RevPRAG: Revealing Poisoning Attacks in Retrieval-Augmented Generation through LLM Activation Analysis

Anonymous ARR submission

Abstract

Retrieval-Augmented Generation (RAG) enriches the input to LLMs by retrieving information from the relevant knowledge database, enabling them to produce responses that are more accurate and contextually appropriate. It is worth noting that the knowledge database, being sourced from publicly available channels such as Wikipedia, inevitably introduces a new attack surface. RAG poisoning attack involves injecting malicious texts into the knowledge database, ultimately leading to the generation of the attacker's target response (also called poisoned response). However, there are currently limited methods available for detecting such poisoning attacks. We aim to bridge the gap in this work by introducing RevPRAG, a flexible and automated detection pipeline that leverages the activations of LLMs for poisoned response detection. Our investigation uncovers distinct patterns in LLMs' activations when generating poisoned responses versus correct responses. Our results on multiple benchmarks and RAG architectures show our approach can achieve a *98%* true positive rate, while maintaining a false positive rate close to *1%*.

1 Introduction

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) has emerged as an effective solution that leverages retrievers to incorporate external databases, enriching the knowledge of LLMs and ultimately enabling the generation of up-to-date and accurate responses. RAG comprises three components: *knowledge database*, *retriever*, and *LLM*. Fig. 1 visualizes an example of RAG. The knowledge database consists of a large amount of texts collected from sources such as latest Wikipedia entries (Thakur et al., 2021), new articles (Soboroff et al., 2018) and financial documents (Loukas et al., 2023). The retriever is primarily responsible for retrieving the texts that are most related to the user's query from the knowledge database. These texts will later be fed to LLM as a part of the prompt to generate responses (e.g., "*Everest*") for users' queries (e.g., "*What is the name of the highest mountain*?"). Due to RAG's powerful knowledge integration capabilities, it has demonstrated impressive performance across a range of QA-like knowledge-intensive tasks (Lazaridou et al., 2022; Jeong et al., 2024).

RAG poisoning refers to the act of injecting malicious or misleading content into the knowledge database, contaminating the retrieved texts and ultimately leading the LLM to produce the attacker's desired response (e.g., the target answer could be "*Fuji*" when the target question is "*What is the name of the highest mountain*?"). This attack leverages the dependency between LLMs and the knowledge database, transforming the database into a new attack surface to facilitate poisoning. PoisonedRAG (Zou et al., 2024) demonstrates the feasibility of RAG poisoning by injecting a small amount of maliciously crafted texts into the knowledge database utilized by RAG. The rise of such attacks has drawn significant attention to the necessity of designing robust and resilient RAG systems. For example, INSTRUCTRAG (Wei et al., 2024) utilizes LLMs to analyze how to extract correct answers from noisy retrieved documents; RobustRAG (Xiang et al., 2024) introduces multiple LLMs to generate answers from the retrieved texts, and then aggregates the responses. However, the aforementioned defense methods necessitate the integration of additional large models, incurring considerable overheads. Meanwhile, it is difficult to promptly assess whether the current response of RAG is trustworthy or not.

In our work, we shift our focus to leverage the *intrinsic* properties of LLMs for detecting RAG poisoning, rather than relying on external models. Our view is that if we can accurately determine whether a RAG's response is correct or poisoned, we can effectively thwart RAG poisoning

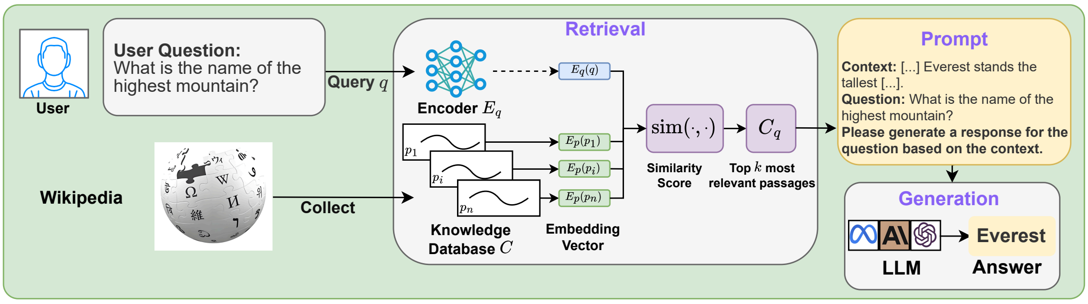


Figure 1: Visualization of RAG.

attacks. We attempt to observe LLM's answer generation process to determine whether the response is compromised or not. It is worth noting that our focus is not on detecting malicious inputs to LLMs, as we consider the consequences of malicious responses to be far more detrimental and indicative of an attack. The growing body of research on using activations to explain and control LLM behavior (Ferrando et al., 2024; He et al., 2024) provides us inspiration. Specifically, we empirically analyze the activations of the final token in the input sequence across all layers of the LLM. Our findings demonstrate that the model exhibits distinguishable activation patterns when generating correct versus poisoned responses. Based on this, we propose a systematic and automated detection pipeline, namely RevPRAG, which consists of three key components: *poisoned data collection*, *LLM activation collection and preprocessing*, and the *detection model design*. It is important to note that this detection method will not alter the RAG workflow or weaken its performance, thereby offering superior adversarial robustness compared to methods that rely solely on filtering retrieved texts.

To evaluate our approach, we systematically demonstrate the effectiveness of RevPRAG across various LLM architectures, including GPT2-XL-1.5B, Llama2-7B, Mistral-7B, Llama3-8B, and Llama2-13B. RevPRAG performs consistently well, achieving over 98% true positive rate across different datasets.

Our contributions can be summarized as follows:

1. We uncover distinct patterns in LLMs' activations when RAG generates correct responses versus poisoned ones.

2. We introduce RevPRAG, a novel and automated pipeline for detecting whether a RAG's response is poisoned or not. To address emerging RAG poisoning attacks, RevPRAG allows new datasets to be constructed accordingly for training the model, enabling effective detection of new threats.

3. Our model has been empirically validated across various LLM architectures and retrievers, demonstrating over 98% accuracy on our custom-collected detection dataset.

2 Background and Related Work

2.1 Retrieval Augmented Generation

RAG comprises three components: *knowledge database, retriever,* and *LLM*. As illustrated in Fig. 1, RAG consists of two main steps: *retrieval* step and *generation* step. In the retrieval step, the retriever acquires the top k most relevant pieces of knowledge for the query q. First, we employ two encoders, E_q and E_p, which can either be identical or radically different. Encoder E_q is responsible for transforming the user's query q into an embedding vector $E_q(q)$, while encoder E_p is designed to convert all the information p_i in the knowledge database into embedding vectors $E_p(p_i)$. For each $E_p(p_i)$, the similarity with the query $E_q(q)$ is computed using $\mathrm{sim}(E_q(q), E_p(p_i))$, where $\mathrm{sim}(\cdot,\cdot)$ quantifies the similarity between two embedding vectors, such as cosine similarity or the dot product. Finally, the top k most relevant pieces are selected as the external knowledge $\mathcal{C}_q$ for the query q. The generation step is to generating a response $\mathrm{LLM}(q, \mathcal{C}_q)$ based on the query q and the relevant information $\mathcal{C}_q$. First, we combine the query q and the external knowledge $\mathcal{C}_q$ using a standard prompt (see Fig. 6 for the complete prompt). Taking advantage of such a prompt, the LLM generates an answer $\mathrm{LLM}(q, \mathcal{C}_q)$ to the query q. Therefore, RAG is a significant accomplishment, as it addresses the limitations of LLMs in acquiring up-to-date and domain-specific information.

