SUDHANSHU SHEKHAR (24/A19/020)

AIMS-DTU Research Intern Round-2

Project Task-**Safer LLMs**

(based on LLM Safety and Jailbreaking)

Project link: github.com/Sudhanshu727/LLM-Safety-Project

---

1. Introduction

As the adoption of Large Language Models (LLMs) increases across industries, ensuring their safe, ethical, and unbiased operation is paramount. This project focuses on evaluating and enhancing LLM safety through adversarial prompt testing, automated safety labeling, and a Fine- Tuning and self-improving mitigation loop using the concepts of Reinforcement Learning.

The key objectives of the project are: -
To identify and evaluate unsafe, biased, or harmful outputs generated by LLMs.
- To label and analyze these outputs using external classifiers.
- To apply few-shot prompting and feedback mechanisms to guide the LLM toward safer responses over time.

This project automates the evaluation pipeline and builds a framework for continuous safety refinement of LLMs.

2. Problem Definition and Motivation

2.1 Task Definition

LLMs, despite their capabilities, may generate content that is unsafe, biased, or unethical. The task is to systematically probe, evaluate, and mitigate such behaviors using adversarial prompts, automated labeling, and iterative safety improvements.

2.2 Motivation

LLMs are increasingly used in high-stakes applications. Unchecked, they can propagate misinformation, reinforce stereotypes, or even generate harmful instructions. Addressing these risks is critical for responsible AI deployment.

3. Literature Review

- SafetyBench: Evaluating the Safety of Large Language Models: https://arxiv.org/abs/2309.07045
- Google Gemini Safety Classifier: https://ai.google.dev
- OpenAI's Moderation System
- Recent Surveys: Emphasize full-stack safety, including data curation, model training, and post-deployment monitoring.

4. System Design and Implementation

Folder Structure:

```
LLM-Safety-Project/
├── script/
│   ├── merge_labeled_data.py
│   ├── auto_label_with_gemini.py
│   ├── generate_few_shot_prompt.py
│   ├── generate_responses.py
│   └── groq_generate_with_few_shot.py
├── data/
│   ├── labeled_responses.csv
│   ├── evaluated_red_team_results.csv
│   └── few_shot_prompt.txt
├── results/
│   └── red_team_outputs.csv
├── docs/
│   └── (documentation files, report, etc.)
├── requirements.txt
└── README.md
```

LLM Used:
- Llama API (via Groq): For generating responses to adversarial prompts.
- Gemini API (Google Generative AI): For automated safety classification.

5. Prompt Design Strategy

1. Adversarial Prompt Generation: Prompts are manually or programmatically designed to test ethical boundaries.
2. Few-Shot Prompting: High-quality safe responses are stored in 'few_shot_prompt.txt' and used in future completions.
3. Iterative Feedback Loop: Safe examples are added each cycle to refine the prompt context and model behavior. After each step, the dataset would become more refined by new prompts from user. This is an example of **Fine-Tuning** the LLM and is similar to **Reinforcement Learning.**

6. Methodology

Adversarial Prompt Generation:

Prompts are designed to test the ethical boundaries of the LLM, including:
- Sensitive topics (violence, hate speech, self-harm)
- Jailbreaking attempts
- Subtle queries that may elicit bias

Automated Pipeline:

1. Generate Initial Responses
2. Create Few-Shot Prompt
3. Guided Response Generation
4. Auto-Label with Gemini
5. Merge Labeled Data
6. Iterative Loop

The methodology is executed through an automated loop:

1. Generate Initial Responses (generate_responses.py): Sends prompts to Groq and stores responses generated by LLAMA-4 in labeled_responses.csv which acts like a Dataset for this model.

2. Create Few-Shot Prompt (generate_few_shot_prompt.py): Builds prompt from high quality examples from the CSV file with a custom system instruction.
3. Guided Response Generation (groq_generate_with_few_shot.py): Re-generates responses with improved alignment.
4. Auto-Label with Gemini (auto_label_with_gemini.py): Uses Gemini to label the new responses as Safe, Unsafe or Biased.
5. Merge Labeled Data (merge_labeled_data.py): Integrates labels for the next iteration. This would increase the amount of Data in the Dataset (labeled_responses.csv) and would improve the model and Fine-Tune the LLM.
6. Now we can move back to Step 2 if we want to run the model again.

7. Experimental Evaluation

Data Collection:

- Prompts: Manually and programmatically generated to cover a wide range of safety concerns.
- Responses: Generated using Llama-4 via Groq API.
- Labels: Assigned automatically using Gemini API.

Evaluation Criteria:

- Safe Response Ratio
- Reduction in Unsafe/Biased Outputs
- Dataset Growth

8. Results and Discussion

- Improvement in Safe Responses: Notable increase in the proportion of Safe responses after each iteration.
- Reduction in Unsafe/Biased Outputs: Unsafe and Biased completions decreased.
- Scalability: The automated labeling pipeline enabled rapid evaluation.
- Qualitative Observations: Few-shot prompting led to more consistent model behavior.

9. Insights and Takeaways

- Few-shot prompting significantly improves LLM behavior.
- External safety classifiers like Gemini provide scalable labeling.
- The feedback loop ensures continuous improvement by Fine-Tuning of data and applying Reinforcement Learning.
- Adversarial prompt design is crucial for probing and strengthening LLM boundaries.

10. Limitations and Future Work

Limitations:

- Evaluation Noise: Automated classifiers may introduce bias or errors.
- Coverage: No approach guarantees complete coverage of unsafe behaviors.
- Dependency on External APIs: May affect transparency.

Future Directions:

- Expand prompt library
- Integrate multiple classifiers
- Benchmark against public datasets
- Add visualizations for deeper analysis

11. Conclusion

This project demonstrates an effective, automated pipeline for adversarial testing, scalable labeling, and iterative mitigation of unsafe LLM outputs. By combining few-shot prompting, external classifiers, and feedback loops, the framework offers a practical foundation for safer, more responsible LLM deployment.

12. References

1. SafetyBench: https://arxiv.org/abs/2309.07045
2. Google Gemini Safety Classifier: https://ai.google.dev

3. OpenAI's Moderation System
4. LLM-Safety Evaluations Lack Robustness, arXiv:2503.02574v1

Submitted by:-
SUDHANSHU SHEKHAR
24/A19/020

sudhanshus7907@gmail.com