6.884 Final Project: AdaPTing language models to natural language feedback

Arkadiusz Balata
arkadius@mit.edu

Binwei Yan
bineva@mit.edu

Roger Jin
rogerjin@mit.edu

Abstract

We propose a general method for fine-tuning language models in response to natural-language feedback - so that, in theory, a user could say "be funnier" to GPT-2 and GPT-2 would update its weights to produce a funnier generation. Our method relies purely on pre-trained language models and does not require any data collection or access to feedback-specific classifiers. We hope that our method can allow language models to behave in a more human way, as well as users of language models to interact with language models in a more human way—in words rather than equations.

1 Introduction

Traditional language models generate text left to right, iteratively generating a probability distribution over the next tokens given the current prefix, picking a next token from the distribution, and then appending it to the prefix. Such a system can then be trained via traditional continuous optimization techniques via MLE on $(\text{prefix}, \text{next token})$ pairs on a massive text corpus. Emergent properties appearing at various scales of massiveness include sample-efficient domain transfer (Radford et al., 2019), few-shot learning (Brown et al., 2020), and even simple instances of chain-of-thought reasoning (Wei et al., 2022). Composing the descriptive abilities of multimodal $\{\text{other domain}\} + \text{text}$ models with the reasoning capabilities of massive language models have led to performance gains in both image and video captioning (Zeng et al., 2022). Causal cross-entropy loss has gotten us this far, but modern LMs are not devoid of flaws.

1.1 What LMs still need to learn, and why RL is the future

Just as a language model learns to produce a description of a zebra from the Wikipedia articles and so forth it is trained on, a language model also learns to generate racially charged diatribes from toxic language also present in its traning data (Liang et al., 2021). As toxicity is an emergent property of a full piece of text rather than a property of a single next token, any attempt to apply gradient descent to toxicity minimization of a multi-word generation must somehow backpropagate through a selection operator. In the case of greedy decoding, this selection operator is the argmax function, which is piecewise constant and thus has gradient of zero almost everywhere. In the case of sampling, this selection operator samples from the categorical distribution outputted by the model over the vocabulary. Categorical distributions admit reparameterization in terms of Gumbel random variables (Jang et al., 2016), similar to the reparameterization trick used in the training of variational autoencoders (Kingma and Welling, 2013), but in practice, a more popular approach is to frame language modeling as a Markov decision process (MDP), where states are prefixes and actions are next words. Under this interpretation, a language model is a stochastic policy on this MDP and is thus trainable via policy space methods from reinforcement learning. The key difference between the RL framework and the supervised learning framework, which is relevant here, is that in RL, rewards are treated as constant with respect to the policy parameters in the policy gradient theorem, while in supervised learning, losses must be differentiable with respect to the policy parameters.

Using this framework, Perez et al. recently trained language models to generate prompts which induce toxic generations in a target language model. Other emergent properties of full generations that have recently been realized by framing these properties as rewards to RL agents have been code correctness in the context of competitive programming (Li et al., 2022) and alignment with human preferences (Ouyang et al., 2022).

1.2 Towards natural-language supervision of language models

Suppose that we have a pre-trained language model that we like, but we would like its generations to have a bit more humor to them. Humor is an emergent property of a full generation rather than a property of any particular next token, so we can turn to RL to fine-tune our LM. However, we somehow need a reward signal, and for that we would need to train a reward classifier, perhaps on data labeled by expensive and possibly noisy human annotators. If we wanted later on to make future generations more empathetic, or less biased, or more Shakespearean, we would need specialized classifiers for all of these cases. It would be nice if we could just say to our language model "Be funnier," or "Be kinder," and it would update its weights to react to those pieces of feedback.

We propose AdaPT, an initial attempt at a framework which enables users to give natural language feedback to language models in the absence of specialized scoring models. The main idea is that a language model already has some innate understanding of "funniness" and "kindness," and our assumption is that critical feedback to the model, such as "Be funny" is more likely to follow a generation for which the criticism applies than one for which it does not (i.e. "Be funnier" is more likely to follow a boring sentence than a funny one). We thus expect that if we decrease the probability, in expectation across all generations, that a generation is followed by some piece of critical feedback, the model will have effectively updated in a direction that eliminates the need for the feedback. We can thus use a (fixed-parameter) language model to compute $-\log \mathbb{P}\,(\text{feedback}|\text{generation})$, and use this as a reward for the language model we are interested in training.

