POLYTECHNIQUE MONTREAL
COMPUTER ENGINEERING
COURSE NUMBER: INF8450AE
PROFESSOR: SARATH CHANDAR

APPLYING RLHF TO HUMOR

FABRICE LAMARCHE
LOIC LARROCHE
ZACHARIE GARNIER-CUCHET

Montreal, December 20, 2024

Contents

List of Figures

Applying RLHF to humor

Fabrice LAMARCHE Loïc LARROCHE Zacharie GARNIER-CUCHET

December 20, 2024

Abstract

As part of the Reinforcement Learning course, this work explores the application of Reinforcement Learning from Human Feedback (RLHF), to the behavioural alignment of a language model. Our project aims to develop a generative model based on Microsoft's Phi-3-instruct, capable of producing more humorous and engaging responses using RLHF along with the Proximal Policy Optimization (PPO) algorithm. The originality lies in the construction of a reward model using BERT and the use of the TRL library to optimise the rewards between the generation model and the reward model. The aim is to demonstrate how RLHF can transform a traditional generation model into a more dynamic and fun conversational assistant.

1 Introduction

In this paper, we explore the application of the Proximal Policy Optimization (PPO) algorithm in the context of reinforcement learning with human feedback (RLHF). The main aim of this project is to align a language generation model, `Phi-3`, proposed by Microsoft, so that it produces responses that are more fun and adapted to the interactive context with the user.

The project is based on three main elements:

A reward model: built and trained on the basis of `BERT Base`, it evaluates the 'funny-ness' of the responses generated according to defined criteria.

A generation model : `Phi-3`, derived from Microsoft, is used as the basis for generating textual responses.

An optimisation algorithm: the PPO algorithm is used to adjust the weights of the generation model according to the reward scores, using the `TRL` library.

In this project, RLHF is used to combine the strengths of generation models and reward models, guiding learning to maximise user satisfaction on specific criteria.

The article is structured as follows:

- Theoretical presentation of the RLHF and the PPO algorithms.
- Presentation of the reward model.
- Presentation of the generation model.
- Training the generative model via PPO.
- Evaluation of the generator model

This work illustrates the potential of RLHF for adapting language models to subjective and specific criteria, with an application centered on the creation of humorous responses.

1.1 Reinforcement learning with human feedback (RLHF)

Reinforcement learning with human feedback (RLHF)[2] is a method for aligning a model with specific preferences or goals defined by human users. This approach is based on an interaction loop in which the model's performance is evaluated using a reward signal derived from human preferences. The aim is to guide learning to optimise this signal, even when the final objective is subjective or difficult to formalise directly.

The RLHF process has three main stages:

1. Training a reward model. A reward model is trained to evaluate the outputs of the main model. This model is built from human-annotated data, where pairs of responses are ranked according to how well they match a given criterion. In our project, the responses generated by the `Phi-3` model are evaluated for their 'fun' nature. The reward model we train is not based on human preference data but rather on simple classification of sentence for humor. This will be developped further in Section 2

2. Generation of responses by the basic model. The generative model (`Phi-3`) produces text responses to given prompts. These responses are then evaluated by the reward model to assign a score that reflects the quality or appropriateness of the responses.

3. Reinforcement learning optimisation Using an algorithm such as PPO (Proximal Policy Optimization), the generative model is adjusted to maximise reward scores. Unlike traditional supervised training, where the model learns to directly predict a correct output, RLHF allows the model to explore various outputs and learn to optimise a complex goal defined by human feedback.

In our project, RLHF is used to align the `Phi-3` model to produce more 'fun' and engaging responses. The reward model, based on `BERT Base`, plays a central role in quantifying the humorous aspect of the responses. The use of PPO via the TRL library facilitates the implementation of the optimisation stage by effectively integrating the reward signal into the learning process.

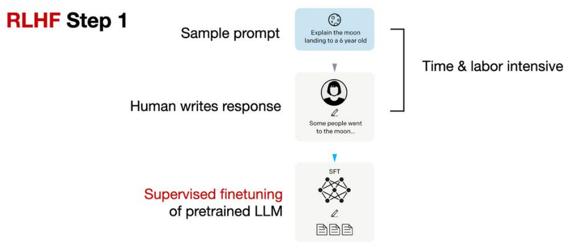
RLHF Step 1
Sample prompt
Explain the moon landing to a 6 year old
Human writes response
Some people went to the moon...
Time & labor intensive
Supervised finetuning of pretrained LLM
SFT

Figure 1: RLHF Step 1. Source : [1].

