to Supervised Fine-tuning. "ST" represents Single-task, while "MT" represents Multi-task. The **bolded** results indicate the best results, and the <u>underlined</u> results indicate the second-best results.

of 82.9 compared to BERT-large's 82.1. These results demonstrate that while single-task fine-tuning yields the best performance, multi-task training with PPO can still achieve competitive results, even surpassing state-of-the-art models like BERT-large.

To assess the consistency of our findings across different models, we evaluated Qwen2.5-7B-Instruct and MPT-7B-chat alongside LLAMA2-7B-chat-hf on the STS-B dataset from the GLUE benchmark and the COPA dataset from the SuperGLUE benchmark. The results are summarized in Table 2. The results confirm that PPO-based fine-tuning consistently outperforms the BERT-large model, as well as the zero-shot and few-shot prompting baselines for all LLMs, highlighting its effectiveness across different LLMs.

3.4 Comparison of RL Algorithms: PPO vs. GRPO

Our objective is to improve the natural language understanding capabilities of the base (policy) model through RL fine-tuning. In this context, we compare two RL approaches: PPO and Group Relative Policy Optimization (GRPO) (Shao et al., 2024). PPO requires updating a separate critic model to compute value functions, which introduces (modest) additional memory constraints. On the other hand, GRPO was designed to bypass the critic model entirely. Instead, GRPO samples multiple trajectories per prompt and computes each trajectory's advantage by comparing its reward to the batch's average (and standard deviation). This method not only simplifies the architecture but also reduces memory usage.

For our experiments, we utilized the TRL library (von Werra et al., 2020) on a single Nvidia A100 GPU, with a batch size of 16 and gradient checkpointing enabled. While SFT involves a simple forward pass, loss computation, and backward pass per step, both PPO and GRPO add extra steps such as LLM sampling, reward calculation, and advantage estimation.

As detailed in Table 3, both PPO and GRPO deliver notable performance improvements over SFT. Notably, PPO only incurs about a 4% increase in per-step runtime compared to SFT. However,

Models	STS-B	COPA
BERT-large	86.5	70.6
LLAMA2-7B-chat-hf		
Zero-shot prompting	27.5	57.0
Few-shot prompting	45.5	73.4
PPO-ST	92.6	88.6
Qwen2.5-7B-Instruct		
Zero-shot prompting	83.7	96.6
Few-shot prompting	87.0	96.0
PPO-ST	92.2	97.0
MPT-7B-chat		
Zero-shot prompting	19.7	57.4
Few-shot prompting	21.7	57.2
PPO-ST	89.3	84.0

Table 2: Performance comparison of LLAMA2-7B-chat-hf, Qwen2.5-7B-Instruct (Hui et al., 2024), and MPT-7B-chat (MosaicML, 2023) models on the GLUE STS-B and SuperGLUE COPA tasks under zero-shot prompting, few-shot prompting, and PPO based fine-tuning. Results are sourced from the official GLUE benchmark and SuperGLUE benchmark evaluation servers. For STS-B, we report Spearman correlation, and for COPA, accuracy is used as the evaluation metric.

GRPO's need to generate multiple responses per sample results in a higher runtime, despite its memory efficiency. Overall, our analysis highlights the trade-offs between these RL algorithms: PPO offers efficient runtime with the cost of additional overhead from the critic model, while GRPO reduces memory usage at the expense of increased sampling time.

3.5 Evaluating Zero-Shot Generalization of PPO Fine-Tuned Models and Comparison with GPT-4o

We evaluate the zero-shot generalization capabilities of LLAMA2 7B and 13B models fine-tuned using PPO on a single dataset and subsequently tested across multiple other datasets (Table 4). For sentiment analysis tasks, the models were fine-tuned on SST-2 and evaluated on diverse datasets, including Financial PhraseBank (Malo et al., 2014), Labelled Financial News (Sood, 2024), Mental Health (Gaes, 2023), and Emotion (Saravia et al., 2018). Similarly, for natural language inference (NLI) tasks, the models were fine-tuned on MNLI and evaluated on Babi-nli (Weston et al., 2015) and SIGA-nli (Nizamani et al., 2024).

Our results demonstrate that PPO fine-tuning improves the zero-shot performance of LLAMA2-chat-hf models compared to **GPT-4o**, a strong baseline. For sentiment analysis, LLAMA2-13B-chat-hf achieves 97.7% accuracy on Financial PhraseBank, slightly outperforming GPT-4o (97.5%). On Labelled Financial News, LLAMA2-13B-chat-hf records 72.3%, exceeding GPT-4o by 4.5%. Similarly, on the Mental Health dataset, LLAMA2-7B-chat-hf achieves 67.2%, marking a notable gain of 7.3% over GPT-4o. For the Emotion dataset, LLAMA2-7B-chat-hf achieves 78.0%, with a smaller gain of 0.4%. For NLI tasks, LLAMA2-13B-chat-hf achieves 69.4% accuracy on Babi-nli, surpassing GPT-4o by 6.2%. Additionally, LLAMA2-13B-chat-hf achieves 46.3% accuracy on SIGA-nli, outperforming GPT-4o by more than 10%. On average, both 7B and 13B versions of PPO fine-tuned LLAMA2-chat-hf models demonstrate a performance gain of over 4% compared to GPT-4o, which is significantly larger in size and highly optimized.