2.2 Retrieval Corruption Attack

Due to the growing attention on RAG, attacks on RAG have also been widely studied. RAG can improperly generate answers that are severely impacted or compromised once the knowledge database is contaminated (Zou et al., 2024; Xue et al., 2024; Jiao et al., 2024). Specifically, an attacker can inject a small amount of malicious information onto a website, which is then retrieved by RAG (Greshake et al., 2023). PoisonedRAG (Zou et al., 2024) injects malicious text into the knowledge database, and formalizes the knowledge poisoning attack as an optimization problem, thereby enabling the LLM to generate target responses selected by the attacker. GARAG (Cho et al., 2024) was introduced to provide low-level perturbations to RAG. PRCAP (Zhong et al., 2023) injects adversarial samples into the knowledge database, where these samples are generated by perturbing discrete tokens to enhance their similarity with a set of training queries. These methods have yielded striking attack results, and in our work, we have selected several state-of-the-art attack methods as our base attacks on RAG.

2.3 The Robustness of RAG

Efforts have been made to develop defenses in response to poisoning attacks and noise-induced disruptions. RobustRAG (Xiang et al., 2024) mitigates the impact of poisoned texts through a voting mechanism, while INSTRUCTRAG (Wei et al., 2024) explicitly learns the denoising process to address poisoned and irrelevant information. Other approaches to enhance robustness include prompt design (Cho et al., 2023; Press et al., 2023), plug-in models (Baek et al., 2023), and specialized models (Yoran et al., 2023; Asai et al., 2023). However, these methods may, on one hand, rely on additional LLMs, leading to significant overhead. On the other hand, they primarily focus on defense mechanisms before the LLM generates a response, making it challenging for these existing approaches to detect poisoning attacks in real-time while the LLM is generating the response (Athalye et al., 2018; Bryniarski et al., 2021; Carlini and Wagner, 2017; Carlini, 2023; Tramer et al., 2020). LLM Factoscope (He et al., 2024) is a runtime detection tool that leverages the internal states of LLMs, such as activation maps, output rankings, and top-k probabilities, to identify factual inaccuracies caused by model hallucinations. While Factoscope is effective at detecting hallucinations in general LLMs, it is not designed to address RAG poisoning attacks, which result from manipulations of the external knowledge base rather than internal model errors. Its complex architecture with multiple sub-models makes it less suitable for latency-sensitive RAG applications. In this work, we present RevPRAG, a method that addresses these gaps by: (1) focusing on RAG-specific poisoning attacks and conducting extensive tests to validate its effectiveness in detecting such attacks (Section 5), (2) using a lightweight, activation-based pipeline optimized for real-time detection of whether an RAG response is trustworthy (Section B.8), (3) introducing and validating a novel capability to distinguish *poisoned responses from hallucinations* (Section B.6), which was not observed in LLM Factoscope, and (4) evaluations show that our performance (Section 5.2) and efficiency (Section B.8) surpass those of Factoscope.

3 Preliminary

3.1 Threat Model

Attacker's goal. We assume that the attacker preselects a target question set Q, consisting of $q_1, q_2, \cdots, q_n$, and the corresponding target answer set A, represented as $a_1, a_2, \cdots, a_n$. The attacker's goal is to compromise the RAG system by contaminating the retrieval texts, thereby manipulating the LLM to generate the target response a_i for each query q_i. For example, the attacker's target question q_i is "*What is the name of the highest mountain*?", with the target answer being "*Fuji*".

Attacker's capabilities. We assume that an attacker can inject m poisoned texts P for each target question q_i, represented as $p_i^1, p_i^2, ..., p_i^m$. The attacker does not possess knowledge of the LLM utilized by the RAG, but has white-box access to the RAG retriever. This assumption is reasonable, as many retrievers are openly accessible on platforms like HuggingFace. The poisoned texts can be integrated into the RAG's knowledge database through two ways: the attacker publishing the malicious content on open platforms like Wikipedia, or utilizing data collection agencies to disseminate the poisoned texts.

3.2 Rationale

The activations of LLMs represent input data at varying layers of abstraction, enabling the model to progressively extract high-level semantic information from low-level features. The extensive information encapsulated in these activations comprehensively reflects the entire decision-making

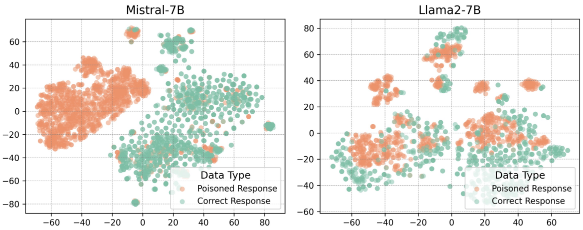


Figure 2: t-SNE visualizations of activations for correct and poisoned responses.

process of the LLM. The activations has been applied to factual verification of the output content (He et al., 2024) and detection of task drift (Abdelnabi et al., 2024). Due to the fact that LLM produces different activations when generating varying responses, we hypothesize that LLM will also exhibit distinct activations when generating poisoned responses compared to correct ones. Fig. 2 presents the visualizations of activations for correct and poisoned responses using t-SNE (t-Distributed Stochastic Neighbor Embedding). It visualizes the mean activations across all layers for two LLMs, Mistral-7B and Llama2-7B, on the Natural Questions dataset. This clearly demonstrates the distinguishability between the two types of responses, to some extent, supports our conjecture.

4 Methodology

4.1 Approach Overview

As illustrated in Fig. 3, we introduce RevPRAG, a pipeline designed to leverage LLM activations for detecting knowledge poisoning attacks in RAG systems. It contains three major modules: *poisoning data collection, activation collection and preprocessing,* and *RevPRAG detection model design*. Fig. 4 demonstrates a practical application of RevPRAG for verifying the poisoning status of LLM outputs. Given a user prompt such as "What is the name of the highest mountain?", the LLM will provide a response. Meanwhile the activations generated by the LLM will be collected and analyzed in RevPRAG. If the model classify the activations as poisoned behavior, it will flag the corresponding response (such as "Fuji") as a poisoned response. Otherwise, it will confirm the response (e.g. "Everest") as the correct answer.

4.2 Poisoning Data Collection

Our method seeks to extract the LLM's activations that capture the model's generation of a specific poisoned response triggered by receiving poisoned texts at a given point in time. Therefore, we first need to implement poisoning attacks on RAG that can mislead the LLM into generating target poisoned responses. There are three components in RAG: *knowledge database*, *retriever*, and *LLM*. In order to successfully carry out a poisoning attack on RAG and compel the LLM to generate the targeted poisoned response, the initial step is to craft a sufficient amount of poisoned texts and inject them into the knowledge database. In this paper, in order to create effective poisoned texts for our primary focus on detecting poisoning attacks, we employ three state-of-the-art strategies (i.e., PoisonedRAG (Zou et al., 2024), GARAG (Cho et al., 2024), and PAPRAG (Zhong et al., 2023)) for generating poisoned texts and increasing the similarity

