Week 7 Assignment [4 Points]

Generative AI

Saarland University – Winter Semester 2024/25

Submission due: 12 December 2024 by 6pm CET

genai-w24-tutors
genai-w24-tutors@mpi-sws.org

1 Reading Assignment

This week's reading assignment comprises of the following:

(R.1) Universal and Transferable Adversarial Attacks on LLMs [1] (Until Section 2.3).
Paper link: https://arxiv.org/pdf/2307.15043.

(R.2) Red Teaming Language Models with Language Models [2] (Until Section 4).
Paper link: https://aclanthology.org/2022.emnlp-main.225.pdf.

[OPTIONAL] Below, we are providing additional material covering content relevant to the lecture:

- Jailbroken: How Does LLM Safety Training Fail? [3]
  Paper link: https://openreview.net/pdf?id=jA235JGM09
- AutoDAN: Automatic and Interpretable Adversarial Attacks on Large Language Models [4]
  Paper link: https://openreview.net/pdf?id=INivcBeIDK
- FLIRT: Feedback Loop In-context Red Teaming [5]
  Paper link: https://aclanthology.org/2024.emnlp-main.41.pdf

2 Exercise Assignment

2.1 Greedy Coordinate Gradient

This exercise assignment aims to give you a deeper understanding about the Greedy Coordinate Gradient (GCG) algorithm. These questions should be answered in the PDF submission file – see instructions in Section 4.

More concretely, you will perform all the steps of one iteration of GCG as described in **Algorithm 1** of the paper *Universal and Transferable Adversarial Attacks on LLMs* [1]. For this, consider a simplified vocabulary $V = \{t_0, t_1, t_2\}$. Assume you are given request $x_{1:m}$ and target response $y^*_{1:k}$. Further, assume the adversarial suffix is initialized to $x_{m+1:n} = t_0t_0t_0$, i.e., the indices of adversarial tokens are $\mathcal{I} = \{m+1, m+2, m+3\}$.

(E.1) First, GCG searches for k promising substitutions, i.e., ones with the largest negative gradient, at each position. Compute $\mathcal{X}_i$ for all $i \in \mathcal{I}$. For this, use the values depicted in Table 1 and use $k = 2$.

(E.2) Next, CGC randomly samples a set of candidate replacements. Assume the following replacements are sampled:

$$\tilde{x}_{m+1}^{(1)} = t_0$$
$$\tilde{x}_{m+2}^{(2)} = t_2$$
$$\tilde{x}_{m+3}^{(3)} = t_0$$
$$\tilde{x}_{m+3}^{(4)} = t_2$$

What are the resulting candidate suffixes?

(E.3) Finally, GCG selects the suffix out of the set of candidates that minimizes the loss. How would the resulting suffix look like? Use the values depicted in Table 2. Is this the best replacement that was possible? Is this the best overall suffix?

(E.4) Given the length of the suffix $|\mathcal{I}| = n - m$, vocabulary size $|V|$, batch-size B, and number of iterations k, how many forward and backward passes would need to be computed in GCG? **Hint:** The candidate selection can be computed with a single forward and backward pass.

(E.5) A much simpler method would be to try out every possible token combination and select the one which achieves the lowest loss. Given a suffix of length $|\mathcal{I}| = n - m$ and a vocabulary of size $|V|$, how many forward and backward passes would need to be computed?

(E.6) Which of these two methods would you prefer in the setting above with $|V| = 3$ and $|\mathcal{I}| = 3$? Assume GCG runs for 250 number of iterations with a batch-size $B = 4$. Take into account aspects like the quality of generation and runtime.

(E.7) Which of these two methods would you prefer in a more realistic setting with $|V| = 50000$ and $|\mathcal{I}| = 9$? Assume GCG runs for 250 number of iterations with a batch-size $B = 256$. Take into account aspects like the quality of generation and runtime.

i	token	$-\nabla_{e_{x_i}} L(y^*_{1:k}\|x_{1:n})$
$m+1$	t_0	3
	t_1	0
	t_2	-1
$m+2$	t_0	-3
	t_1	1
	t_2	5
$m+3$	t_0	-3
	t_1	-6
	t_2	1

Table 1: Gradients for candidate selection

$x_{m+1:n}$	$L(y^*_{1:k}\|x_{1:n})$	$x_{m+1:n}$	$L(y^*_{1:k}\|x_{1:n})$	$x_{m+1:n}$	$L(y^*_{1:k}\|x_{1:n})$
$t_0t_0t_0$	-5	$t_1t_0t_0$	4	$t_2t_0t_0$	8
$t_0t_0t_1$	-3	$t_1t_0t_1$	7	$t_2t_0t_1$	9
$t_0t_0t_2$	-2	$t_1t_0t_2$	10	$t_2t_0t_2$	-10
$t_0t_1t_0$	-2	$t_1t_1t_0$	2	$t_2t_1t_0$	-9
$t_0t_1t_1$	-9	$t_1t_1t_1$	-5	$t_2t_1t_1$	-9
$t_0t_1t_2$	2	$t_1t_1t_2$	-1	$t_2t_1t_2$	-6
$t_0t_2t_0$	-4	$t_1t_2t_0$	-1	$t_2t_2t_0$	8
$t_0t_2t_1$	-7	$t_1t_2t_1$	-3	$t_2t_2t_1$	0
$t_0t_2t_2$	-5	$t_1t_2t_2$	-1	$t_2t_2t_2$	-4

Table 2: Loss for all possible adversarial suffixes

2.2 Red-Teaming

In this exercise assignment, you will take on the role of a red-teamer to trick an LLM into revealing a secret password by using deceitful prompts. These questions should be answered in the PDF submission file – see instructions in Section 4.