To ensure robust comparisons, we quantify uncertainty in our evaluations by generating 100 predictions for each example in the dataset. The evaluation metric is then computed over the entire dataset for each set, yielding a distribution of values. The 95% confidence interval is defined by the 2.5th and 97.5th percentiles of this distribution. Results are presented in Table 8.

These results demonstrate the effectiveness of simple PPO fine-tuning on a single task-specific dataset in significantly enhancing model performance on similar tasks. LLAMA2-chat-hf models fine-tuned with PPO consistently outperform GPT-4o across diverse downstream tasks, reinforcing PPO fine-tuning as a robust approach for improving the NLU capabilities of LLMs.

We measured inference time on the Financial PhraseBank dataset with a batch size of 4. The BERT-base model, with 110M parameters, required 0.035s per step, while the LLAMA2-7B model, with 7B parameters and multi-token generation, took 0.997s per step. This difference is expected given the larger model size and the need for multiple forward passes in LLAMA2-7B. While LLM inference is slower, our focus is on improving natural language understanding with PPO, which achieves strong performance gains on both in-distribution and out-of-distribution NLU and NLI tasks.

3.6 Evaluation of Instruction-Following in Out-of-Distribution Tasks

To assess the instruction-following capabilities of LLMs in tasks differing from their fine-tuned for-

Algorithm	SST-2	MRPC	RTE	CoLA	QNLI	Avg.	Per-Step Runtime (s)
SFT	73.8	85.8	80.4	50.7	93.6	76.9	4.124
PPO	96.4	89.4	84.3	**59.9**	**93.2**	84.6	4.299
GRPO	**96.7**	**91.2**	**88.5**	55.2	93.1	**84.9**	5.155

Table 3: Comparison of SFT, PPO, and GRPO fine-tuning methods on the LLAMA2-7B-chat-hf model across five GLUE benchmark tasks (SST-2, MRPC, RTE, CoLA, QNLI), along with per-step runtime.

Tasks	LLAMA2-7B PPO-ST	LLAMA2-13B PPO-ST	GPT-4o
Sentiment Analysis			
Financial PhraseBank	97.2	**97.7**	97.5
Labelled Financial News	70.2	**72.3**	67.8
Mental Health	**67.2**	66.6	59.9
Emotion	**78.0**	76.4	77.6
Natural Language Inference			
Babi-nli	68.3	**69.4**	63.2
SIGA-nli	46.2	**46.3**	35.4
Average	71.2 (4.3↑)	**71.5** (4.6↑)	66.9

Table 4: Accuracy of different models across downstream tasks. For sentiment analysis tasks, models are fine-tuned on SST-2 and zero-shot evaluated on Financial PhraseBank (Malo et al., 2014), Labelled Financial News (Sood, 2024), Mental Health (Gaes, 2023), and Emotion (Saravia et al., 2018). Similarly, for natural language inference tasks, models are fine-tuned on MNLI and zero-shot evaluated on Babi-nli (Weston et al., 2015) and SIGA-nli (Nizamani et al., 2024). PPO-ST represents fine-tuning using Proximal Policy Optimization. Gains over GPT-4o model in the average row is indicated with green arrows.

mat, we conduct evaluations using the LLAMA2-7B-chat-hf model fine-tuned on the SST-2 dataset. Specifically, we evaluate the performance of this model on the Amazon review task, which requires generating an integer rating between 1 and 5 based on the provided textual review. Although SST-2 and Amazon reviews both involve sentiment analysis, the two tasks differ distinctly in their input-output formatting, providing a clear measure of instruction-following adaptability.

We compare zero-shot prompting of three versions of the LLAMA2-7B-chat-hf model: the original non-fine-tuned model, a version fine-tuned using SFT, and another fine-tuned with PPO. The 95% confidence intervals (CI) reported here are defined by the 2.5th and 97.5th percentiles of the bootstrap distribution. Using a consistent prompt template across models, we find that the PPO-fine-tuned model achieves an accuracy of 39.35% (95% CI: 38.39, 40.29), significantly outperforming the original model, which achieves 27% accuracy (95% CI: 19.00, 36.03). Conversely, the SFT-fine-tuned model demonstrates extremely poor performance, achieving less than 1% accuracy.