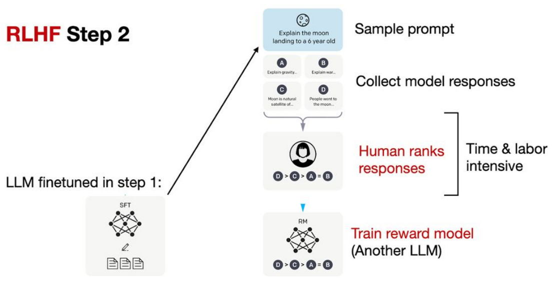

Figure 2: RLHF Step 2. Source : [1].

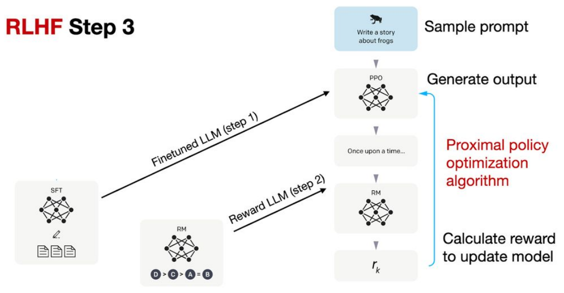

Figure 3: RLHF Step 3. Source : [1].

1.2 Proximal Policy Optimization and TRL

1.2.1 Fundamentals of Proximal Policy Optimization (PPO)

The Proximal Policy Optimization (PPO) algorithm is a reinforcement learning method based on a policy gradient approach. It is based on the idea of updating a parameterised policy $\pi_\theta(a|s)$ by maximising a gain objective while controlling the extent of changes to the current policy to avoid unstable or catastrophic updates.

The main objective of PPO is to balance exploration and exploitation by introducing a penalty on the distance between successive policies. This is achieved using a policy probability ratio:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

PPO maximises a 'clipped' version of the gain objective to avoid excessive updates:

$$L^{\text{clip}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\cdot A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\cdot A_t\right)\right]$$

where A_t is the benefit calculated from the critical estimator, and ϵ controls the maximum size of the changes.

1.2.2 TRL Library: PPO Abstraction and Facilitation

The Transformers Reinforcement Learning (TRL) library simplifies the application of PPO in reinforcement learning with language models. It provides a well-integrated implementation of PPO specifically designed to interact with Transformers models. TRL automates several critical steps in the process, making PPO accessible for large-scale model training. Here's an overview of how it works:

- **Preparing models:** TRL makes it possible to use generation models and easily define an initial policy. The reward model, which guides the alignment, can also be provided in the form of another Transformer, such as a pre-trained BERT model.
- **Collection of trajectories:** TRL generates interaction data (state-action-reward) using the generation model to produce textual output. The generated responses are evaluated by the reward model to provide a quality score.
- **Benefit calculation:** The benefit A_t is calculated from the reward scores. TRL includes an automatic calculation of benefits using estimators such as GAE (Generalized Advantage Estimation).
- **Policy update:** TRL uses PPO's clipped objective to adjust the weights of the generation model according to the calculated benefits. The library handles clipping, gradient control and parametric updates.
- **Distributed optimisation:** TRL is optimised for distributed drives on GPUs.

1.2.3 Integration in our project

In our project, TRL was used to align the `Phi-3` model using PPO. The library enabled:
- Direct integration with `Hugging Face Transformers`, making the management of generation and reward models fluid.
- Automation of PPO updates, reducing the complexity of manual implementation.

- Efficient calculation of rewards from our BERT Base model, without having to implement intermediate steps such as benefit normalisation.
In summary, TRL proved to be a key tool in our implementation, facilitating the application of PPO to adapt our model to subjective criteria such as the production of fun responses.

2 The Reward Model

The reward model plays a crucial role in the RLHF approach by guiding the reinforcement learning agent towards behaviours aligned with defined preferences. In our project, we use BERT as the basic model to build the reward model. We also do not use human ranking of model outputs to train the reward model, meaning that our approach is not exactly identical to RLHF, but rather we use a reward model that aims to capture implicitly some of the preference that a human-annotated model would have represented.

