

# Reinforcement Learning from Human Feedback as an Alternative to Create Domain Expert A Case Study on Medical LLMs

Arya Kondawar(IMT2020080)

Jainav Sanghvi(IMT2020098)

Monjoy Narayan Choudhury(IMT2020502)

The code for our work is present at:

<https://github.com/MnCSSJ4x/MedGPT>

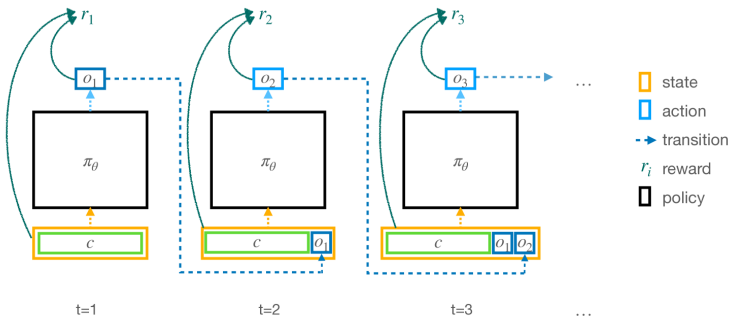
# Problem Statement and Objectives

- ▶ To present Reinforcement Learning from Human Feedback (RLHF) as an alternative to drive the Natural Language Generation model towards favorable generation policy with the help of preference data that is easily available.
- ▶ To benchmark Preference-Based Reinforcement Learning as a paradigm to improve LLMs (especially those with a relatively lesser number of parameters) with easily available preference data.
- ▶ Present qualitative arguments from both human standpoints in the improvement of performance with considerably less training using RL notions and leverage "AI as an evaluator" to automate such qualitative evaluation.

# Value Addition

- ▶ Supervised Fine-Tuning is the defacto method for training LLMs. However, this requires a large amount of data (possibly billions of tokens).
- ▶ Such Data is not available in abundance for domains like healthcare.
- ▶ Standard metrics (BLEU, ROUGE, etc.) are helpful but don't fully capture the qualitative aspects of a "good" medical response. Incorporating human feedback into the training process, can help us leverage this expertise to guide the language model towards more human desirable outputs.

# Formulating Natural Language Generation as a RL Problem



**Figure:** Text generation from LLMs modeled as a Markov decision process. The generation process is auto-regressive, utilizing the token output (action) from the previous time step and the context (state) as input to produce the next token through the language model (policy). A reward function scores the generated output for a given context

Natural Language Generation Tasks can be seen as an RL Problem with the following characteristics.

- ▶ The generation process resembles an agent traversing in an MDP
- ▶ The action space is extremely large and is dependent on Vocabulary Size.
- ▶ The structure of the state space is complex, as a state is essentially a text sequence. The challenge of having good embeddings of text sequence translates to having a good state representation.
- ▶ The reward function used for training can differ from the evaluation reward function as human feedback need not be the same in both stages.
- ▶ The transition function is deterministic (given a state and action).

# Understanding RLHF Paradigm

RLHF is based on Preference-Based Reinforcement Learning (PBRL) and can be explained using the figure below:

Step 1

**Collect demonstration data, and train a supervised policy.**

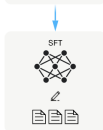
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3 with supervised learning.



Step 2

**Collect comparison data, and train a reward model.**

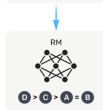
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



Step 3

**Optimize a policy against the reward model using reinforcement learning.**

A new prompt is sampled from the dataset.



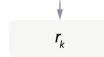
The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



# Reward Modeling Step

- ▶ The Reward Modelling step involves outputting a score for a pair of (prompt, response). Training a model to output a score on a given input is a common task in ML. A simple formulation of regression or classification can be done.
- ▶ In our case, we work with classification where one has to choose one response over the other.
- ▶ In hypothesis the reward model should be at least as powerful as the LLM to be able to score the LLM's responses well. In practice, we achieve this by using the same fine-tuned model. For decoder-based generative models we do this by generating sentence embeddings which are used to generate the logits for the response chosen.