Qualitative analysis of sampled outputs reveals that the PPO-fine-tuned model reliably adheres to the instruction format and generates detailed reasoning to support its predictions. In contrast, the SFT-fine-tuned model often fails to adapt its responses to the required format, demonstrating limited generalization capabilities. PPO fine-tuning maintains proximity to the original model distribution through its clipping mechanism and the KL-divergence minimization term in the objective function (part of the TRL library), thereby preserving and enhancing the model's intrinsic instruction-following capabilities. In contrast, SFT fine-tuning appears to narrow the model's learned distribution to task-specific training data, negatively impacting its original instruction-following proficiency.

Method	Accuracy	95% CI
Non-fine-tuned	27.00	(19.00, 36.03)
SFT	0.00961	(0.00, 0.03)
PPO	**39.35**	(38.39, 40.29)

Table 5: Performance of LLAMA2-7B-chat-hf on the Amazon Review dataset. Best results are highlighted in bold.

3.7 Impact of Fine-Tuning on Language Modeling Ability

We experiment with SFT and PPO to improve NLU capabilities of LLMs and observe improved performance using PPO. However, it is crucial to ensure that fine-tuning methods do not significantly degrade the models' general language generation abilities. To assess this, we directly evaluate the PPL (jel, 1977; Chelba and Jelinek, 2000) of LLAMA2-7B-chat-hf models fine-tuned on the SST-2 dataset using the WikiText-2 test set (Merity et al., 2016), which follows a natural human-written text distribution. We compare these fine-tuned models against the original, non-fine-tuned baseline model, with the expectation that the PPL of the fine-tuned models should closely match the baseline. Our results reveal that the original LLAMA2-7B-chat-hf achieves a perplexity of 6.939. The PPO-fine-tuned model closely maintains this baseline performance with a perplexity of 6.966, indicating minimal impact on its general language modeling capabilities.

In contrast, the SFT-fine-tuned model displays a higher perplexity of 7.384, suggesting a significant reduction in generation capabilities due to convergence toward task-specific training distributions. We conjecture that PPO's clipping mechanism and the KL-divergence minimization term in the objective function (part of the TRL library), effectively constrains policy updates, preventing large deviations from the reference model and thereby preserving the original language modeling capabilities of LLMs. These findings underscore PPO's effectiveness in maintaining the general language abilities of LLMs during fine-tuning.

Method	**perplexity**
Non-fine-tuned	6.939
SFT	7.384
PPO	6.966

Table 6: Perplexity of LLAMA2-7B-chat-hf on the WikiText-2 test set. Lower perplexity indicates better language modeling ability.

3.8 Integrating a Reward Model

While our primary reward function is based on matching generated outputs to true labels, we recognize that more sophisticated reward designs may be necessary for complex NLU tasks. To address this, we investigate the effect of integrating a reward model into our PPO training, with the aim of enhancing not only classification performance on SST-2 but also the quality of generated analyses.

Reward Modeling Setup. For the first 5,000 training samples of the SST-2 dataset, LLAMA2-7B-chat-hf generates four responses per data point. Each response includes a sentiment judgment (Positive/Negative) and a supporting analysis. To robustly rank these responses, we use GPT-4o as an evaluator. GPT-4o ranks the responses based on: (i) the correctness of the sentiment judgment (i.e., matching the ground truth), (ii) the consistency between the judgment and its accompanying analysis, and (iii) the overall factual correctness and helpfulness of the analysis. To ensure clear differentiation, we include two reference responses—one with only the correct answer and one with only the incorrect answer—and define the ranking order as: correct answer with analysis > only correct answer > incorrect answer with analysis > only incorrect answer.

Training the Reward Model. A reward model is then trained on this ranked dataset using a BERT-based architecture (bert-base-cased). For each input x, we consider pairs of responses (y_w, y_l), where y_w denotes a response ranked higher by our evaluator (GPT-4o) due to its correct sentiment and coherent analysis, and y_l denotes a lower-ranked response. The model learns to assign higher scores to better responses via a pairwise ranking loss:

$$L(\theta) = -\mathbb{E}_{(x,y_w,y_l)\sim D} \Big[\log \sigma\big(r_\theta(x, y_w) - r_\theta(x, y_l)\big)\Big], \quad (1)$$

where $r_\theta(x, y)$ is the score assigned to response y given x, and σ is the sigmoid function converting the score difference into a probability. This loss encourages the reward model to output higher scores for responses with superior judgments and analyses.

Incorporating the Reward Model into PPO Training. During PPO training on SST-2, LLM is tasked with generating both a sentiment judgment and an analysis. The trained reward model provides the reward signal by scoring these outputs. As shown in Table 7a, while the PPO model trained with reward signals from the reward model (PPO-RM) produces analyses of higher quality, it suffers from a significant reduction in classification performance, dropping from 96.4% to 89.7%. We believe this discrepancy might be due to the limited sample size used for reward model training and potential reward hacking (Amodei et al., 2016) during optimization. However, we will explore this further in our future works.

Method	Accuracy (%)
PPO	**96.4**
PPO-RM	89.7

(a) SST-2 performance on GLUE.