2 Methodology

Generally, we choose to finetune on GPT-2 (due to the limitation of GPU memory,) using the stable-baselines3 (Raffin et al., 2021) implementation of the PPO algorithm(Schulman et al., 2017), which actually uses PPO as a policy optimization method and the Actor Critic framework as a framework. In order to incorporate the natural language feedback, we add a "feedback prompt" before the feedback, and then calculate the probability of the feedback after the "feedback prompt".

For example, with some prompts like p = "How was your day?", we first generate some context after the prompt, getting g = "I had a good day!" With these generated contexts, we then add the feedback prompt (fp) ="The feedback of the above contexts is:" and the feedback f being "be sadder" to the end of the generation. Now, putting p+g+fp to the reward model(which is also another GPT2 model in this case), the reward of the each sentence is the $-log(Pr(f|p,g,fp))$. We hope that will help the model avoid the same reward again.

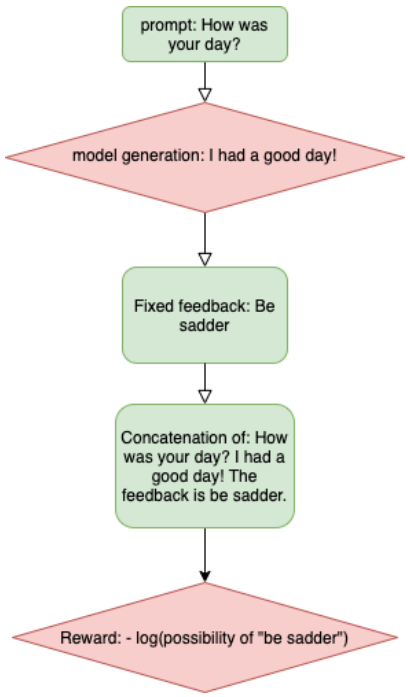


Figure 1: Diagram

We tried two ways to generate the rewards. In the first version, we minimize the possibility of

$$\Pr(f|p,g,fp)$$

, while in the second version, we minimize the possibility of

$$\Pr(p,g,fp,f) = \Pr(f|p,g,fp) * \Pr(p,g,fp)$$

as an importance sampling

2.1 PPO Algorithm

Proximal policy optimization (PPO) is an algorithm using fixed-length trajectory segments. T timesteps of data are collected by each N (parallel) actor in each iteration. We then construct the surrogate loss using these NT timesteps of data and optimize it for K epochs using minibatch SGD.

2.2 Actor and Critic

In the actor and critic framework, the actor is the policy which is being optimized. The critic is the value function which is being learned.

In this model, inherited from GPT-2, we extract the features of the sentence to be the hidden layer of the next token by first generating the next token and then generating the corresponding hidden features of this token, which is of 768 dimensions.

The action-net is set to be a fully-connected layer inherited from the classifier layer in the pertained model. The value network is set as a linear layer from the hidden layer to the final value, with dimensions of 768 to 1.

2.3 Environment and Feedback

The environment is another GPT2 model extracted from pretrained models. With the concatenation of prompt and generations and feedback prompt and feedback, we mask the positions of the feedback and get the loss of the generation. Intuitively, we hope to eliminate the possibility of generating the same feedback. The reward only occurs when the generation reaches the final token, making it sparse.

3 Evaluation

3.1 Experiments

3.1.1 Utterance Feedback

The general hypothesis of what our model could achieve was that we could incorporate utterances as feedback and command a language model to behave differently. In our first set of experiments, we trained the model with the feedback "Be happier." and "Be sadder." In order to evaluate whether the models behaved with the emotions we wanted them to, we finetuned BERT-base (Devlin et al., 2018) using the GoEmotions dataset (Demszky et al., 2020). This allowed us to classify generations into one of 27 emotions or neutral. These fine-grained emotions can map into the coarse Ekman emotions (Ekman, 1992) of anger, disgust, fear, joy, sadness, and surprise. To evaluate, we generate 1000 generations after training of our model is complete. We run emotion classification inference on the generations. Table 1 shows the average probability of each emotion over the 1000 generations - with the aforementioned feedback and using the base GPT2 model on the prompt "Tell me a story." Table 2 does the same, but for the prompt "How was your day?"

We observe that the average probability for each emotion is not in line with what we would expect. We expect that "Be happier" feedback would cause the average probability of joy to increase relative to the control. However, in both cases, it decreases. Something similar can be said for the average probability of sadness - it does not increase with the "Be sadder" feedback. We note that instead of large changes in any emotions, the feedback-adapted generations seem to be more neutral. This may be caused by the model expecting such phrases only when its preceding text is not neutral. For example, the probability of "Be happier" is minimized not when the model is happy, but when the model is neutral, since neutral text is infrequently turned emotional, but happy text can be asked to be turned into an even happier text. However, this effect may also be due to a limitation in model capacity, which will be discussed in section 4.

