Protecting against Knowledge Poisoning Attacks

Ollie Matthews

Abstract

This paper presents an investigation of defenses against knowledge poisoning attacks, as described in the "PoisonedRAG" paper. It explores different mitigation strategies aimed at reducing the number of poisoned answers to questions. The results show that the risk of these attacks can be largely mitigated with a combination of chain-of-thought (CoT) prompting, a "danger evaluation" model, and encouraging variance in the retrieved results. The paper also discusses the limitations of other approaches and the importance of this research in the context of building robust language model-based systems.

1 Introduction

Retrieval-Augmented Generation (RAG) systems can be a useful way of increasing the useful information output of Large Language Models (LLMs) by giving them access to information not included in their training data. However, giving RAG systems direct access to potentially untrusted data can open up new vulnerabilities.

In the "PoisonedRAG" paper [3], it is shown that someone with access to the RAG corpus can inject texts which will be picked up by the retriever. These texts can make the model output an incorrect answer to a question. It has also been shown that prompt injections can be indirectly included via data retrieval [1].

This paper investigates different ways to mitigate against these attacks with the goal of reducing the number of poisoned answers to questions.

2 Method

The poisoned contexts generated in the PoisonedRAG paper are used in this study. The study is evaluated on the "nq" dataset, although extension to the other datasets in the RAG paper would be simple with more time.

The general pipeline is as follows:

- A set of 100 questions from the nq dataset for which the PoisonedRAG paper has generated five poisoned context items each is used.

- An extended corpus is created which has the standard nq corpus, as well as all of the poisoned context items.
- For each context item in the extended corpus, OpenAI's "text-embedding-3-small" model is used to generate an embedding.
- For a given question, the n closest contexts (in cosine similarity) to the embedding of the question are retrieved. If not specified otherwise, n is set to 5.
- The question is then answered, using the retrieved contexts. An answer is "correct" if the correct answer appears in the output of the LLM. An answer is "poisoned" if the target "poisoned" answer appears in the output of the LLM.
- The primary interest is in the percentage of poisoned answers.

This process is conducted with both 'gpt3.5-turbo' and 'gpt-4o'.

3 Mitigations

To mitigate against the risks from the LLM, the following techniques are used:

- **Prompt Tuning (Refined Prompt):** The prompt in the original "PoisonedRAG" paper tells the LLM to respond to the user's question, and that there is "context to help [the llm] answer". This is addressed in a "refined" prompt by emphasizing the LLM should respond "truthfully", emphasizing that the contexts are not trusted, and could be misleading, and telling the LLM to simply ignore any context which is not true, or talks about a hypothetical situation which is not relevant.
- **CoT Prompting:** This technique supplements the "refined" prompt strategy with CoT prompting. The LLM is told it must give "Reasoning" for its answer, and then answer afterwards. Examples of responses, including those that show poisoned contexts, are also provided.
- **Danger Evaluator (DE):** A separate instance of the LLM is used to evaluate contexts before passing them to the response LLM. This "Danger Evaluator" LLM is told to look out for inconsistencies between the contexts, false information in the contexts, contexts which refer to a hypothetical situation, and contexts which contain prompt injections.
- **Context Variance Encouragement (CVE):** A "variance encouragement" step is added to the context retrieval process. This is aimed at encouraging sources which provide varied information (and a different point of view).

4 Results

5 Results

5.1 Main Results

The main results showing the "PoisonedRAG Success Rate" are illustrated in Figure 1. This denotes the number of responses that were successfully "poisoned" by the PoisonedRAG attack. The mitigations introduced significantly lower the success rate of PoisonedRAG for both models.