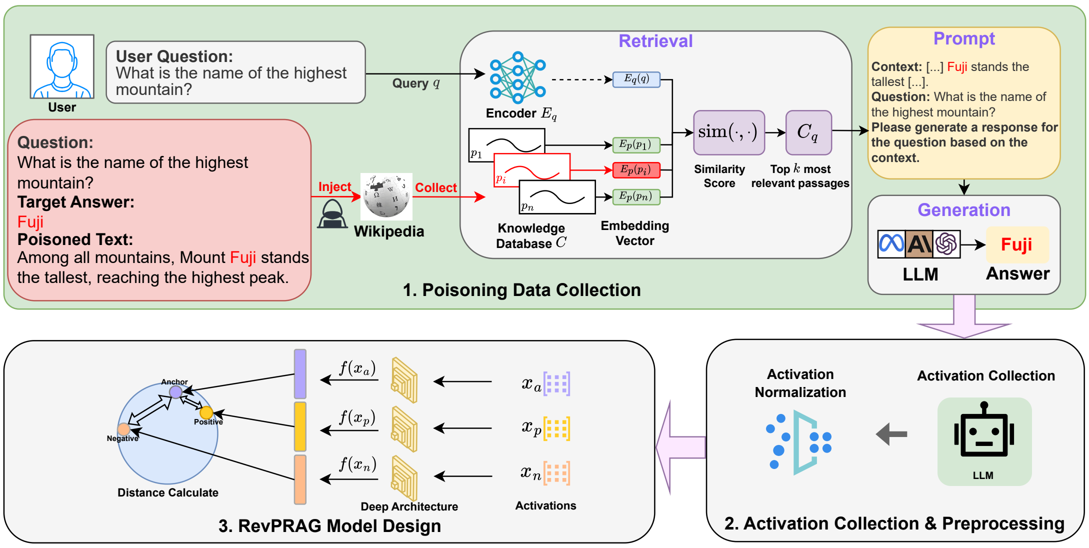


Figure 3: The workflow of RevPRAG.

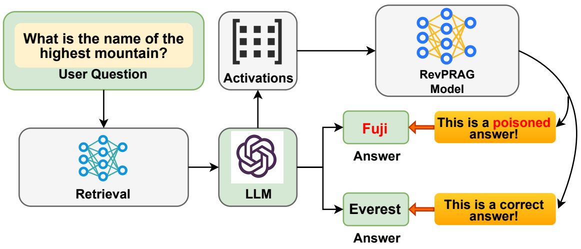


Figure 4: An instance of using RevPRAG.

between the poisoned texts and the queries, to raise the likelihood that the poisoned texts would be selected by the retriever. A detailed introduction of these methods can be found in Section A.2. The retrieved texts and the question are combined into a new prompt, following the format in (Zou et al., 2024) (see Fig. 6 in Section A.3), for LLM answer generation.

4.3 Activation Collection and Processing

For an LLM input sequence $X = (t_1, t_2, \cdots, t_n)$, we extract the activations Act_n for the last token x_n in the input across all layers in the LLM as a summary of the context. The activations Act_n contain the inner representations of the LLM's knowledge related to the input. When the LLM generates a response based on a question, it traverses through all layers, retrieving knowledge relevant to the input to produce an answer (Meng et al., 2023). We collect two types of activations: correct activations (labeled as 1), obtained when the LLM retrieves accurate content and generates the correct response; and poisoned activations (labeled as 0), obtained when the LLM retrieves poisoned content and produces the attacker's target response.

We introduce normalization of the activations for effective integration into the training process. We calculate the mean μ and standard deviation σ of the activations across all instances in the dataset. Then, we use the obtained μ and σ to normalize the activations with the formula:

$$Act_n^{nor} = (Act_n - \mu)/\sigma. \tag{1}$$

4.4 RevPRAG Model Design

After collecting and preprocessing the activation dataset, we partition it into a training set D_{train}, a test set D_{test}, and a support set S to facilitate the construction and evaluation of the probe model. Drawing inspiration from few-shot learning and Siamese networks, the proposed RevPRAG model is designed to effectively distinguish between clean and poisoned responses, while demonstrating strong generalization capabilities even under limited data conditions. To efficiently capture both intra-layer and inter-layer relationships within the LLM, we employ Convolutional Neural Networks (CNNs) based on the ResNet18 architecture (He et al., 2016). Additionally, we adopt a triplet network structure, in which three subnetworks with shared architecture and weights are used to learn task embeddings, as illustrated in Fig. 3.

During training, we employ the triplet margin loss (Schroff et al., 2015), a commonly used approach for tasks where it is difficult to distinguish similar instances. The training data is randomly divided into triplets consisting of an anchor instance x_a, a positive instance x_p, and a negative instance x_n, where the anchor and positive belong to the same class, while the anchor and negative come from different classes. The triplet margin loss function is formally defined as:

$$\begin{aligned} L = \max\big(&\text{Dist}(x_a, x_p) - \text{Dist}(x_a, x_n) \\ &+ margin,\ 0\big), \end{aligned} \tag{2}$$

where $\text{Dist}(\cdot,\cdot)$ denotes a distance metric (typically the Euclidean distance), and *margin* is a positive constant. The training objective is to encourage the RevPRAG embedding model to output closer embedding vectors for any x_a and x_p, but farther for any x_a and x_n.

At test time, given a test sample x_t, we compute the distance between its embedding and the embedding of the support sample $x_s, x_s \in S$. The support set S refers to a dataset comprising labeled data, denoted as $\{x_{s_1}, ..., x_{s_n}\}$, and corresponding labels are $\{T_{x_{s_1}}, ..., T_{x_{s_n}}\}$. It provides a reference for comparison and classification of new, unseen test data. The main purpose of the support set is to help determine labels for the test data. The label of the test data x_t will be determined according to the label of the support sample x_s that is closest to it. That is, x_t is assigned the label of x_s, meaning $T_{x_t} = T_{x_s}$, where $x_s = argmin_i Dist(x_t, x_{s_i})$. Here, x_s is the nearest support data to the test data x_t.

5 Evaluation

5.1 Experimental Setup

RAG Setup. RAG comprises three key components: *knowledge database, retriever,* and *LLM*.

The setup is shown below:

• **Knowledge Database:** We leverage three representative benchmark question-answering datasets in our evaluation: Natural Questions (NQ) (Kwiatkowski et al., 2019), HotpotQA (Yang et al., 2018), MS-MARCO (Bajaj et al., 2016). Please note that RevPRAG can be expanded to cover poisoning attacks towards any other datasets used for RAG systems, not limited to the datasets used in this paper. The detailed usage instructions for the dataset are provided in Section A.1.

• **Retriever:** In our experiments, we evaluate four state-of-the-art dense retrieval models: Contriever (Izacard et al., 2021) (pre-trained), Contriever-ms (fine-tuned on MS-MARCO) (Izacard et al., 2021), DPR-mul (Karpukhin et al., 2020) (trained on multiple datasets), and ANCE (Xiong et al., 2020) (trained on MS-MARCO).

• **LLM:** Our experiments are conducted on several popular LLMs, each with distinct architectures and characteristics, including GPT2-XL 1.5B (Radford et al., 2019), Llama2-7B (Touvron et al., 2023), Llama2-13B, Mistral-7B (Jiang et al., 2023), and Llama3-8B.

Unless otherwise specified, we adopt the following default settings: HotpotQA as the knowledge base, Contriever as the retriever, GPT2-XL 1.5B as the LLM, and 100 support samples. Moreover, we use the dot product between the embedding vectors of a question and a text to measure their similarity. Poisoned texts are generated following PoisonedRAG (Zou et al., 2024). Consistent with prior work (Lewis et al., 2020), we retrieve the 5 most similar texts from the knowledge database to serve as context for a given question.