Method	GPT Eval. Score
PPO	3.479
PPO-RM	**4.104**

(b) Quality of generated analyses.

Table 7: Comparison of reward function designs for LLAMA2-7B-chat-hf. The model trained with a rule based reward (PPO) achieves a high SST-2 classification accuracy of 96.4%, while incorporating a sophisticated reward model (PPO-RM) significantly reduces accuracy (89.7%) but yields substantially improved analysis quality, with a GPT evaluation score of 4.104 compared to 3.479 for the simple reward. Best results are highlighted in bold.

Evaluation of Generated Analyses. To further assess the impact of our reward design, we evaluated the quality of generated analyses. We sampled 100 data points from three models: the original LLAMA2-7B-chat-hf, the PPO model trained using only correct-answer rewards (PPO), and the PPO model trained with the reward model (PPO-RM). GPT-4o then scored each analysis on a scale from 1 to 5 based on answer correctness and logical coherence. As indicated in Table 7b, the PPO model using reward model signals achieved the highest average score, suggesting that a more complex reward function can enhance the quality of generated outputs.

In summary, while the integration of a reward model in PPO training significantly reduces classification performance compared to using only correct-answer rewards, it considerably improves the GPT evaluation scores of the analyses produced by the LLM.

4 Conclusion

Prompting-based approaches, including zero-shot and few-shot prompting, are commonly used to adapt LLMs to downstream tasks. However, our experiments show that when applied to LLAMA2-7B-chat-hf, these methods underperform on NLU benchmarks such as GLUE and SuperGLUE (table 10), often trailing smaller encoder-only models like BERT-base. To address this, we investigate two fine-tuning strategies that update only LoRA layers for computational efficiency: SFT and PPO. While SFT yields modest improvements, PPO provides substantial gains by framing NLU tasks as reinforcement learning problems. PPO-tuned models not only outperform strong baselines like BERT-large but also generalize well across model families and tasks. Notably, PPO-trained LLAMA2-7B-chat-hf outperforms GPT-4o by 10.8% on SIGA-nli and 7.3% on the Mental Health dataset, demonstrating strong zero-shot generalization from single-task fine-tuning. More broadly, we highlight a promising direction: adapting LLMs to new tasks without labeled data by using reward-driven learning. With a well-defined reward function, PPO can steer models toward high-reward behaviors—offering a scalable, label-efficient alternative to SFT.

Limitations

This work takes an initial step toward framing NLU as a reinforcement learning problem for LLMs under 14B parameters. While our long-term goal is to reduce reliance on curated datasets by leveraging richer, task-specific reward models, we currently adopt a simple binary reward signal based on exact label matching. This design enables a controlled evaluation of PPO as an effective adaptation strategy, showing consistent gains over prompting and supervised fine-tuning. Although we present a preliminary exploration of model-driven reward functions in Section 3.8, further research is needed to develop robust and generalizable reward signals that can support more complex or weakly supervised tasks without requiring extensive manual annotation. Overall, our findings suggest that casting nuanced tasks as reinforcement learning problems, through the design of appropriate environments and reward functions, offers a scalable and flexible alternative to standard fine-tuning, particularly when the model is already well-initialized through pretraining.

References

1977. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna

Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, and 1 others. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling. *Comput. Speech Lang.*, 14:283–332.

Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, and 1 others. 2013. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Hugging Face. Trl documentation. `https://huggingface.co/docs/trl/en/index`. Accessed: 2025-09-16.

Joan Gaes. 2023. Depression Dataset on hugging face. `https://huggingface.co/datasets/joangaes/depression`.

Alex Havrilla, Yuqing Du, Sharath Chandra Raparthy, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, Sainbayar Sukhbaatar, and Roberta Raileanu. 2024. Teaching large language models to reason with reinforcement learning. *arXiv preprint arXiv:2403.04642*.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, and 1 others. 2018. Deep q-learning from demonstrations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Zhiting Hu and Tianmin Shu. 2023. Language models, agent models, and world models: The law for machine reasoning and planning. *arXiv preprint arXiv:2312.05230*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala. 2014. Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, 65.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *Preprint*, arXiv:1609.07843.

Volodymyr Mnih. 2016. Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*.

MosaicML. 2023. Introducing mpt-7b: A new standard for open-source, commercially usable llms. Accessed: 2023-05-05.

Rashid Nizamani, Sebastian Schuster, and Vera Demberg. 2024. SIGA: A naturalistic NLI dataset of English scalar implicatures with gradable adjectives. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium. Association for Computational Linguistics.

John Schulman. 2015. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2018. High-dimensional continuous control using generalized advantage estimation. *Preprint*, arXiv:1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017a. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017b. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Arav Sood. 2024. Sentiment analysis - labelled financial news data.