2.1 BERT as a pre-trained model

BERT (Bidirectional Encoder Representations from Transformers) is a language model pre-trained on large amounts of textual data, capable of producing rich contextual representations for each word in a given sequence. Thanks to its bidirectional transformer architecture, BERT can capture global dependencies between words in a given context, making it an excellent choice for tasks requiring a fine-grained understanding of textual relationships.

For our specific purpose, we adapt BERT with a binary classification head. This head consists of a fully connected layer followed by a softmax function, which produces two outputs:

- 1. the probability that the sentence is funny.
- 2. the probability that the sentence is not funny.

By training this model with a dataset annotated specifically for humorous sentences, we hope to obtain a reward model capable of quantitatively assessing how funny a generated response is. This score will then be used as a reward signal for optimising the generation model using the PPO algorithm.

2.2 Dataset

To train the reward model, we used a dataset made up of sentences classified into two categories: serious sentences and humorous sentences (jokes). This dataset provides our model with a variety of examples of sentences, both funny and not funny, so that it can learn to distinguish between the two.

The dataset used comes from Humor Detection on the Kaggle's site and contains a large set of sentences representative of the two classes. Each entry in the dataset has been manually annotated, with labels indicating whether the sentence is perceived as funny or not by the annotators. This dataset is essential for training the binary classification head, which assigns probabilities to each sentence to predict whether it is 'funny' or 'not funny'.

Serious sentences cover a variety of contexts, including factual statements, descriptions and general observations, while jokes are taken from a variety of sources such as joke websites, humour forums and other online resources. These two types of sentences are balanced in the dataset to ensure effective training of the model. One large caveat of this dataset is that it contains a large amount of toxic data, particularly in the 'funny' sentences, where we observed a lot of racist and misogynistic 'jokes' based on stereotype. Given more time we would have liked to have pre-processed this dataset to eliminate toxic examples.

The exact source of this dataset is available in the bibliography under reference [3].

2.3 Traning and result

After training the reward model, we evaluated its performance in predicting the 'funniness' of sentences, using the probabilities generated by the binary classification head. To evaluate our model, we tested it on several sentences to observe its predictions. Here are a few examples of the sentences tested with the associated probability results :

- "Why did the scarecrow win an award? Because he was outstanding in his field!"
Funny probability = 0.98

- "Oh no, I've been forgotten again! Time to wriggle my way back into the spotlight!"
Funny probability = 0.34

- "Plants need sunlight, water, carbon dioxide, and nutrients from the soil to grow."
Funny probability = 0.07

These results illustrate the model's ability to classify sentences according to their humorous nature, with scores consistent with expectations. The model obtained an accuracy score of 0.98 on the dataset's test set, meaning that it probably overfit to the task, but this is not too problematic for our application since we hypothesize that the model will still be able to capture 'funny-ness' of model outputs.

3 The Generator Model

The generation model used in our project is the Phi-3-mini-128k-instruct, available from Hugging Face. This model is part of the family of *instructed* models, designed to follow instructions given by the user and generate appropriate responses. It is particularly well suited to tasks such as instruction-driven text generation, where the aim is to produce accurate, informative or creative responses to the given context.

The Phi-3 128k Instruct model is based on the transformer architecture, and has been trained with a wide variety of data including dialogues, instructions and example responses. This wide diversity enables the model to understand and respond to complex queries with consistent and relevant answers. In addition, it is pre-trained on a generalist dataset, which allows it to be easily adapted to specific tasks, such as aligning with human preferences in scenarios like RLHF.

The model was particularly well suited to our task, where the aim is to produce humorous responses based on the reward model's evaluations. By combining this model with the PPO algorithm, we aim to refine the responses generated to make them more 'funny' and engaging with the user.

We also chose this model because it is one of the smallest model capable of generating coherent outputs to user's prompts. This made it possible to train the model using quantization[4] and LoRA [5].

The Phi-3 128k Instruct model is available on the Hugging Face platform[6].

4 Generator Training with TRL

In this project, we implemented the training of the text generator using the Transformer Reinforcement Learning Library (TRL). TRL provides a structured framework for applying reinforcement learning algorithms like Proximal Policy Optimization (PPO) to fine-tune language models. PPO, a policy-gradient method, allowed us to optimize the generator by directly aligning its outputs with a custom-defined reward function, focusing on improving specific aspects of the generated text.