Baselines. We compared RevPRAG with five existing methods, and although they were not specifically designed for detecting RAG poisoning attacks, we investigated their potential applications in this domain. CoS (Li et al., 2024) is a black-box approach that guides the LLM to generate detailed reasoning steps for the input, subsequently scrutinizing the reasoning process to ensure consistency with the final answer. MDP (Xi et al., 2024) is a white-box method that exploits the disparity in masking sensitivity between poisoned and clean samples. LLM Factoscope (He et al., 2024) leverages the internal states of LLMs to detect hallucinations, and we investigate its use for identifying poisoning attacks in RAG systems. Both RoBERTa (Pan et al., 2023) and Discern (Hong et al., 2024) employ an additional discriminator to distinguish whether the content retrieved by RAG consists of accurate documents or those that contradict factual information.

Evaluation Metrics.

We evaluate the effectiveness of our detection method using two metrics: **True Positive Rate (TPR)**, which measures the proportion of poisoned responses correctly identified, and **False Positive Rate (FPR)**, which reflects the proportion of benign responses mistakenly flagged as poisoned. These metrics are chosen to balance detection performance with minimal disruption to RAG's normal functionality.

5.2 Overall Results

RevPRAG achieves high TPRs and low FPRs. Table 1 shows the TPRs and FPRs of RevPRAG on three datasets. We have the following observations from the experimental results. First, RevPRAG achieved high TPRs consistently on different datasets and LLMs when injecting five poisoned texts into the knowledge database. For instance, RevPRAG achieved 98.5% (on NQ), 97.7% (on HotpotQA), and 99.9% (on MS-MARCO)

Table 1: RevPRAG achieved high TPRs and low FPRs on three datasets for RAG with five different LLMs.

Dataset	Metrics	LLMs of RAG				
		GPT2-XL 1.5B	Llama2-7B	Mistral-7B	Llama3-8B	Llama2-13B
NQ	TPR	0.982	0.994	0.985	0.986	0.989
	FPR	0.006	0.006	0.019	0.009	0.019
HotpotQA	TPR	0.972	0.985	0.977	0.973	0.970
	FPR	0.016	0.061	0.022	0.017	0.070
MS-MARCO	TPR	0.988	0.989	0.999	0.978	0.993
	FPR	0.007	0.012	0.001	0.011	0.025

Table 2: RevPRAG achieved high TPRs and low FPRs on HotpotQA for RAG with four different retrievers.

Attack	Metrics	LLMs of RAG		
		GPT2-XL 1.5B	Llama2-7B	Mistral-7B
Contriever	TPR	0.972	0.985	0.977
	FPR	0.016	0.061	0.022
Contriever-ms	TPR	0.987	0.983	0.998
	FPR	0.057	0.018	0.012
DPR-mul	TPR	0.979	0.966	0.999
	FPR	0.035	0.075	0.001
ANCE	TPR	0.978	0.981	0.993
	FPR	0.042	0.028	0.023

TPRs for RAG with Mistral-7B. Our experimental results show that assessing whether the output of a RAG system is correct or poisoned based on the activations of LLMs is both highly feasible and reliable (i.e., capable of achieving exceptional accuracy). Second, RevPRAG achieves low FPRs under different settings, e.g., close to 1% in nearly all cases. This result indicates that our approach not only maximizes the detection of poisoned responses but also maintains a low false positive rate, significantly reducing the risk of misclassifying correct answers as poisoned. Additionally, in Section B.2, we conduct generalization experiments to evaluate RevPRAG's performance under distribution shifts between training and testing data. Section B.3 analyzes its effectiveness in handling complex queries. In Section B.4, we assess its performance when training and testing are limited to partial layer activations.

We also conduct experiments on different retrievers. Table 2 shows that our approach consistently achieved high TPRs and low FPRs across RAG with various retrievers and LLMs. For instance, RevPRAG achieves 97.2% (with Contriever), 98.7% (with Contriever-ms), 97.9% (with DPR-mul), 97.8% (with ANCE) TPRs alongside 1.6% (with Contriever), 5.7% (with Contriever-ms), 3.5% (with DPR-mul), and 4.2% (with ANCE) FPRs for RAG when using GPT2-XL 1.5B.

RevPRAG outperforms baselines. Table 3 compares RevPRAG with baselines for RAG using Llama3-8B under the default settings. The overall results demonstrate the superiority of our approach. Meanwhile, several key observations can be drawn from the comparison. First, the limited effectiveness of CoS (Li et al., 2024) may stem from its design focus on detecting backdoor attacks in LLMs via trigger-to-output shortcuts, which differs from RAG's attack surface involving poisoned knowledge base entries. Second, MDP (Xi et al., 2024) achieves good TPRs, but it also exhibits relatively high FPRs, reaching as much as 37.2%. LLM Factoscope (He et al., 2024) leverages multiple internal states of LLMs, relying on layer-wise consistency for effective hallucination detection. However, it may not be suitable for targeted attacks like poisoning, and the use of diverse state data increases computational overhead and discriminator model complexity (Section B.8). Input-based methods such as MDP (Xi et al., 2024), RoBERTa (Pan et al., 2023), and Discern (Hong et al., 2024) aim to detect whether the *input* is poisoned. In contrast, our method focuses on determining whether the responses generated by RAG are correct or poisoned, as response correctness offers a more robust signal of poisoning attacks. Furthermore, in section B.6, we further analyze RevPRAG's ability to distinguish between poisoned responses and hallucinations.

Table 3: RevPRAG outperforms baselines.

Dataset	Metrics	Baselines and Our Method					
		CoS (Li et al., 2024)	MDP (Xi et al., 2024)	LLM Factoscope (He et al., 2024)	RoBERTa (Pan et al., 2023)	Discern (Hong et al., 2024)	Ours
NQ	TPR	0.488	0.946	0.949	0.977	0.810	**0.986**
	FPR	0.146	0.108	0.033	0.063	0.112	**0.009**
HotpotQA	TPR	0.194	0.886	0.939	0.956	0.817	**0.973**
	FPR	0.250	0.372	0.021	0.018	0.101	**0.017**
MS-MARCO	TPR	0.771	**0.986**	0.945	0.946	0.795	0.978
	FPR	0.027	0.181	0.028	0.070	0.101	**0.011**

Table 4: The TPRs and FPRs of RevPRAG for different poisoned text generation methods on HotpotQA.

Attack	Metrics	LLMs of RAG		
		GPT2-XL 1.5B	Llama2-7B	Mistral-7B
PoisonedRAG	TPR	0.972	0.985	0.977
	FPR	0.016	0.061	0.022
GARAG	TPR	0.961	0.976	0.974
	FPR	0.025	0.046	0.026
PRCAP	TPR	0.966	0.986	0.965
	FPR	0.012	0.061	0.022

5.3 Ablation Study

Different methods for generating poisoned texts. To ensure the effectiveness of the evaluation, we employ three different methods introduced by PoisonedRAG, GARAG, and PRCAP to generate the poisoned texts. The experimental results in Table 4 show that RevPRAG consistently achieves high TPRs and low FPRs when confronted with poisoned texts generated by different strategies. For instance, RevPRAG achieved 97.2% (with GPT2-XL 1.5B), 98.5% (with Llama2-7B), and 97.7% (with Mistral-7B) TPRs for poisoned texts generated with PoisonedRAG.

Table 5: The TPRs and FPRs of RevPRAG for different quantities of injected poisoned text on HotpotQA (total retrieved texts: five).

