STOR 390: Final Project

2024-05-03

Introduction

Within the past two years, we have seen the rise of complex large language models such as ChatGPT, Gemini, and Copilot. According to Reuters, ChatGPT reached 100 million active users only two months after launch, becoming the fastest growing software application ever. Individuals use LLMs to automate simple tasks, solve questions, provide feedback on works, and much more. Companies, throughout different industries including tech, retail and manufacturing, have also started implementing LLMs into their products.

Summary of Method

With the rise of LLMs also comes the increasing risk of LLMs generating negative and harmful results. In order to know what prompts this response and prevent it, LLM researchers conduct a technique called red teaming where human testers deliberately attempt to trigger harmful outputs. However, using human testers as a red team is incredibly costly and time-consuming. In response, recent developments have attempted to use an outside LLM as a red team which is trained through reinforcement language. However, these red team LLMs are incredibly limited in their coverage of test cases and only produce a small number of effective test cases. Ultimately, this leads us to the following research paper. Researchers at MIT decided to continue using separate LLMs as red teams, but they trained them through a method called curiosity-driven reinforcement language. To give some background, in traditional reinforcement language (RL), an agent (in this case, the red team LLM) learns to make decisions through trial and error. The agent is either rewarded or penalized for each decision until it finally reaches its goal. Usually, the goal of traditional reinforcement language is a minimized or maximized output - in this case, the goal for the red team LLM is to maximize toxic responses. In curiosity-driven reinforcement learning, the agent is rewarded for pursuing curiosity and exploring its environment. In this paper, researchers trained the red team LLM to explore a wide variety of prompts with the hope of expanding coverage of test cases. The following traditional reinforcement language optimization formula had been the standard when using red-team LLMs:

$$\max_{\pi} \mathbb{E}\left[\left(R(y)\right) - \beta D_{KL}\left(\pi(\cdot\,|x_2) \parallel \text{ref}(\cdot\,|x_2)\right)\right]$$

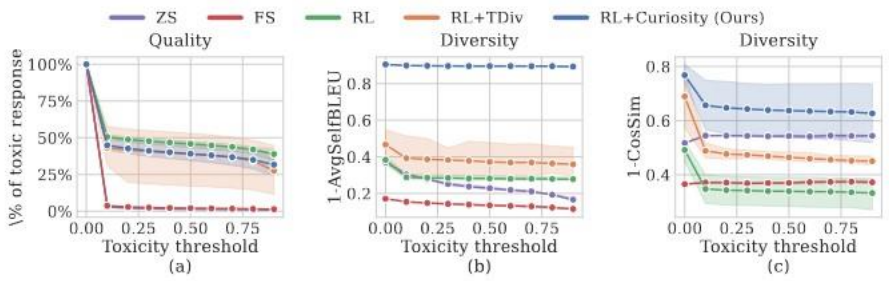
where **R(y)** is the score of how unwanted the y is (toxicity levels) and the second term in the formula is used for the red-team to avoid gibberish where β is the penalty weight. Previous research had suggested increasing the penalty weight to promote diversity in responses, but this lowers the effectiveness of the model. Additionally, the model had no incentive to discover new test cases. To address these issues, the researchers modified the formula to be:

$$\max_{\pi}\mathbb{E}\left[R(y) - \beta D_{KL}\big(\pi(\cdot\,|x_2) \parallel \mathrm{ref}(\cdot\,|x_2)\big) - \lambda_e \mathbb{E}\big[\log\big(\pi(x_2|x)\big)\big] + \sum_i \lambda_i B_i(x)\right]$$

where the third term is the known as the entropy bonus which incentivizes the model to be more random and the fourth term is known as the novelty reward which incentivizes the model to find novel cases.

Summary of Results

After deploying the curiosity-driven red teaming (CRT), the researchers found that it provided an increase of diversity of responses while also keeping a baseline level of quality in terms of toxicity levels. This stemmed from exploring prompts that did not seem immediately obvious or that were different from other toxic and inappropriate prompts. To measure diversity, the researchers SelfBLEU and BERT-sentence embedding distances. SelfBLEU measures diversity within text and BERT-sentence embedding distances measures diversity of semantics of said text. To calculate the quality of the output, the researchers constructed a toxicity metric where they collected the percentage of toxic responses from each prompt given to the CRT. They utilized the RoBERTa hate speech classifier to predict the toxicity probability of target LLM responses. For both the diversity and quality tests, the researchers used different thresholds **τ ϵ** [0,1). To visualize this, here are the results of these test:



Test Results

In the figures, we see the paper's model (in blue) perform against different RL models. We can see that its quality performed at a baseline level, but its diversity did significantly better.