# Reward Modeling - Core idea formulation

Training data format:

$$(x, y_w, y_l)$$

For each training sample  $(x, y_w, y_l)$ :

$$s_w = r_\theta(x, y_w)$$

$$s_l = r_\theta(x, y_l)$$

Loss value:

$$-\log(\sigma(s_w - s_l))$$

Goal: find  $\theta$  to minimize the expected loss for all training samples:

$$-\mathbb{E}_x \log(\sigma(s_w - s_l))$$

where  $d = s_w - s_l$ , and  $s_w, s_l$  corresponds to scores from winning and losing response  $y_w$  and  $y_l$  respectively for the prompt  $x$  and network  $r_\theta$ .

The loss value is large for negative  $d$ , which incentivizes the reward model to not give the winning response a lower score than the losing response.



# Proximal Policy Optimisation

- ▶ The process starts with fine-tuning some or all of the parameters of a copy of the initial LM with a policy-gradient RL algorithm, Proximal Policy Optimization (PPO).
- ▶ Prompts are randomly selected from a distribution. Each of these prompts is input into the LLM model to get back a response, which the RM gives a score.
- ▶ In addition, per-token probability distributions from the RL policy are compared to the ones from the initial model to compute a penalty on the difference between them. This penalty is the KL divergence between the sequences of distributions over tokens. This is exactly similar to TRPO which was discussed in class.
- ▶ **Reason for KL penalty:** The intuition is that there are many possible responses for any given prompt, the vast majority of them the RM has never seen before and might give an extremely high or low score by mistake. This prevents that bias.

# PPO Algorithm

---

**Algorithm 1** PPO, Actor-Critic Style

---

```
for iteration=1, 2, ... do
  for actor=1, 2, ..., N do
    Run policy  $\pi_{\theta_{\text{old}}}$  in environment for  $T$  timesteps
    Compute advantage estimates  $\hat{A}_1, \dots, \hat{A}_T$ 
  end for
  Optimize surrogate  $L$  wrt  $\theta$ , with  $K$  epochs and minibatch size  $M \leq NT$ 
   $\theta_{\text{old}} \leftarrow \theta$ 
end for
```

---

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \dots + (\gamma\lambda)^{T-t+1}\delta_{T-1},$$

where  $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$

No clipping or penalty:

$$L_t(\theta) = r_t(\theta)\hat{A}_t$$

Clipping:

$$L_t(\theta) = \min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta)), 1 - \epsilon, 1 + \epsilon)\hat{A}_t$$

KL penalty (fixed or adaptive)

$$L_t(\theta) = r_t(\theta)\hat{A}_t - \beta \text{KL}[\pi_{\theta_{\text{old}}}, \pi_{\theta}]$$

# Our Pipeline

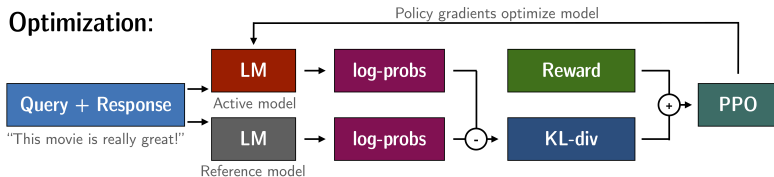
## Rollout:



## Evaluation:



## Optimization:



Our based SFT model that is to be used for all three stages is an instance of "vicgalle/gpt2-open-instruct-v1" on Huggingface. To prompt this model we use the following prompt template:

*Below is an instruction that describes a task. Write a response that appropriately completes the request.*

*### Instruction: Pretend you are a medical expert and answer to the following question - {**query**}*

*### Response:*

Here query is the actual question present in the dataset.