Richard S Sutton. 1988. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023a. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2020. Superglue: A stickier benchmark for general-purpose language understanding systems. *Preprint*, arXiv:1905.00537.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning*, 8:279–292.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart Van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

A Related Works

Policy-based reinforcement learning (RL) directly optimizes an agent's policy by learning its parameters to maximize long-term rewards. Unlike value-based methods like Q-learning (Watkins and Dayan, 1992) and DQN (Hester et al., 2018), which indirectly derive policies through value functions, policy-based methods represent the policy as a parameterized function. This function, $p_\theta(a|s)$, defines the probability of taking action a in state s, where θ represents the policy parameters. The goal is to learn optimal parameters θ^* that maximize the expected cumulative reward, typically through policy gradient methods (Sutton et al., 1999). These methods excel in high-dimensional or continuous action spaces, where value-based methods can struggle (Deisenroth et al., 2013).

Policy-based methods in reinforcement learning (RL) have evolved significantly over time, starting with REINFORCE (Williams, 1992), which optimizes policies using the policy gradient theorem but suffers from high variance due to its reliance on Monte Carlo estimates of the reward. Monte Carlo estimates refer to calculating the total reward based on full episodes of interaction, meaning updates are made only after an entire sequence of actions and rewards is observed, which can lead to noisy and slow learning. To address this, actor-critic methods like A2C and A3C (Mnih, 2016) introduced a critic that estimates the value of the current state, allowing for smoother updates by reducing the variability in policy updates and speeding up convergence. However, these methods still faced instability when large updates caused the new policy to diverge too far from the previous one. Trust Region Policy Optimization (TRPO) (Schulman, 2015) tackled this by limiting the size of policy updates using a KL divergence constraint, but its implementation was complex and computationally expensive. Proximal policy optimization (PPO) (Schulman et al., 2017a) simplified this process by introducing a clipped objective function that keeps policy updates within a stable range while being easier to implement. PPO's balance between simplicity and stability has made it a widely adopted method in modern RL research.

In NLP, PPO has been effectively used in reinforcement learning from human feedback (RLHF) to align LLM outputs with human preferences, as seen in works like InstructGPT (Ouyang et al., 2022) and Constitutional AI (Bai et al., 2022). These approaches treat the LLM as a policy, where model responses are actions, and human feedback serves as rewards. PPO updates the policy based on the reward model trained on human preferences. Additionally, policy-based RL methods have been applied to enhance LLM reasoning capabilities (Ziegler et al., 2019; Havrilla et al., 2024; Hu and Shu, 2023). In this work, we apply PPO to fine-tune LLMs on NLU tasks.

B Preliminaries on Application of PPO for Fine-tuning LLMs

Proximal policy optimization (PPO) (Schulman et al., 2017b) is an online reinforcement learning algorithm. In this section, we describe the process to fine-tune an LLM using PPO. During training, at each timestep t, the LLM (policy) generates a token prediction a_t (action) based on the state s_t, which consists of the sequence of generated tokens up to timestep $t-1$. The final generated output is evaluated in the context of the downstream task, where the environment provides feedback in the form of rewards. The model updates its parameters based on these rewards to improve its ability to generate accurate predictions over time.

Actor Model PPO uses gradient ascent to optimize the following objective, aiming to maximize cumulative rewards:

$$J(\theta) = \mathbb{E}_{(s_t,a_t)\sim\pi_{\theta'}}\Bigg[\min\Bigg(\frac{p_\theta(a_t \mid s_t)}{p_{\theta'}(a_t \mid s_t)}\hat{A}_t, \\ \text{clip}\Big(\tfrac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\Big)\hat{A}_t\Bigg)\Bigg] \tag{2}$$

Here, $p_\theta(a_t|s_t)$ is the probability of taking action a_t in state s_t under the current policy, while $p_{\theta'}(a_t|s_t)$ represents this probability under the old policy. In PPO, the training data—specifically, the state-action pairs (s_t, a_t)—are sampled using the old policy $\pi_{\theta'}$ (the LLM before it is updated), rather than the new policy currently being optimized. Thus, the ratio $\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)}$ accounts for how much the new policy has changed relative to the old policy and adjusts the likelihood of an action accordingly. This ratio is multiplied by $\hat{A}_t$, the Generalized Advantage Estimation (GAE) (Schulman et al., 2018), which measures how much *better or worse* an action a_t is compared to the expected outcome under the current policy.

$$\hat{A}_t = R_t + \gamma V_{t+1} - V_t + \gamma\lambda\hat{A}_{t+1},$$

Here, $R_t + \gamma V_{t+1} - V_t$ represents the temporal difference (TD) error (Sutton, 1988). In this expression, R_t is the immediate reward received after taking action a_t, V_t is the expected reward before the action, and γV_{t+1} is the discounted estimate of the future reward after the action. This term reflects how the action a_t performed when compared to the expected return at state s_t. The second term, $\gamma\lambda\hat{A}_{t+1}$, is the smoothing factor in GAE, where λ is the trade-off parameter. This recursive estimate allows the model to incorporate future information, making the advantage estimate more stable. Smaller values of λ emphasize on immediate rewards, while larger values capture longer-term dependencies. The discount factor γ controls how much emphasis is placed on future rewards compared to immediate ones, with higher values of γ giving more weight to future rewards. V_t, which represents the expected future reward from state s_t, is estimated by a *critic model*.