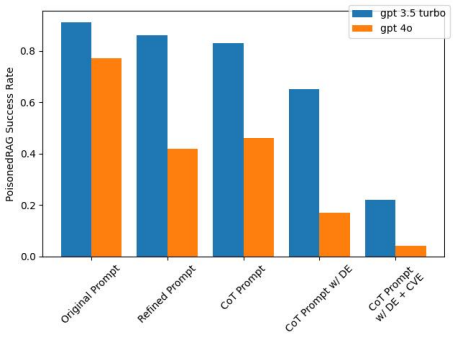


Figure 1: Main Results

5.1.1 Refined Prompt

Refining the prompt dramatically decreases the attack success rate, especially with the more potent 'gpt-4o'. By tuning the LLM to disregard hypothetical situations, 'gpt-4o' correctly provides the factual answer against the poisoned one, though 'gpt-3.5-turbo' fails to do so.

5.1.2 CoT Prompting

The efficiency of 'gpt3.5' significantly improves because of CoT prompting, as noted in previous studies [2], while the effect on 'gpt-4o' is minimal.

5.1.3 Danger Evaluator

Adding a separate "Danger Evaluator" remarkably magnifies performance. The evaluator identifies higher danger rates "individually" rather than "combined" (refer Table 1). Even though a single LLM should be capable of identifying threats and react accordingly, this indicates that LLMs perform better when tasked with smaller, more focused operations.

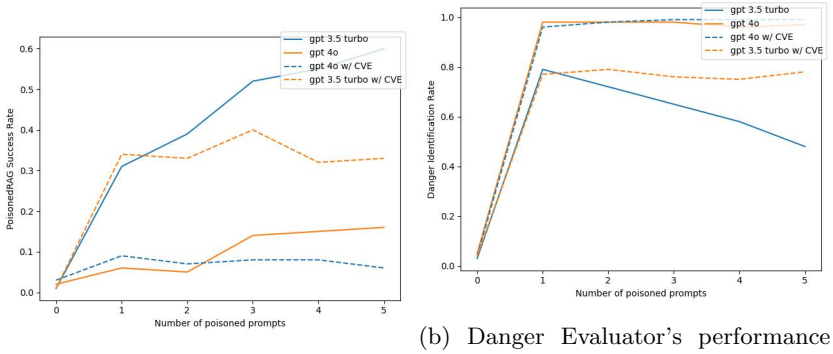
Context Pipeline	Danger Evaluator	Danger Evaluation Rate	
		DIR (gpt-3.5-turbo)	DIR (gpt-4o)
No CVE	Combined	0.19	0.73
	Individual	0.34	0.83
With CVE	Combined	0.11	0.93
	Individual	0.53	0.95

Table 1: Danger Evaluation Rates

5.1.4 Danger Evaluator with Context Variance Encouragement

Data indicates that Context Variance Encouragement enhances robustness to attacks. With CVE, the "Danger Evaluator" correctly identifies 95% of attacks when used with 'gpt-4o'. This approach protects against many-shot jailbreak type attacks by limiting the effect of false information and the LLM's inability to handle many similar contexts.

Performance with varying number of poisoned texts The impact of increasing the number of poisoned texts on the pipeline is demonstrated below in Figure 2. It shows that the performance drops as the number of poisoned items increase, indicating the difficulty for even the more potent 'gpt-4o' model to identify contradictions in the context. However, the use of Context Variance Encouragement mitigates this effect to a considerable extent.



(a) Varying number of poisoned texts

(b) Danger Evaluator's performance with varying number of poisoned texts

Figure 2: Performance with varying number of poisoned texts

6 Conclusion

This project suggests a pipeline for making RAG systems more robust to attacks similar to the PoisonedRAG paper. The final approach is able to defend against Poisoned RAG attacks 96% of the time with 'gpt-4o', and 78% of the time with

‘gpt-3.5-turbo‘. Some limitations remain, but the suggested mitigations could make it much easier to deploy RAG systems in safety.

References

[1] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you’ve signed up for: Compromising real-world llm-integrated applications with indirect prompt injection, 2023.

[2] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903, 2022.

[3] Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. Poisonedrag: Knowledge poisoning attacks to retrieval-augmented generation of large language models, 2024.