Additionally, we adopted Parameter-Efficient Fine-Tuning (PEFT) techniques to train the generator efficiently without requiring a full retraining of the model's parameters. Specifically, we used LoRA, which leverages quantization and low-rank adaptation. This enabled us to fine-tune a large-scale pre-trained language model while reducing computational costs and memory requirements. By focusing on a subset of parameters, this method maintained the original model's structure while enhancing its ability to generate text aligned with the reward signal.

4.1 Prompt Collection for Initial States

An important component of training our LLM with TRL involved designing a diverse set of initial prompts to serve as starting points for text generation. These prompts act as the initial state in our framework, ensuring the model explores a wide range of contexts during training.

To achieve this, we collected a total of 4,500 unique prompts:

2,500 prompts were generated using ChatGPT [7], designed to encourage the LLM to produce content that could be perceived as humorous. These prompts were carefully crafted to stimulate creative and varied responses, enhancing the model's ability to understand and generate humor. Here are few examples :
- "Write a joke about superheroes."
- "What would you do if you laughed in a serious meeting?"
- "Hit me with your best dad joke!"

1,000 prompts where also generated by ChatGPT with the aim to provide serious (non-funny) outputs from the model. Examples are :
- "How many months are in a year?"
- "How do I handle a difficult colleague?"

The last 1,000 prompts were sourced from the SciQ dataset, which consists of serious prompts focused on scientific or factual topics. These prompts aimed to anchor the model's output in structured thinking ([8]).

This diverse set of prompts was instrumental in allowing the model to begin generation from various initial states, which we hypothesize contributes to better generalization across tasks. The inclusion of both humorous and serious prompts ensures the model does not overfit to a single style or tone, broadening its utility and robustness.

4.2 training evaluation

To evaluate the effectiveness of our training process, we monitored two key metrics: loss and reward over the course of the training epochs. The Figure 4 illustrates the variation of loss and reward during training. It provides a visual representation of the interplay between the model's optimization and its performance improvement.

Each Epoch here refer to a single step from the PPO-Trainer. We used a batch size of one once again because of compute limitations.

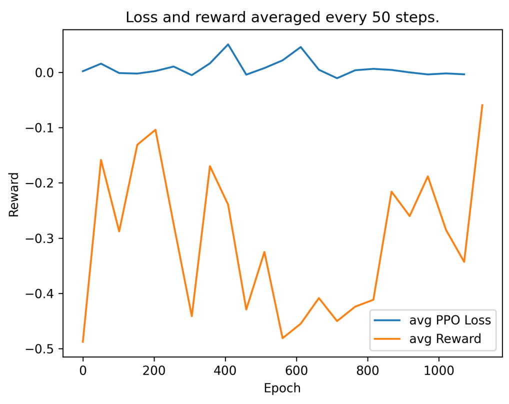

Figure 4: Loss and reward evolution trough training

The loss shows a slight decrease over the training epochs, indicating gradual optimization of the model's parameters. Although the decrease is modest, it suggests that the model is adapting to the task's requirements. Despite exhibiting significant variance during training, the rewards show an overall upward trend. This indicates that the generator is increasingly producing outputs that align with the reward function, albeit with some fluctuations in performance.

While performance seems to be increasing during training, we will evaluate the performance futher in Section 5.

4.3 Training limitation

Due to computational constraints, we were unable to complete the training of our generator beyond around 1000 epochs. Specifically, the training environment on Google Colab offering GPU support automatically disconnected after a maximum runtime of approximately 2 hours, limiting our ability to extend the optimization process.

While we explored alternative solutions, such as using paid services, we were unable to secure the necessary GPU allocation to continue training within our available means. As a result, our reported results reflect the progress made up to the 1000th epoch, and we believe longer training could have resulted in further improvements.

This limitation highlights the critical role of computational resources in achieving comprehensive training and underscores the challenges faced in resource-constrained environments. Future work could benefit significantly from access to more robust infrastructure to fully evaluate and refine the model's capabilities.