The clipping function clip(ratio, $1-\epsilon$, $1+\epsilon$) limits the change between the current and old policy, ensuring stable updates by preventing large deviations. This helps avoid too-large policy changes that could destabilize training. In summary, PPO optimizes the policy using gradient ascent to maximize cumulative rewards while ensuring stable updates through clipping, with the GAE providing a more stable and accurate advantage estimate by incorporating future information recursively.

Critic Model The critic model consists of a value head, which is a multi-layer perceptron attached to the final layer of the LLM. It takes the LLMs representation of the generated token sequence up to timestep t (i.e., the state s_t) and predicts a scalar value representing the value function V_t for that state. The critic model is updated using the square of TD error, which is computed as:

$$\delta_t = (R_t + \gamma V_{t+1} - V_t)^2, \tag{3}$$

where δ_t represents the L-2 loss between the actual reward R_t, combined with the discounted estimate of future rewards γV_{t+1}, and the current predicted value V_t for state s_t. By minimizing this TD error via gradient descent, the critic model updates its value function predictions, improving alignment with the actual rewards and future outcomes. In summary, LLM uses the PPO objective to update its policy based on feedback from the critic model, while the critic model is updated to better predict the value function for future states.

We implement PPO using TRL library (Face) that includes a KL divergence term in the PPO objective 2.

C Hyperparameter Settings

For PPO-based fine-tuning, grid search is performed to select the batch size in 4, 8, 12, and 16 for each task. A batch size of 24 was used across all tasks during supervised fine-tuning (SFT). The PPO epoch is set to 4, meaning that each sampled batch is used for updating the model four times. The initial learning rate for all tasks was set to 9×10^{-6}. We utilized the Adafactor optimizer for PPO training and AdamW for SFT. A cosine annealing learning rate scheduler with a warmup phase was employed, where the learning rate was gradually increased during the first 10% of training steps and then reduced to one-tenth of the initial value by the end of training. We use a rank $r = 16$ for the LoRA layers. We trained both PPO and SFT models until convergence on the validation set. The best hyperparameters were selected based on performance on the validation set. The final reported results for the GLUE and SuperGLUE are from their corresponding evaluation server. For evaluation, multinomial sampling with a temperature of 1 was used to generate a single response per data sample. The model generated responses with lengths between 12 and 32 tokens, with the generation process concluding using a special identifier "`</Judgement>`".

For few-shot evaluations, we conducted a sweep over 1–5 shots on the validation set and reported results using the best-performing configuration for each experiment. We ensured that the total input length, including few-shot demonstrations, remained within the model's context window (e.g., in MultiRC, where input lengths can be long). For our experiments, we utilized the TRL library (von Werra et al., 2020) on a single Nvidia A100 GPU.

D Reward Curve for PPO Fine-Tuning

We present the reward curve from fine-tuning LLAMA2-7B-chat-hf using PPO in a multitask setting on the GLUE dataset. Figure 2 illustrates the reward values over training iterations, offering insights into the training dynamics of the model. The curve serves as a key performance metric, tracking the model's learning progress across multiple tasks. The consistent upward trend demonstrates that PPO fine-tuning effectively improves LLAMA2-

Tasks	LLAMA2-7B PPO-ST	LLAMA2-13B PPO-ST	GPT-4o
Sentiment Analysis			
Financial PhraseBank	(96.2, 98.1)	(96.9, 98.5)	(96.6, 98.4)
Labelled Financial News	(66.1, 74.6)	(69.0, 76.6)	(63.2, 72.2)
Mental Health	(66.6, 67.7)	(66.0, 67.1)	(59.3, 60.5)
Emotion	(77.4, 78.6)	(75.8, 77.0)	(77.0, 78.2)
Natural Language Inference			
Babi-nli	(64.3, 71.5)	(65.1, 73.0)	(58.8, 67.6)
SIGA-nli	(39.0, 53.9)	(40.6, 53.7)	(28.5, 42.2)

Table 8: To quantify uncertainty in our evaluations, we generate 100 predictions for each example in the dataset. The evaluation metric is then computed for each set over the entire dataset, forming a distribution of values. The 95% confidence interval is defined by the 2.5th and 97.5th percentiles of this distribution. For sentiment analysis, models fine-tuned on SST-2 are evaluated in a zero-shot setting on Financial PhraseBank, Labelled Financial News, Mental Health, and Emotion datasets. For natural language inference, models fine-tuned on MNLI are zero-shot evaluated on Babi-NLI and SIGA-NLI.