Emotion	Control	Happy	Sad
Anger	0.087	0.092	0.047
Disgust	0.005	0.008	0.004
Fear	0.006	0.005	0.003
Joy	0.167	0.100	0.108
Sadness	0.033	0.030	0.015
Surprise	0.249	0.262	0.159
Neutral	0.454	0.503	0.664

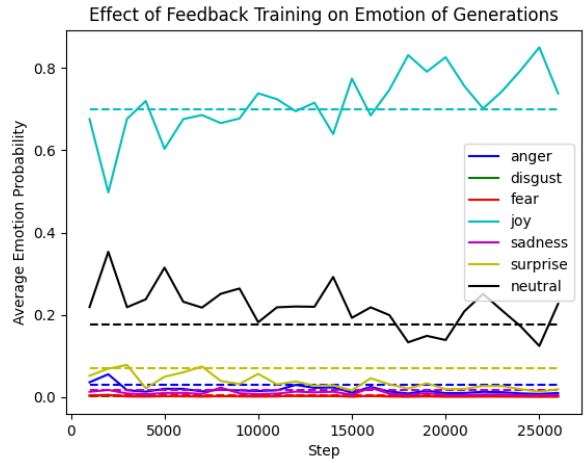
Table 1: Comparison of Average Emotion with the prompt "Tell me a story." using no feedback (control), "Be happier" (happy), and "Be sadder" (sad) as the feedback.

Emotion	Control	Happy	Sad
Anger	0.027	0.031	0.024
Disgust	0.002	0.002	0.002
Fear	0.002	0.001	0.003
Joy	0.367	0.345	0.417
Sadness	0.018	0.012	0.018
Surprise	0.451	0.404	0.361
Neutral	0.133	0.205	0.175

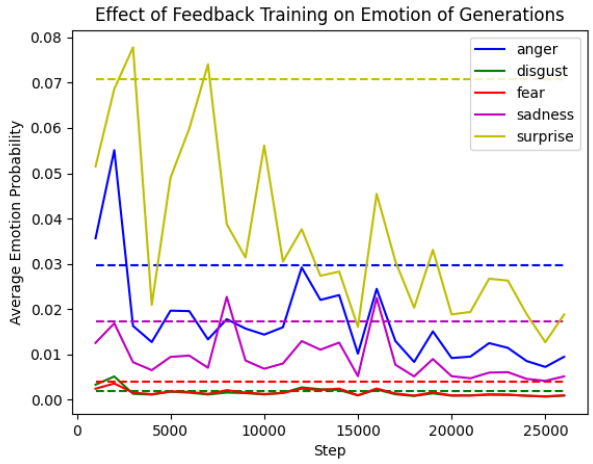
Table 2: Comparison of Average Emotion with the prompt "How was your day?" using no feedback (control), "Be happier" (happy), and "Be sadder" (sad) as the feedback.

3.1.2 First-person Feedback

Since utterances appear to not work well with our model, we also opt to test first-person feedback. This is defined as feedback that is written from the



(a)



(b)

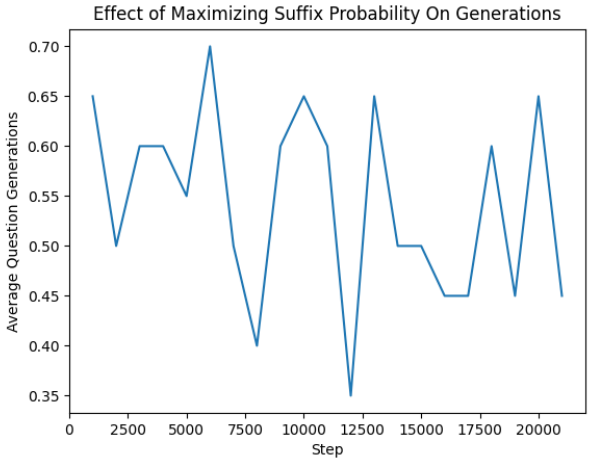
Figure 2: (a) A graph showing how the mean probability of a generation having each emotion changes with training time and (b) a zoomed in version of (a) without the most probable emotions (provided for clarity). The dashed lines represent the baseline (without feedback) emotion of generations with this prompt.

point of view of the model.