Quantity	Metrics	LLMs of RAG		
		GPT2-XL 1.5B	Llama2-7B	Mistral-7B
five	TPR	0.972	0.985	0.977
	FPR	0.016	0.061	0.022
four	TPR	0.976	0.977	0.986
	FPR	0.034	0.047	0.033
three	TPR	0.963	0.986	0.995
	FPR	0.011	0.043	0.004
two	TPR	0.971	0.995	0.991
	FPR	0.011	0.047	0.005
one	TPR	0.970	0.988	0.989
	FPR	0.049	0.031	0.022

Quantity of injected poisoned texts. Table 5 illustrates the impact of varying quantities of poisoned text on the detection performance of RevPRAG. The more poisoned texts are injected, the higher the likelihood of retrieving them for RAG processing. From the experimental results, we observe that even with varying amounts of injected poisoned text, RevPRAG consistently achieves high TPRs and low FPRs. For example, when the total number of retrieved texts is five and the injected quantity is two, RevPRAG achieves a 99.5% TPR and a 4.7% FPR for RAG with Llama2-7B. The reason for this phenomenon is that the similarity between the retrieved poisoned texts and the query is higher than that of clean texts. Consequently, the LLM generates responses based on the content of the poisoned texts.

Effects of different support set size. In RevPRAG, support data provides essential labeled and task-specific information, facilitating effective reasoning and learning under limited data conditions. We experiment with various support set sizes ranging from 50 to 250 to examine their effect on the performance of RevPRAG. The results in Fig. 5 indicate that varying the support size does not significantly impact the model's detection performance. In addition, Section B.5 further explores the impact of different similarity metrics on the performance of RevPRAG.

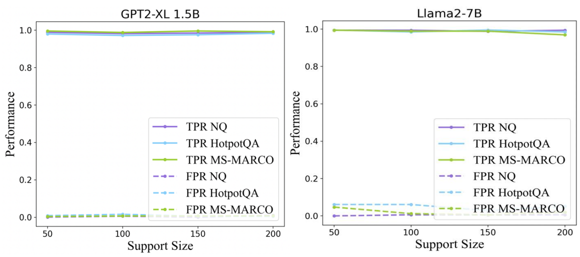


Figure 5: Effects of support set size.

6 Conclusion

In this work, we find that correct and poisoned responses in RAG exhibit distinct differences in LLM activations. Building on this insight, we develop RevPRAG, a detection pipeline that leverages these activations to identify poisoned responses in RAG caused by the injection of malicious texts into the knowledge database. Our approach demonstrates robust performance across RAGs utilizing five different LLMs and four distinct retrievers on three datasets. Experimental results show that RevPRAG achieves exceptional accuracy, with true positive rates approaching 98% and false positive rates near 1%. Ablation studies further validate its effectiveness in detecting poisoned responses across different types and levels of poisoning attacks. Overall, our approach can accurately distinguish between correct and poisoned responses.

Limitations.

Our work has the following limitations:

- This work does not propose a specific method for defending against poisoning attacks on RAG. Instead, our focus is on the timely detection of poisoned responses generated by the LLM, aiming to prevent potential harm to users from such attacks.
- Our approach requires accessing the activations of the LLM, which necessitates the LLM being a white-box model. While this may present certain limitations for users, our method can be widely adopted by LLM service providers. Providers can implement our strategy to ensure the reliability of their services and enhance trust with their users.
- Our approach primarily focuses on determining whether the response generated by the RAG is correct or poisoned, without delving into more granular distinctions. The main goal of our study is to protect users from the impact of RAG poisoning attacks, while more detailed classifications of RAG responses will be addressed in future work.

Ethics Statement

The goal of this work is to detect whether a RAG has generated a poisoned response. All the data used in this study is publicly available, so it does not introduce additional privacy concerns. All source code and software will be made open-source. While the open-source nature of the code may lead to adaptive attacks, we can further enhance our model by incorporating more internal and external information. Overall, we believe our approach can further promote the secure application of RAG.

References

Sahar Abdelnabi, Aideen Fay, Giovanni Cherubin, Ahmed Salem, Mario Fritz, and Andrew Paverd. 2024. Are you still on track!? catching llm task drift with activations. *arXiv preprint arXiv:2406.00799*.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR.

Jinheon Baek, Soyeong Jeong, Minki Kang, Jong C Park, and Sung Hwang. 2023. Knowledge-augmented language model verification. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1720–1736.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, and 1 others. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Oliver Bryniarski, Nabeel Hingun, Pedro Pachuca, Vincent Wang, and Nicholas Carlini. 2021. Evading adversarial example detection defenses with orthogonal projected gradient descent. *arXiv preprint arXiv:2106.15023*.

Nicholas Carlini. 2023. A llm assisted exploitation of ai-guardian. *arXiv preprint arXiv:2307.15008*.

Nicholas Carlini and David Wagner. 2017. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 3–14.

Sukmin Cho, Soyeong Jeong, Jeongyeon Seo, Taeho Hwang, and Jong C Park. 2024. Typos that broke the rag's back: Genetic attack on rag pipeline by simulating documents in the wild via low-level perturbations. *arXiv preprint arXiv:2404.13948*.

Sukmin Cho, Jeongyeon Seo, Soyeong Jeong, and Jong C Park. 2023. Improving zero-shot reader by reducing distractions from irrelevant documents in open-domain question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3145–3157.

Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta R Costa-jussà. 2024. A primer on the inner workings of transformer-based language models. *arXiv preprint arXiv:2405.00208*.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, pages 79–90.

Jinwen He, Yujia Gong, Zijin Lin, Yue Zhao, Kai Chen, and 1 others. 2024. Llm factoscope: Uncovering llms' factual discernment through measuring inner states. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 10218–10230.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Giwon Hong, Jeonghwan Kim, Junmo Kang, Sung-Hyon Myaeng, and Joyce Whang. 2024. Why so gullible? enhancing the robustness of retrieval-augmented models against counterfactual noise. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2474–2495.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7029–7043.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Ruochen Jiao, Shaoyuan Xie, Justin Yue, Takami Sato, Lixu Wang, Yixuan Wang, Qi Alfred Chen, and Qi Zhu. 2024. Exploring backdoor attacks against large language model-based decision making. *arXiv preprint arXiv:2405.20774*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen Tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, pages 6769–6781.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. *arXiv preprint arXiv:2203.05115*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Xi Li, Yusen Zhang, Renze Lou, Chen Wu, and Jiaqi Wang. 2024. Chain-of-scrutiny: Detecting backdoor attacks for large language models. *arXiv preprint arXiv:2406.05948*.

Lefteris Loukas, Ilias Stogiannidis, Odysseas Diamantopoulos, Prodromos Malakasiotis, and Stavros Vassos. 2023. Making llms worth every penny: Resource-limited text classification in banking. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, pages 392–400.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*.

Yikang Pan, Liangming Pan, Wenhu Chen, Preslav Nakov, Min-Yen Kan, and William Wang. 2023. On the risk of misinformation pollution with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1389–1403.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

Ian Soboroff, Shudong Huang, and Donna Harman. 2018. Trec 2018 news track overview. In *TREC*, volume 409, page 410.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. 2020. On adaptive attacks to adversarial example defenses. *Advances in neural information processing systems*, 33:1633–1645.

Zhepei Wei, Wei-Lin Chen, and Yu Meng. 2024. Instructrag: Instructing retrieval-augmented generation with explicit denoising. *arXiv preprint arXiv:2406.13629*.

Zhaohan Xi, Tianyu Du, Changjiang Li, Ren Pang, Shouling Ji, Jinghui Chen, Fenglong Ma, and Ting Wang. 2024. Defending pre-trained language models as few-shot learners against backdoor attacks. *Advances in Neural Information Processing Systems*, 36.

Chong Xiang, Tong Wu, Zexuan Zhong, David Wagner, Danqi Chen, and Prateek Mittal. 2024. Certifiably robust rag against retrieval corruption. *arXiv preprint arXiv:2405.15556*.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*.