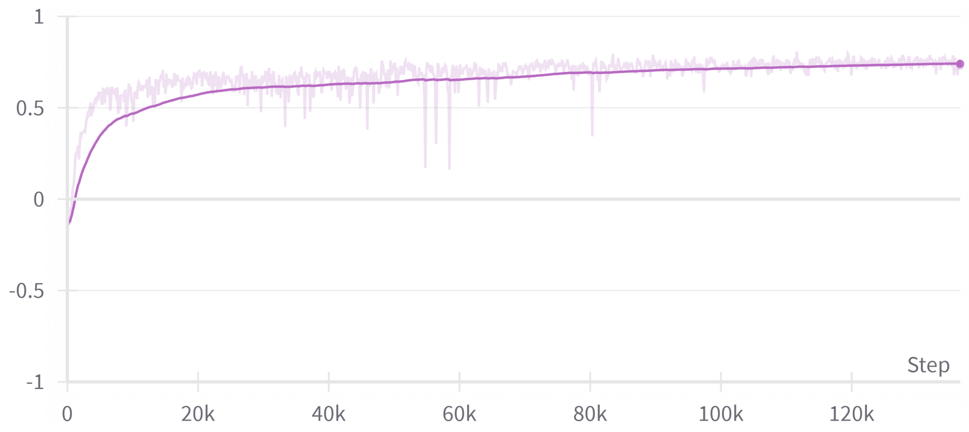

Figure 2: Reward curve for multitask PPO fine-tuning of LLAMA2-7B-chat-hf on the GLUE dataset. The plot illustrates the relationship between training iterations (x-axis) and reward values (y-axis), demonstrating the effectiveness of the PPO optimization approach in improving model performance over time.

7B-chat-hf's ability to generate task-relevant outputs.

E Confidence Intervals on Validation Set of GLUE Benchmark

Our GLUE results are obtained from the official online evaluation servers, which impose a submission limit and keep test-set labels hidden, so applying a bootstrapping procedure directly is not feasible. We compute 95% confidence intervals on the validation sets of the GLUE benchmark (available offline) using the following procedure (used in the paper). For each example in a given dataset, we generate 100 independent predictions. We then compute the evaluation metric over the entire dataset for each set of predictions, yielding a distribution of metric values. The 2.5th and 97.5th percentiles of this distribution define the bounds of our 95% confidence interval. The resulting intervals are reported in the table 9.

F Results on SuperGLUE Benchmark

We fine-tuned the LLAMA2-7B-chat-hf model using PPO on the SuperGLUE dataset and compared its performance against several baselines, including BERT-large, BERT-large++, and zero-shot and few-shot prompting of LLAMA2-7B-chat-hf. The term "BERT++" refers to a BERT model fine-tuned using the supplementary training on intermediate labeled-data tasks (STILTs) approach (Phang et al., 2018), where the model is first fine-tuned on related transfer tasks before being fine-tuned on SuperGLUE tasks. For example, MNLI from the GLUE benchmark (Wang et al., 2019) is used as an intermediate task for CB, RTE, and BoolQ (Wang et al., 2020). In contrast, our experiments with LLM did not use

Dataset	PPO (Eval Metric)	PPO (95% CI)	SFT (Eval Metric)	SFT (95% CI)
CoLA	64.54	(59.24, 70.08)	60.10	(54.79, 65.13)
MRPC	88.41	(86.92, 91.00)	85.28	(84.24, 87.19)
STS-B	88.57	(87.07, 89.98)	85.68	(83.85, 86.72)
QQP	83.34	(82.83, 83.73)	79.76	(79.22, 80.17)
QNLI	93.17	(92.50, 93.88)	93.67	(92.90, 94.35)
MNLI-matched	88.63	(88.05, 89.30)	87.02	(86.32, 87.64)
MNLI-unmatched	88.95	(88.28, 89.53)	87.27	(86.54, 87.89)
RTE	84.90	(80.26, 89.64)	81.95	(77.56, 86.95)
SST-2	96.10	(94.56, 97.54)	96.44	(94.95, 97.37)
WNLI	74.65	(63.30, 85.87)	66.20	(49.30, 71.83)

Table 9: Validation set results on GLUE benchmark. Metrics: CoLA (Matthews corr.), MRPC & QQP (F1), STS-B (Spearman corr.), all others (accuracy). 95% confidence intervals are shown in parentheses.