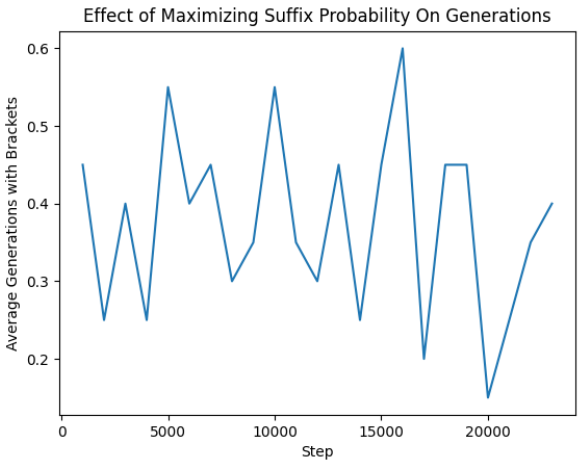
With the same evaluation setup as before (using a pretrained emotion BERT classifier), we prompt the model with "My favorite food is" and give the feedback "Ewwww gross!!". Instead of evaluating the model after training concludes, we evaluate the emotions of generations every 1000 steps of training. In figure 2, we observe that with this feedback, the average probability of joy increases with training time. The probability of all other emotions and neutrality decreases as well. Our goal was to decrease the probability of "disgust" being an emotion. It is not visible in the graph, but this probability decreases by about one order of magnitude.

It is noteworthy that this type of feedback reaches our goal of constructing specific types of generations better than utterance feedback. However, this method has the limitation that the feedback must be constructed carefully and from the model's point of view, which may feel unnatural to humans giving the feedback.

3.1.3 Suffix Enforcement



(a)



(b)

Figure 3: (a) A graph showing how the probability of a generation being a question changes with training time and (b) a graph showing how the probability of a generation having brackets changes with training time

Suffix Enforcement is the idea that instead of minimizing the probability of the feedback in the language model, we can maximize it. This way, the model can generate generations that end in the

way that the feedback suggests. We ran two experiments with this type of feedback. For both experiments, we prompt the model with the end of the text token. Then, we provide "?" and "(Andreas et al. 2022)" as the feedback. We hypothesized that the "?" feedback would cause the model to generate more questions as opposed to statements in the aggregate, and that the "(Andreas et al. 2022)" feedback would cause it to generate more citations/use more brackets in its generations. The results for this experiment are seen in figure 3. As proxies for whether a question was asked, we look for sentences that end in a question mark or start with a question word ({who, what, when, where, why, how}). As a metric for whether more brackets were used, we counted what portion of generations had opening and closing brackets. We find that training time has no relation to these metrics, implying that our method could not be used to enforce generations to end with an input suffix.

4 Conclusion

It seems that this method is not working for multiple reasons. As is addressed in the paper recently, (Scheurer et al., 2022), small language models do not really care about the feedback. Even though this might be a good way of finetuning the large language models, our failure implies the incapability of small models to learn from themselves.

We have also observed that more fluent generations (like "Ewww gross") work better than commands like "Be happier", which also implies that the small model is not used to command-like conversations.

Even though our method is not working as we expected, we have illustrated that future work on language models based on human feedback should definitely use larger models. It also might be possible to involve human feedback in the process of distillation or the pre-train process. In conclusion, we proposed a way to train on human feedback with not so good results for multiple reasons, but there is still space for future work.

Acknowledgments

We would like to thank Professor Jacob Andreas for a great semester and very helpful guidance on this project.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *arXiv*.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan S. Cowen, Gaurav Nemade, and Sujith Ravi. 2020. Goemotions: A dataset of fine-grained emotions. *CoRR*, abs/2005.00547.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Paul Ekman. 1992. An argument for basic emotions. *Cognition and Emotion*, 6(3-4):169–200.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical Reparameterization with Gumbel-Softmax. *arXiv*.

Diederik P. Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. *arXiv*.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson D'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. 2022. Competition-Level Code Generation with AlphaCode. *arXiv*.

Paul Pu Liang, Chiyu Wu, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2021. Towards Understanding and Mitigating Social Biases in Language Models. *arXiv*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *arXiv*.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. 2022. Red Teaming Language Models with Language Models. *arXiv*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8.

Jérémy Scheurer, Jon Ander Campos, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. 2022. Training language models with natural language feedback.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. *arXiv*.

Andy Zeng, Adrian Wong, Stefan Welker, Krzysztof Choromanski, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, and Pete Florence. 2022. Socratic Models: Composing Zero-Shot Multimodal Reasoning with Language. *arXiv*.