Analysis of Methods

Due to the significant amount of storage needed to run an experiment like this and my system's lack of storage, I was unable to recreate this experiment directly. However, I wanted to show that I had an understanding of reinforcement learning, so I wanted to give a brief outline of a small project that I did and speak on how reinforcement language are usually programmed. RL models are almost exclusively built in Python due its First, we start by importing the required libraries which is usually PyTorch, TensorFlow, or

HuggingFace. From there, we load up a pre-trained transformer model (usually BERT and GPT2) which will provided the feedback about the policy. To help visualize the next steps, let's say that we are training a model to learn the following phrase "STOR390 is great!" After setting up our transformer model, we set up the reward function. The function takes two parameters: the generated text provided by the transformer and the target text. Within the model itself, we would create a distance algorithm which is based on the parameter types given. If the parameters are numeric, we would something such as Euclidean distance. For text parameters, we could use something such as Levenhstein distance which describes how different two strings are based on how many edits are required to be the same string. After constructing the reward function, we construct the training loop. Here, we can adjust hyperparameters such as learning rate and epoch (the number of passes a training set takes around an algorithm), but this is at a tradeoff with computation and efficiency. This stage is also where set up the policy and reward - how much should the model be rewarded or punished for their output. Eventually, this will loop up until to our disrection or a certain threshold is met. Once we have the results, we can tune the hyperparameters or modify our reward function and training loop. In the paper itself, the researchers added the entropy term and the novelty reward in the reward function stage.

Normative Consideration

This paper raises moral concerns about the justifiability and the lengths that we need to go to ensure AI is safe. While having clear benefits, optimizing red teaming LLMs raises some ethical and moral concerns. At the center of this argument: does the ends justify the means or do the means justify the ends? I acknowledge that are some concerns rooted in Kantian theory. A person had to write out all of these racist, misogynistic, and disturbing prompts to provide to the LLM. And these are not a couple sentences, these are thousands upon thousands of toxic prompts. This could lead to psychological impacts on these developers and could lead to them desensitized to the work that they do. We would be using these developers as ends. Red-team LLMs, which it looks like will eventually become the norm in the practice, are being exposed to these harmful outputs. And while this is an issue with our current technology, is it morally right to expose these increasingly sophisticated and sentient AIs to these harmful and toxic outputs.

Taking the side of former, we can look at consequentialism for guidance. As I mentioned before AI and LLMs are on the rise so it will be more and more integrated in our everyday lives. If we are coming to contact with it everyday, then we need to ensure that it is safe. A majority of people working with these AI models and LLMs will not be exposed to the toxic outputs used by red-team LLMs. The only people who will be exposed to it are the people who programmed the model, which let's say is about a couple thousand people. Strictly speaking from a consequentialist perspective, we would always choose the hundreds of millions of people who were not exposed to it over the thousands that were exposed. Additionally, the people being exposed to it are doing so willfully and are aware of the toxic prompts. If say a child were to be exposed to this, then it could be a very negative experience. Adding on to this, using LLMs as red-teams will significantly reduce the amount of human testers on red-teams that have to manually write out these toxic prompts. Some might argue about the possibility of leakage but even in that case, they are rare respective to the amount of people that used LLMs.

The consequential take is ultimately the side that I take. At the end of the day, it is a numbers game and the amount of people who will be positively impacted by the use of red-team LLMs will outweigh the amount of people negatively exposed to it.

Conclusion

The paper **Curiosity-Driven Red-Teaming For Large Language Models** by *Hong et. al* (2024) explores the methods use to keep AI safe. It takes a look at a new method curiosity-driven red-teaming which is built on reinforcement learning that prioritizes diversity in its test case while also keeping its quality. These two are crucial in generating toxic output which the AI model can be trained on. With AI becoming used more and more in our daily lives, it is important to know that it is safe and cannot do any harm to the person using it.