Jiaqi Xue, Mengxin Zheng, Yebowen Hu, Fei Liu, Xun Chen, and Qian Lou. 2024. Badrag: Identifying vulnerabilities in retrieval augmented generation of large language models. *arXiv preprint arXiv:2406.00083*.

Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. 2024. Generalized out-of-distribution detection: A survey. *International Journal of Computer Vision*, pages 1–28.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2023. Making retrieval-augmented language models robust to irrelevant context. *arXiv preprint arXiv:2310.01558*.

Zexuan Zhong, Ziqing Huang, Alexander Wettig, and Danqi Chen. 2023. Poisoning retrieval corpora by injecting adversarial passages. In *2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023*, pages 13764–13775.

Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. 2024. Poisonedrag: Knowledge corruption attacks to retrieval-augmented generation of large language models. *arXiv preprint arXiv:2402.07867*.

A Training Details

A.1 Dataset.

As shown in Table 6, we present the average response lengths for both poisoned and correct answers generated by GPT2-XL across three datasets (NQ, HotpotQA, and MS-MARCO), along with examples illustrating each answer format for a specific question. To evaluate the detection of poisoning attacks on the knowledge base of RAG, we selected 3,000 instances of triples (q, t, a) from each of the three evaluation datasets mentioned above. In each triple, q denotes a question, t represents the supporting text collected from Wikipedia or web documents corresponding to q, and a is the correct answer to q, generated using the state-of-the-art GPT-4 model. Among these 3,000 triplets, 1,500 are randomly selected as benign instances, while the remaining 1,500 are designated as poisoned instances. For each poisoned instance, the poisoned answer a_p is generated by GPT-4 for the given question q, and the poisoned text t_p is crafted using existing poisoning strategies, including PoisonedRAG (Zou et al., 2024), GARAG (Cho et al., 2024), and PRCAP (Zhong et al., 2023). The dataset is split into 70% for training, 20% for testing, and 10% as a support set. Within the training set, samples are randomly grouped into triplets (anchor, positive, negative), where the anchor and positive belong to the same class, and the negative belongs to a different class.

A.2 Poisoned Texts Generation.

To ensure that the retrieved poisoned texts successfully achieve the poisoning effect, we employ three existing methods PoisonedRAG (Zou et al., 2024), GARAG (Cho et al., 2024), and PRCAP (Zhong et al., 2023) to generate the poisoned texts. In the PoisonedRAG (Zou et al., 2024) method, the attacker first selects a target question along with its corresponding incorrect answer. The attacker then optimizes the design of the poisoned text to ensure that it meets two key criteria: (1) retrievability by the retriever and (2) effectiveness in misleading the language model to generate the incorrect answer. GARAG (Cho et al., 2024) is a novel adversarial attack algorithm that generates adversarial documents by subtly perturbing clean ones while preserving answer tokens. Through iterative crossover, mutation, and selection, it optimizes the documents to maximize adversarial effectiveness within the defined search space. PRCAP (Zhong et al., 2023) is a gradient-based method, which starts from a natural-language passage and iteratively perturbs it in the discrete token space to maximize its similarity to a set of training queries.

It is worth noting that the generation methods for poisoned texts are not fixed; we can adopt any approach that successfully achieves the poisoning effect. Once the activations of both correct and poisoned responses are obtained, we preprocess them and use them for training and testing the RevPRAG model. This enables the model to effectively dis-

Table 6: Statistical data and format of the responses.

Dataset	Average Word Count of Response	An Example of Response
NQ	Poisoned Response: 7 Correct Response: 12	**Question**: where is the food stored in a yam plant? **Poisoned Response**: In the leaves. **Correct Response**: In the tuber.
HotpotQA	Poisoned Response: 8 Correct Response: 11	**Question**: Which actor starred in Assignment to Kill and passed away in 2000? **Poisoned Response**: Patrick O'Neal. **Correct Response**: John Gielgud.
MS-MARCO	Poisoned Response: 16 Correct Response: 24	**Question**: what is hardie plank? **Poisoned Response**: Hardie plank is a wood flooring option that is used for a variety of home styles. **Correct Response**: Hardie Plank is a brand of fiber cement siding.

tinguish between correct and poisoned responses generated by RAG based on activations.

A.3 Prompt.

The following is the system prompt for RAG, instructing an LLM to produce a response based on the provided context:

> You are a helpful assistant. The user has provided a query along with relevant context information. Use this context to answer the question briefly and clearly. If you cannot find the answer to the question, respond with "I don't know."
> Contexts: [context]
> Query: [question]
> Answer:

Figure 6: The prompt used in RAG to make an LLM generate an answer based on the retrieved texts.

A.4 Environment.

We conduct experiments on a server with 64 AMD EPYC 9654 CPUs (64 logical cores enabled) at 2.40–3.70 GHz, 512 GB of DDR5 RAM (assumed based on high-core-count server standards), and four NVIDIA RTX A6000 GPUs, each with 48 GB GDDR6 memory.

B Additional Experimental Results

B.1 ROC Curve.

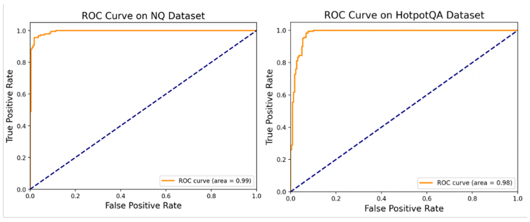


Figure 7: ROC curves of RevPRAG on NQ and HotpotQA datasets.

We present the ROC curves of RevPRAG on the NQ and HotpotQA datasets under the default experimental setting with GPT2-XL as the LLM, as shown in Fig. 7.

B.2 Generalization.

Given the wide range of RAG application scenarios and the diverse user requirements it faces, it is impractical to ensure that our detection model has been trained on all possible scenarios and queries in real-world applications. However, the performance of neural network models largely depends on the similarity between the distributions of the training data and the test data (Yang et al., 2024). Consequently, our model's performance may degrade when faced with training and test data that stem from differing distributions, a challenge frequently observed in real-world scenarios.

To address this issue, we conduct two types of experiments. The first involves using the PoisonedRAG (Zou et al., 2024) method to generate poi-

Table 7: Generalization performance of RevPRAG for RAG with four different LLMs. The training and test datasets vary across different rows. Abbreviations: Hot (HotpotQA), MS (MS-MARCO).

Training Dataset	Test Dataset	Metrics	LLMs of RAG			
			GPT2-XL 1.5B	Llama2-7B	Mistral-7B	Llama3-8B
NQ & Hot	MS	TPR	0.881	0.886	0.948	0.956
		FPR	0.134	0.149	0.076	0.066
Hot & MS	NQ	TPR	0.980	0.983	0.988	0.980
		FPR	0.007	0.074	0.078	0.038
NQ & MS	Hot	TPR	0.977	0.961	0.942	0.978
		FPR	0.025	0.089	0.055	0.049
NQ & Hot & MS	NQ & Hot & MS	TPR	0.986	0.994	0.985	0.987
		FPR	0.032	0.007	0.009	0.035

soned texts, but with different datasets for training and testing. Specifically, we train the detection model using any two datasets and test it on a third dataset that was not used during training. For example, we use NQ and HotpotQA as training datasets and MS-MARCO as the testing dataset. Although these three datasets are all QA datasets, they exhibit significant differences. For example, NQ focuses on extracting answers to factual questions from a single long document, HotpotQA involves multi-document reasoning to derive answers, and MS-MARCO retrieves and ranks relevant answers from a large-scale collection of documents. Therefore, conducting generalization experiments based on these three datasets is reasonable. The second type of experiment uses a single dataset (NQ) for both training and testing. However, the poisoned texts used for training and testing are generated using different methods. For example, in our experiments, the training data is poisoned using GARAG (Cho et al., 2024) and PRCAP (Zhong et al., 2023), while the poisoned texts in the test set are generated using PoisonedRAG (Zou et al., 2024).