# Dataset Description

- ▶ We use the Comprehensive Medical Q&A Dataset from Kaggle. This dataset consists of about 16000 rows with each row having a question, question type, and answer entry, all in natural language.
- ▶ For the reward modeling dataset since annotating this many rows won't be possible due to time constraints, we create a preference dataset using the base dataset in the following way:
  - ▶ We used the SFT model to create responses to the questions given in the base dataset. This was used as the answer which the user rejected while the actual answer presented was the answer preferred by the user.
  - ▶ This approach of using AI models as a proxy to generating multiple human responses is often used by public LLMs like chatGPT to circumvent the lack of human answer generators and reduce the time taken to generate such answers.

# Experiments

We perform the following ablation based on the size of the generation of text using our Language Model.

- ▶ The first set of experiments are performed on a model which is made where the response to the queries is only allowed to be of at most 200 words/token length
- ▶ The second set of experiments are performed on the models where the response to the queries can be of at most 500 words/tokens.

# Methods of Analysis of Results

To analyse the results generated we present two evaluation techniques.

- ▶ **Qualitative Analysis:** Here we introduce the evaluation paradigm of using "AI as a qualitative evaluator".
- ▶ **Quantitative Analysis:** Here we use the BART-Score metric score.

# General hyperparameters and training details

## ► **Reward Modeling Steps**

- Number of epochs trained - 10 with checkpoint every epoch. We finally stored the model that gave the best evaluation accuracy on the train-eval split of 0.8.
- Used Parameter Efficient Fine Tuning using LoRA to speed up and free memory during the learning process.

## ► **PPO Training**

- Number of epochs trained - 1 as this step takes hours for a single epoch as it consist of interaction with 3 models.
- Used Parameter Efficient Fine Tuning using LoRA to speed up and free memory during the learning process.



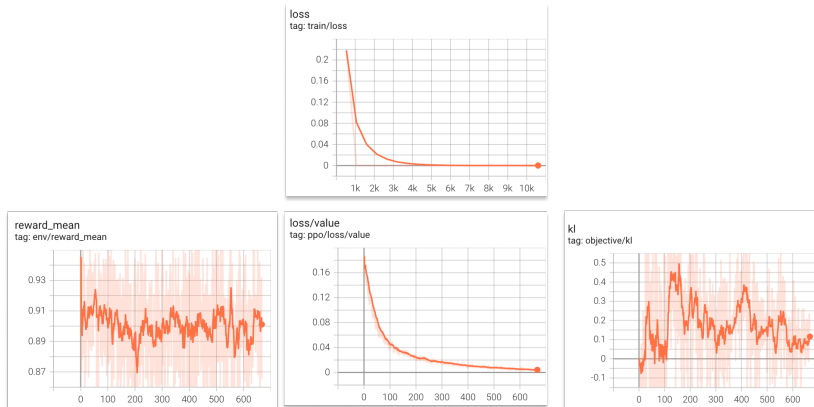
# Training Setup and Time Taken

Our training setup consists of 2x NVIDIA 2090Ti (each of 15GB VRAM) connected in parallel (used for training 200 words model) and 1x Quatro RTX8000 with 49GB VRAM (used for training 400 words model).

The time taken for the 2 models can be seen below:

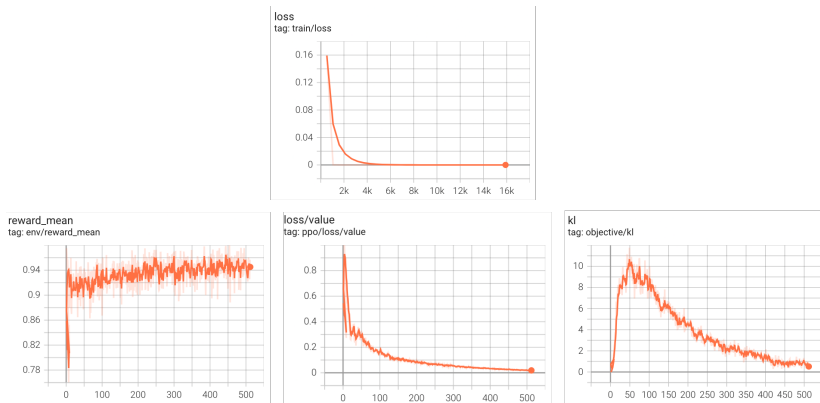
<b>Model Type</b>	<b>Stage</b>	
	<b>Reward Modeling</b>	<b>PPO Step</b>
200 words/tokens	1 hrs 10 mins	4 hrs 48 mins
400 words/tokens	2 hrs 55 mins	14 hr 35 mins

