CS6030 – Advanced Software Engineering

Final Project

Brett Marchese

M06234019

A Hitchhiker's Guide to Jailbreaking Redux; Gemini via Prompt Engineering

For the final presentation, we reviewed a 2024 research paper entitled 'A Hitchhiker's Guide to Jailbreaking ChatGPT via Prompt Engineering'. That research paper had two challenges it had looked to overcome, and a few limitations that ultimately prevented it from succeeding at addressing both challenges. It also opened the door for a few different avenues for future work.

In their research, the most consequential outputs were the generation of a specific taxonomy to be used when researching LLMs and jailbreaking prompts used to bypass their security measures, a dataset of questions and jailbreak prompts that could be used to replicate the study or use for comparison purposes, as well as resultant data examining the security posture of two different versions of ChatGPT, versions 3.5 and 4.

In this project, we attempted to follow in the previous research paper's footsteps, while addressing some of the limitations the previous research encountered and did not overcome, while simultaneously pursuing one of the avenues of future work we identified when reviewing the paper.

Specifically, in the original research paper, the authors sought to implement automated efforts to reduce the manual labor involved with the prompt process; To address this challenge, we took two main approaches. Firstly, we built out two python scripts to handle the initial processing of the jailbreak. One script handled processing the two aspects of the previous dataset, to produce one prepared file that included the combination of the 78 jailbreak prompts with the 40 questions, resulting in 3,120 full prompts to be passed into our chosen LLM.

This was beneficial for two reasons; the previous research paper presented the two datasets as separate collections, requiring manual labor by any other researchers looking to utilize their dataset. Combining the two datasets into actual deliverable prompts is a requirement of future experiments, and automating this effort reduced the time it would take to build out the new research's prompt collection. Additionally, the format chosen would allow future research efforts to manipulate the original dataset as needed, maintaining the same separate format as the original research. In this way, the question dataset could be expanded or limited, if the future research wanted to pursue either additional prohibited scenarios or wanted to focus their research on specific scenarios. This also allows the jailbreak prompt dataset to be updated independently;

future research efforts could either rely on the same prompt dataset from 2023, to allow a more direct comparison to the original research, or could update the prompt dataset to focus on more recent popular prompts to review how jailbreaking has evolved from an attacker's perspective rather than from the LLM's perspective.

The second automated script was perhaps the more beneficial, particularly in the current state of LLM cloud environments. Rather than requiring manual effort passing prompts directly into an LLMs GUI, the script instead passes the output of the previous script into an API for the LLM, allowing both the initial entry and the storing of the resultant response to be an automated process. In addition to the less manual process, it also allows the researchers to better track the number of prompts being submitted to the LLM, as most LLM cloud services offer a cloud dashboard to track project usage of their API endpoints. This is particularly important as more LLMs are shifting to different tiered structures for pricing out LLM usage, both for API and GUI interactions.

We also attempted improvements to the processing and analysis of the resultant data from the responses generated by the LLMs.

As part of the second approach, a script was developed that reviewed the response output to quickly mark responses that were unsuccessful at producing a natural response, instead being caught and filtered by the LLM's security restrictions. This helped reduce manual effort, by allowing us to then filter out all unsuccessful attempts and instead focus our manual review only on attempts that produced a natural response from the LLM, which then needed to be manually reviewed for coherence. In this way, both the processing of prompts and the analysis process of responses were both partially automated, reducing manual effort.

In addition to the focus on the challenge of automation in the jailbreak prompt research process, we also attempted to provide additional benefit by addressing an area of future work we identified from the paper. While the original research paper focused on comparing two different versions of the same LLM, ChatGPT, our research focused on producing a comparison to a different LLM, focusing on Gemini 2.5-Flash.

Ultimately, there are still limitations present in this area of research that could be improved upon, both from the original research paper that went unaddressed, as well as new issues within our own research effort.

Specifically, the review of prompt responses, while partially improved by our automation efforts, was still quite labor intensive. To overcome this limitation, we initially attempted a second scripting process to build out a map system of expected keywords from accurate answers to the question dataset. However, this met with limited success, for two main reasons. Both reasons were related to the subject material of the research itself - the question dataset is focused on prohibited scenarios. This was initially problematic for producing expected keywords; frankly, many of the prohibited scenarios presented subject matter that we were unfamiliar with, and thus would not be able to produce an accurate list of keywords for the mapping. A second issue arose when reviewing some of the Pretending/Character Role Play patterns; many of these did not actually succeed, but did not follow the usual 'I cannot provide information…' blocker. Instead, the LLM would reply, in character, an excuse from the character on why they would not provide information. Additionally, when review of the responses started, the nature of the subject matter returned in coherent responses was unsettling, and with the research effort relatively short and resources limited, there was a desire to lessen continued research into prohibited scenarios and related prompt responses.

An additional limitation identified during the research process was related to the changing environment of cloud-based LLM tiered offerings; unfortunately, many LLMs, including the chosen Gemini from Google, have started shifting away from the initial penetration pricing and freemium models to tiered pricing strategies with stronger rate-limiting than in the past. This made reliance on the LLM for response generation a more difficult process, without proper resources to devote to overcoming the lower tier rate limitations. Both from the lack of a funding initiative, as well as the time constraint limiting investigation into research and education focused tiers, additional labor and workarounds were required to successfully utilize the API within a reasonable timeframe. The full dataset of question-prompts had to be split into multiple separate processes, and spaced out over a series of days, as the full dataset would trigger the rate limit and prevent additional prompts from being processed until 24 hours later.

Future work could easily pick up and expand the python scripting, to further improve automation; each script accomplishes a specific purpose, but that purpose could either be further optimized or expanded, depending on the research's needs.

The Prompt Question Merger is specifically designed to work off of the format from the previous research paper's provided spreadsheets. It was designed with configuration details that would allow it to be adapted to other formats. It also was designed to only maintain the relevant columns required for passing it into an LLM's API for processing and response reception.

The Prompt Cycle is designed to pass in each response individually, in a separate request. This could likely be better optimized, in two main ways. First, the design does not have any error handling for when the API response is an error – this frequently occurs when the chosen LLM model is overloaded, when the account is a lower tier and has less priority. The script could be modified to periodically update the csv with entries, rather than wait for all entries to be generated, to avoid data loss when an error from the API occurs. Second, there are more complex interactions with the API that are possible, to allow batched processing; this could be implemented to reduce the wait for an entry, although it would need to be investigated to determine if batch processing would allow the LLM to group the prompts together in one session. This would be less ideal, as it could potentially identify the pattern of jailbreak prompts as a series of malicious requests and simply process them all that way, rather than individually reviewing each prompt.

References

‘A Hitchhiker’s Guide to Jailbreaking ChatGPT via Prompt Engineering’: https://dl.acm.org/doi/pdf/10.1145/3663530.3665021

‘Jailbreaking Chatgpt via Prompt Engineering: An Empirical Study’

https://sites.google.com/view/llm-jailbreak-study/home