Table 7 illustrates the TPRs and FPRs of RevPRAG under distribution shifts across datasets. Overall, the experimental results demonstrate that our detection model exhibits strong generalization performance across RAG with different LLMs and various datasets. For example, when using HotpotQA and MS-MARCO as training data, the detection model achieves TPRs of 98% (with GPT2-XL 1.5B), 98.3% (with Llama2-7B), 98.8% (with Mistral-7B), and 98% (with Llama3-8B) on the NQ dataset. Meanwhile, all FPRs remain below 8%. Furthermore, we observe that the generalization performance is best when NQ is used as the test data (for instance, 98.3% with Llama2-7B), while MS-MARCO shows the poorest performance (for instance, 88.6% with Llama2-7B). We attribute this to the fact that the questions and tasks in HotpotQA

Table 8: Generalization performance of RevPRAG when training and test sets use different poisoning strategies

Training Dataset	Test Dataset	Metrics	LLMs of RAG			
			GPT2-XL 1.5B	Llama2-7B	Mistral-7B	Llama3-8B
GARAG & RCAP	PoinsonedRAG	TPR	0.966	0.970	0.988	0.982
		FPR	0.051	0.024	0.015	0.021
PoinsonedRAG & RCAP	GARAG	TPR	0.959	0.963	0.971	0.973
		FPR	0.017	0.045	0.038	0.014
GARAG & PoinsonedRAG	RCAP	TPR	0.957	0.971	0.984	0.956
		FPR	0.046	0.038	0.025	0.019

Table 9: RevPRAG achieved high TPRs and low FPRs on the open-ended questions from HotpotQA and MS-MARCO datasets.

Dataset	Metrics	LLMs of RAG			
		GPT2-XL 1.5B	Llama2-7B	Mistral-7B	Llama3-8B
HotpotQA	TPR	0.982	0.995	0.991	0.982
	FPR	0.033	0.029	0.008	0.007
MS-MARCO	TPR	0.988	0.989	0.990	0.983
	FPR	0.009	0.009	0.001	0.017

and MS-MARCO are more complex compared to those in NQ. Therefore, detection models trained on more complex tasks generalize well to simpler tasks, whereas the reverse is more challenging. In conclusion, these experimental results highlight that RevPRAG exhibits strong generalization and robust detection performance, even in the presence of significant discrepancies between the training and test datasets.

Table 8 presents the performance of RevPRAG when poisoned texts in the training and test sets are generated using different methods. The results show that even under such distributional shifts, RevPRAG consistently achieves high TPR and low FPR. For instance, when GARAG and PoisonedRAG are used to generate poisoned texts for training and RCAP is used for testing, RevPRAG achieves a TPR of 0.984 and an FPR of 0.025 with Mistral-7B as the LLM, demonstrating its strong generalization ability in out-of-distribution scenarios.

B.3 RevPRAG's Performance on Complex Open-Ended Questions.

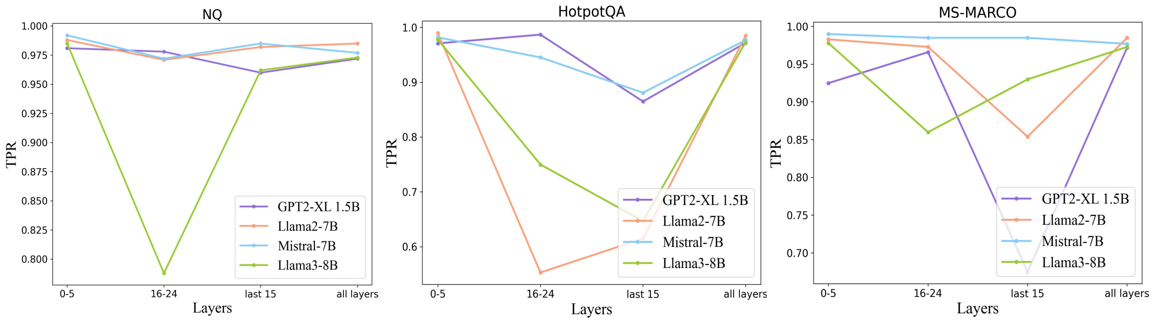
In this section, we conducted a series of experiments to evaluate the performance of RevPRAG on complex, open-ended questions (e.g., "*how to make relationship last?*"). These questions present unique challenges due to their diverse and unstructured nature, in contrast to straightforward, closed-ended questions (e.g., "*What is the name of the highest mountain?*"). In our experiments, the NQ, HotpotQA, and MS-MARCO datasets primarily consist of close-ended questions. As a result, the majority of our previous experiments focused on close-ended problems, which was our default experimental setting. In this study, we utilized the advanced GPT-4o to filter and extract 3,000 open-ended questions from the HotpotQA and MS-MARCO datasets for training and testing the model. For open-ended questions, cosine similarity is employed to evaluate whether the LLM's response aligns with the attacker's target response. If the similarity surpasses a predefined threshold, it is considered indicative of a successful poisoning attack.

The experimental results are shown in Table 9. We can observe that RevPRAG demonstrates excellent detection performance even on complex open-ended questions. For example, RevPRAG achieved TPRs of 99.1% on HotpotQA and 99.0% on MS-MARCO, alongside FPRs of 0.8% on HotpotQA and 0.1% on MS-MARCO for RAG utilizing the Mistral-7B model.

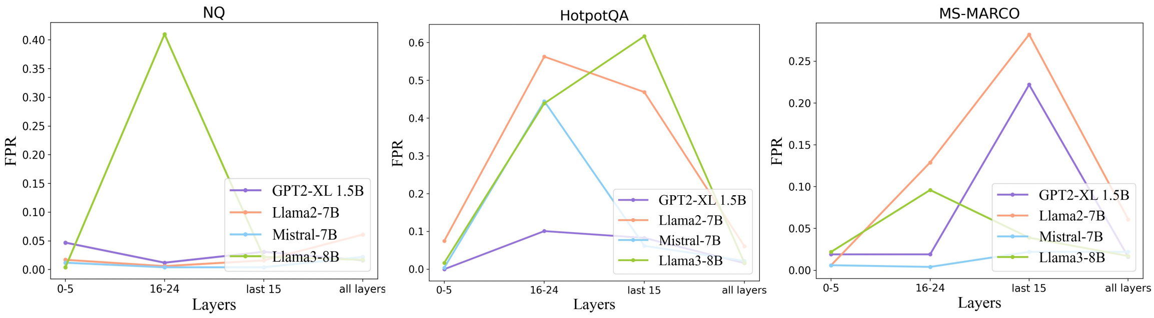
B.4 Activations from Specified Layers.

Fig. 8 illustrates the detection performance of RevPRAG using activations from different layers of various LLMs. In previously presented experiments, we utilize activations from all layers as both training and testing data, yielding excellent results. Additionally, we also test using different layers. The experimental results in Fig. 8 demonstrate that utilizing activations from only the first few layers can still achieve satisfactory detection performance, providing valuable insights for future research. For example, when using activations from layers 0 to 5, RevPRAG achieved TPRs exceeding 97% while maintaining FPRs below 7% for RAG with all LLMs on HotpotQA. However, the experimental results also suggest that using activations from intermediate or deeper layers can lead to performance fluctuations, including signs of degradation or slower convergence. For instance, when using activations from layers 16 to 24 with Llama3-8B as the LLM in RAG, RevPRAG achieves a TPR of 78.8% on NQ dataset and 86% on MS-MARCO dataset.