Models	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE
BERT-large	77.4	75.7/83.6	70.6	70.0/24.0	**72.0/71.3**	71.6
BERT-large++	79.0	**84.7**/90.4	73.8	70.0/24.1	**72.0/71.3**	79.0
LLAMA2-7B-chat-hf						
Zero-shot prompting	75.8	26.4/43.6	57.0	51.9/20.3	27.0/26.2	59.2
Few-shot prompting	80.2	49.8/66.0	73.4	46.6/15.4	36.3/35.3	72.9
PPO-ST	**85.9**	74.7/88.0	**88.6**	**82.5**/50.0	70.6/69.9	**84.3**

Models	WiC	WSC	AXb	AXg	Average
BERT-large	69.5	64.3	23.0	97.8/51.7	69.0
BERT-large++	69.5	64.3	38.0	**99.4**/51.4	71.5
LLAMA2-7B-chat-hf					
Zero-shot prompting	54.4	52.1	9.1	64.0/55.1	49.5
Few-shot prompting	54.4	62.3	9.1	64.0/55.1	54.9
PPO-ST	**72.1**	**78.1**	**52.7**	91.0/**79.8**	**78.3**

Table 10: SuperGLUE test results are scored by the evaluation server (SuperGLUE benchmark). The experimental data for BERT-large and BERT-large++ are taken from the original SuperGLUE paper (Wang et al., 2020). The metrics used in the experiments are as follows: CB: F1 / Acc; MultiRC: F1 / Exact Match; ReCoRD: F1 / Exact Match; AXb: MCC; AXg: Gender parity score / Acc. For the remaining tasks not mentioned, accuracy (Acc) is reported. Average column corresponds to the averaged performance across all the datasets. For tasks with multiple evaluation metrics, we first compute the average of those metrics to obtain a single task score, which is then used in the overall average calculation. The **bolded** results indicate the best results, and the underlined results indicate the second-best results.

this method. Our models were only fine-tuned on the datasets in the SuperGLUE benchmark.

As shown in Table 10, the PPO-tuned LLAMA2-7B-chat-hf achieved the highest average performance, surpassing all baselines. PPO demonstrated particularly strong improvements on reasoning-intensive tasks like COPA and MultiRC, where it significantly outperformed both prompting methods and encoder-only models. These results highlight the effectiveness of PPO in improving the model's capabilities, particularly for tasks requiring reasoning and contextual understanding.

It is worth noting that on MultiRC, few-shot prompting performs slightly worse than zero-shot prompting. This may be because MultiRC involves long input contexts, and incorporating examples in a few-shot prompt can cause the total input length to approach or exceed the LLMs maximum context window.

G Evaluation on Reading Comprehension Tasks

We evaluate LLAMA2-7B-chat-hf on the SQuAD reading comprehension task, where the objective is to select a passage from a given context that best answers a question. Two fine-tuning strategies are compared: SFT, which directly uses the ground-truth answer as the training label, and PPO, which leverages reward functions based on Exact Match (EM) and F1 Score. EM metric is computed by comparing a normalized prediction against the normalized ground truth (with normalization involving lowercasing and punctuation removal); a perfect match yields an EM score of 1, otherwise 0. F1 score measures word-level overlap, balancing how many predicted words are correct (precision) and how many ground-truth words are included (recall).

Models were fine-tuned for one epoch on the SQuAD training set and evaluated on the development set. In our evaluation, zero-shot prompting yields an EM of 7.66 and an F1 score of 32.27. SFT significantly improves these metrics (EM: 59.17, F1: 76.48), while PPO further enhances performance, achieving an EM of 65.74 and an F1 score of 81.82—corresponding to improvements of 6.57 and 5.34 points over SFT, respectively.

These results indicate that optimizing with reward functions based on EM and F1 via PPO leads to further improvements in reading comprehension performance, thereby validating our approach relative to both zero-shot prompting and standard SFT.

Method	**EM**	**F1**
Non-fine-tuned	7.66	32.27
SFT	59.17	76.48
PPO	**65.74**	**81.82**

Table 11: Performance of LLAMA2-7B-chat-hf on the SQuAD dataset. PPO uses Exact Match and F1 as reward signals. Best results are highlighted in bold.

H Comparison of LLaMA2-7B-base and LLaMA2-7B-chat-hf on GLUE Tasks

We fine-tuned LLaMA2-7B-base with PPO and compared it to PPO-tuned LLaMA2-7B-chat-hf on GLUE tasks. The instruction-tuned chat model, with enhanced prompt-following capabilities, outperforms the base model on most tasks, averaging 85.13 vs. 82.51. This highlights the benefit of starting from an instruction-tuned checkpoint for task-specific RL.

Dataset	**LLaMA2-7B-chat-hf**	**LLaMA2-7B-base**
CoLA	**64.54**	57.29
MRPC	**88.41**	80.35
STS-B	**88.57**	86.81
QQP	83.34	**84.39**
QNLI	**93.17**	92.95
MNLI-matched	**88.63**	87.15
MNLI-unmatched	**88.95**	88.22
RTE	**84.90**	83.75
SST-2	96.10	**96.56**
WNLI	**74.65**	67.61
Average	**85.13**	82.51

Table 12: Performance comparison of PPO fine-tuning on LLaMA2-7B-chat-hf and LLaMA2-7B-base across GLUE tasks. Best results for each dataset are highlighted in bold.