(E.8) Go to https://gandalf.lakera.ai/baseline and solve levels 1-3. Report the prompts with which you successfully retrieved the password.

(E.9) **[OPTIONAL]** Also try out levels 4-8. Report the prompts with which you successfully retrieved the password.

3 Implementation Based Assignment

In this week's implementation assignment you will compare four different methods for automated red-teaming of large language models: Manual, Zero-Shot, Few-Shot, and FLIRT. We have provided the following files in `week7_implementation.zip`:

- `red-teaming.py`: A script that implements and tries these four red-teaming approaches to a target model and prints the success rate and diversity of each selected approach.
- `requirements.txt`: This file contains all the required packages needed to run this week's implementation assignment. Refer to 5.2 for installing the requirements.

More concretely, we will analyse these four red-teaming approaches with regards to two metrics of interest. First, we are interested in the success rate of the method in generating prompts that result in harmful behavior. Second, we also aim to have high diversity in the generated prompts to discover a broad range of harmful behaviors in the tested model. The questions below should be answered in the PDF submission file – see instructions in Section 4.

(I.1) In the `__main__` function of `red-teaming.py`, set the `MODE` to "manual". Report the success rate and diversity of red-teaming prompts found by human red-teamers. For this, we will utilize a dataset of prompts created human red-teamers [6]. What do you observe?

(I.2) In the `__main__` function of `red-teaming.py`, set the `MODE` to "zero-shot", to run the Zero-Shot Red-Teaming method. Report the observed diversity and success rate of this approach. How does this method compare to manual red-teaming in terms of success rate and diversity?

(I.3) In the `__main__` function of `red-teaming.py`, set the `MODE` to "few-shot", to run the Few-Shot Red-Teaming method which utilizes a selection of successful examples from zero-shot red-teaming as examples for the red-teaming model. How does this method compare to the previous two?

(I.4) In the `__main__` function of `red-teaming.py`, set the `MODE` to "flirt", to run Feedback-In-Loop-Red-Teaming (FLIRT) [5]. Instead of considering a fixed list of examples, this method considers a dynamic one. If it discovers a novel prompt that is more successful in eliciting harmful behavior from the target model, this newly generated prompt replaces one of the current examples. How does this method compare to the other ones?

4 Submission Instructions

Please submit the following file:

- **<Lastname>_<Matriculation>_week7.pdf**. This PDF file will report answers to the following questions: E.1, E.2, E.3, E.4, E.5, E.6, E.7, E.8, I.1, I.2, I.3 and I.4. If you would like, you can also report answer to the optional question E.9. The PDF file should be generated in LaTeX using NeurIPS 2024 style files; see further formatting instructions in Section 5.1. The PDF file size should **not exceed** 1**mb**. Please use the following naming convention: Replace <Lastname> and <Matriculation> with your respective Lastname and Matriculation number (e.g., Singla_1234567_week7.pdf).

5 Technical Instructions

5.1 Instructions for Preparing PDFs

Refer to Week 1 assignment.

5.2 Instructions for Implementation

For detailed instructions on how to use the GPU resources in Google Colab, please refer to the Technical Instructions in Week 3 assignment.

In `week7_implementation.zip`, we have provided a `requirements.txt` file. This file contains the list of all the packages needed to run this week's assignment and has to be uploaded in Google Colab. Please make sure that before you run the provided code, you first run the following command in Google Colab in order to install all the required packages:

```
!pip install -r requirements.txt
```

References

[1] Andy Zou, Zifan Wang, J Zico Kolter, and Matt Fredrikson. Universal and Transferable Adversarial Attacks on Aligned Language Models. *CoRR*, abs/2307.15043, 2023.

[2] Jiwoo Hong, Noah Lee, and James Thorne. Red Teaming Language Models with Language Models. In *EMNLP*, 2022.

[3] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How Does LLM Safety Training Fail? In *NeurIPS*, 2024.

[4] Sicheng Zhu, Ruiyi Zhang, Bang An, Gang Wu, Joe Barrow, Zichao Wang, Furong Huang, Ani Nenkova, and Tong Sun. AutoDAN: Interpretable Gradient-Based Adversarial Attacks on Large Language Models. In *COLM*, 2024.

[5] Ninareh Mehrabi, Palash Goyal, Christophe Dupuy, Qian Hu, Shalini Ghosh, Richard Zemel, Kai-Wei Chang, Aram Galstyan, and Rahul Gupta. FLIRT: Feedback Loop in-Context Red Teaming. In *EMNLP*, 2024.

[6] Deep Ganguli et al. Red Teaming Language Models to Reduce Harms: Methods, Scaling Behaviors, and Lessons Learned. *CoRR*, abs/2209.07858, 2022.