5 Evaluation

To try to truly appreciate if the trained model was able to express more humorous completions. To do so, we randomly selected 25 'funny' prompts and 25 'serious' prompts that were not part of the training sample, and prompted both the initial and the trained model to complete them. Next we measure the reward they obtained from our reward model, with results shown in Figure 5. What we can observe is that the trained model obtains more reward overall, which signifies an improvement in overall 'funny-ness'. However, interestingly it gets lower rewards from funny prompts and significantly higher rewards from serious prompts, which might indicate that the model was able to achieve humour from non-funny leading prompts. However, we decided to manually look at the model's answers, and unfortunately we saw no significant difference in the responses : we manually annotated the completions from these 50 sentences as to wether they were funny or not. We observed the following : for both models, they mostly generated jokes from the 'funny' prompts that made no sense or that were unsuccessfull attemps at humor. Our trained model and the initial model achieved successfull humor in respectively 2 and 3 instances. For 'serious' prompts we observed zero funny answers from either models. This disapointingly means that the trained model was probably able to gain higher rewards without really generating more humorous responses, indicating flaws in either our reward model or our training approach.

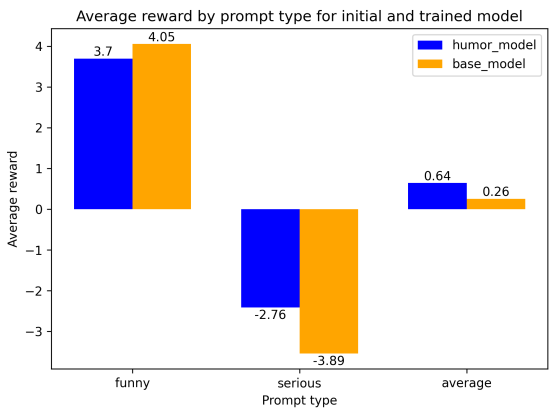

Figure 5: Reward difference between initial and trained model.

6 Conclusion

Our trained model is available on HuggingFace Hub (https://huggingface.co/fbl10/humor-rlhf).

All of our training scripts, figures, training data and annotated outputs are available on Github (https://github.com/FBL10/humor-rlhf).

In this project, we implemented a method to train a language model for humorous text generation using reinforcement learning. Our approach involved a reward model based on BERT Base, fine-tuned with a humor detection dataset using a classification head. For the generator, we leveraged the Phi-3-mini-128k-instruct model, chosen for its capacity to handle extensive prompts and generate complex outputs. The training was conducted using Proximal Policy Optimization (PPO), implemented through the TRL library, and relied on a diverse dataset of 4,500 initial prompts that we carefully curated to ensure broad generalization.

While the methodology was robust and grounded in established techniques, we regret not achieving satisfactory results due to limited computational resources.

Although we could not complete the training or test our generation model comprehensively, we believe that with adequate resources, this approach could yield promising results. Future iterations of this work would benefit greatly from extended training time and computational capacity, allowing the model to refine its ability to produce high-quality, contextually humorous text.

Despite these challenges, this project provided valuable insights into the challenges of reinforcement learning for language models and the potential of tailored datasets in shaping model behavior.

Acknowledgments

We would like to thank SARATH CHANDAR and the teaching team of the INF8450AE course for their support and dedication throughout this semester. We found the course to be very interesting and challenging, and the teaching material to be of great help. We also extend our thanks to the teaching assistants for their availability and valuable explanations.

References

[1] Everyday Series Team. Understanding llm training, rlhf, and its alternatives. https://everydayseries.com/understanding-llm-training-rlhf-and-its-alternatives/, 2023.

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

[3] Eric Chen. Humor detection. https://www.kaggle.com/datasets/amaanmansuri/humor-detection, 2020.

[4] Benoit Jacob, S. Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew G. Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017. doi: 10.1109/CVPR.2018.00286.

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

[6] Hugging Face Team. microsoft/phi-3-mini-128k-instruct. https://huggingface.co/microsoft/Phi-3-mini-128k-instruct, 2024.

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, J. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. Henighan, R. Child, A. Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Ma teusz Litwin, Scott Gray, B. Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, I. Sutskever, and Dario Amodei. Language models are few-shot learners. *Neural Information Processing Systems*, 2020.

[8] Hugging Face Team. Datasets : allenai/sciq. https://huggingface.co/datasets/allenai/sciq, 2024.

[9] raigon44. Joke-generation-and-rating-using-llm. https://github.com/raigon44/Joke-Generation-and-Rating-Using-LLM/, 2023.

[10] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, pages 1–12, 2017. URL https://arxiv.org/abs/1707.06347.

[11] Sergio Paniego and Hugging Face Team. Hugging face - trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2024.