# Model 1 - Generation for 200 words/tokens



**Figure:** Top: Shows the training loss for the reward model training step. Bottom: The first figure from the left shows the Mean Reward which is positive however, it is unstable for the single epoch of the PPO Training step. The loss (in the middle) goes down as expected. KL Divergence initially goes negative leading to irregular results, however, later settles to a positive value leading to a valid model/policy created.

## Model 2 - Generation for 400 words/tokens



**Figure:** Top: Shows the training loss for the reward model training step. Bottom: The first figure from the left shows the Mean Reward which is positive, stable, and higher in comparison to the previous model. The loss (in the middle) goes down as expected. KL Divergence doesn't go negative here and has a much more stable flow.

# Qualitative Analysis

## AI as a qualitative evaluator

To evaluate the model outputs in the case of the SFT model and the model after the PPO step, ideally, we would have domain experts assess the quality of the generated output. However, due to a lack of experts, we devised a strategy to use a production LLM to judge the quality of the prompt. We use the following prompt to set the evaluation process:

*Your job is to evaluate answers from two different virtual assistant and find out which answer is more helpful, truthful, and harmless to human's question. A better answer is usually less repetitive and it can fit well in a dialogue. If an answer looks unnatural or strange in a dialogue, you shouldn't choose it. We expect you to give a decisive answer A or B based on Answer 1 or Answer 2 being better.*

*Question: {question}*

*Answer 1: {answer1}*

*Answer 2: {answer2}*

This prompt is passed via Google's "gemini-1.0-pro" model which is free for experimental use and the output is stored. The following table presents the results for each of the models discussed.

Model Type	Prefers PPO Model	Prefers SFT Model	Total Examples	% Prefers PPO
200 words/tokens	70	58	128	54.60%
400 words/tokens	73	55	128	57.03%

Some good and bad examples can be found in the following link [here](#).

# Quantitative Analysis

## BART Score

BARTScore is a way to evaluate how good a text generation model is by turning the evaluation process itself into another text generation task. It uses a pre-trained model called BART to assess qualities like informativeness, fluency, or factuality of the generated text. It uses the average log-likelihood for target tokens, the calculated scores will be smaller than 0 (the probability is between 0 and 1, so the log of it should be negative). The higher the log-likelihood, the higher the probability.






For example, if TextA gets a score of -1 while TextB gets a score of -100, this means that the model thinks TextA is better than TextB.

Model Type	Mean BART Score	
	SFT Model	PPO Model
200 words/tokens	-4.07321167	-4.07298596
400 words/tokens	-4.023079621	-4.016315414

# Conclusion

- ▶ We present an end-to-end pipeline for training any Language Model using PPO in creating domain-specific language models.
- ▶ More importantly, we see the impact of RLHF on smaller LLMs where Supervised Fine tuning is the defacto method to train which requires a large amount of data.
- ▶ By using an RL algorithm like PPO we can get a better model with relatively less data than what Supervised Finetuning alone could've achieved.
- ▶ The code for our work is present at:  
<https://github.com/MnCSSJ4x/MedGPT>

# References

-  Chaudhari, Shreyas, et al. "RLHF Deciphered: A Critical Analysis of Reinforcement Learning from Human Feedback for LLMs." arXiv preprint arXiv:2404.08555 (2024).
-  Illustrating Reinforcement Learning from Human Feedback (RLHF) - Huggingface Blogs (<https://huggingface.co/blog/rlhf>)
-  TRL Library Documentation for algorithm implementation (<https://huggingface.co/docs/trl/en/index>)
-  Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.
-  Yuan, W., Neubig, G., Liu, P. (2021). Bartscore: Evaluating generated text as text generation. Advances in Neural Information Processing Systems, 34, 27263-27277.