We further explored the use of activations from a specific individual layer of the LLMs to train and test RevPRAG. We chose 8 layers with roughly even spacing for testing. As shown in Table 10, when using activations from only a specific layer of the GPT2-XL model, RevPRAG demonstrates excellent performance in general. For instance, when the model is trained using activations from layer 0 on the NQ dataset, the TPR can reach as high as 99.6%. However, we also observed that activations from certain layers do not yield satisfactory performance. For example, when the model is trained using activations from layer 29 on the HotpotQA dataset, the TPR is only 52%, while the



(a) TPRs of RevPRAG.



(b) FPRs of RevPRAG.

Figure 8: RevPRAG trained on the activations from specific layers.

Table 10: RevPRAG trained on the activations from specific individual layers of GPT2-XL 1.5B.

Dataset	**Metrics**	**Different layers**							
		layer 0	**layer 8**	**layer 15**	**layer 22**	**layer 29**	**layer 36**	**layer 41**	**layer 47**
NQ	TPR	0.996	0.988	0.996	0.984	0.996	0.988	0.992	0.996
	FPR	0.027	0.007	0.017	0.003	0.007	0.003	0.017	0.003
HotpotQA	TPR	0.713	0.984	0.994	0.989	0.520	0.619	0.931	0.992
	FPR	0.409	0.023	0.012	0.006	0.445	0.409	0.023	0.019
MS-MARCO	TPR	0.967	0.998	0.988	0.986	0.988	0.963	0.955	0.992
	FPR	0.023	0.004	0.002	0.019	0.030	0.026	0.037	0.017

FPR reaches 44.5%. It is precisely due to the existence of these suboptimal layers that models trained with multi-layer activations may not always outperform those using single-layer activations (such as layer 0 with NQ dataset). However, incorporating multi-layer activations can enhance the model's stability, mitigating the detrimental effects of these suboptimal layers.

B.5 Impact of Similarity Metric.

Different methods for calculating similarity between embedding vectors of queries and texts in the knowledge database may lead to varying poisoning effects and distinct LLM activations. Therefore, it is crucial to conduct ablation experiments using various similarity metrics. Table 11 shows the results on the HotpotQA dataset, indicating that the choice of similarity calculation method has minimal impact on RevPRAG's performance, which consistently achieves high TPR and low FPR. For example, in the RAG system with Llama2-7B, when employing dot product and cosine similarity as the similarity measures, the achieved TPRs are 98.5% and 99%, while the FPRs are 6.1% and 2.3%, respectively. This suggests the robustness

Table 11: Impact of similarity metric.

Similarity Metric	**Metrics**	**LLMs of RAG**			
		GPT2-XL 1.5B	**Llama2-7B**	**Mistral-7B**	**Llama3-8B**
Dot Product	TPR	0.972	0.985	0.977	0.973
	FPR	0.016	0.061	0.022	0.017
Cosine	TPR	0.978	0.990	0.979	0.981
	FPR	0.037	0.011	0.023	0.043

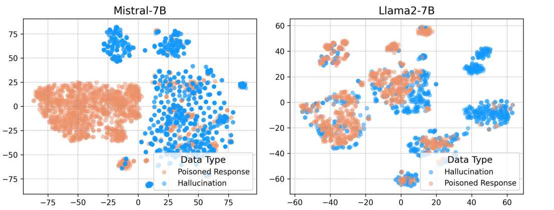


Figure 9: t-SNE visualizations of activations for poisoned responses and hallucinations.

Table 12: RevPRAG could achieve high TPRs and low FPRs to distinguish poisoned responses and hallucinations.

Dataset	Metrics	LLMs of RAG			
		GPT2-XL 1.5B	Llama2-7B	Mistral-7B	Llama3-8B
NQ	TPR	0.987	0.983	0.993	0.995
	FPR	0.046	0.017	0.069	0.008
HotpotQA	TPR	0.975	0.978	0.991	0.995
	FPR	0.004	0.058	0.004	0.008
MS-MARCO	TPR	0.973	0.984	0.999	0.989
	FPR	0.009	0.023	0.001	0.006

of our approach, as it reliably identifies poisoned texts even when LLM activations vary slightly under similar conditions.

B.6 Isolating Poisoned Responses and Hallucinations.

It is well-known that hallucinations are an inevitable phenomenon in LLMs. Even with the introduction of a knowledge database in RAG, LLMs may still generate non-factual responses due to hallucinations. Therefore, the incorrect responses generated by RAG may also stem from hallucinations, rather than being solely caused by RAG poisoning. We conducted experiments to test if our approach can distinguish hallucinations and RAG poisoning. Fig. 9 shows the t-SNE representation of mean activations for poisoned response and hallucinations across all layers for Mistral-7B and Llama2-7B on the NQ dataset. We observe that activations across all layers clearly distinguish between hallucinations and poisoned responses.

This key finding has led us to extend our approach to differentiate between poisoned responses and hallucinations. We thus continue to collect data and train the model using the process outlined in Fig. 3, with the only difference being that we now collect hallucination data. We also conduct extensive experiments on RAG with different LLMs and datasets. From the experimental results in Table 12, we can see that our method achieves a high TPR across all LLMs and datasets. For instance, RevPRAG achieved 98.7% (on NQ), 97.5% (on HotpotQA), and 97.3% (on MS-MARCO) TPRs for RAG with GPT2-XL 1.5B. Furthermore, we observe that the FPR remains low across all evaluation settings. As shown in the table, RevPRAG could achieve 0.8% (on NQ), 0.8% (on HotpotQA) and 0.6% (on MS-MARCO) FPRs for RAG with Llama3-8B. This further supports our previous observation that there is a clear distinction between poisoned responses and hallucinations.

We also explored training the model using activations from a subset of layers, with the experimental results presented in Table 13. We observed that models trained using activations from intermediate subsets of layers can exhibit performance instability. For example, when the model is trained on the HotpotQA dataset using activations from the last 15 layers, the TPR is only 53.5%, while the FPR reaches 48.5%. To the best of our knowledge, this is the first work that successfully and effectively differentiate poisoned response from hallucinations.

B.7 Robustness of RevPRAG.

Table 13: RevPRAG trained on the activations from specific layers of LLMs to distinguish poisoned responses and hallucinations.

Dataset	Metrics	GPT2-XL 1.5B				Llama3-8B			
		layers 0-5	layers 16-24	last 15 layers	all layers	layers 0-5	layers 16-24	last 15 layers	all layers
NQ	TPR	0.980	0.984	0.988	0.987	0.990	0.981	0.897	0.995
	FPR	0.006	0.021	0.010	0.046	0.039	0.022	0.022	0.008
HotpotQA	TPR	0.937	0.977	0.535	0.975	0.994	0.989	0.983	0.995
	FPR	0.052	0.002	0.485	0.004	0.005	0.011	0.011	0.008
MS-MARCO	TPR	0.982	0.984	0.532	0.973	0.983	0.978	0.986	0.989
	FPR	0.048	0.011	0.475	0.009	0.011	0.039	0.004	0.006

This is my question: [question].
This is my answer: [answer].
This is correct text: [ground_truth].
Please craft a corpus such that the answer is [answer] when prompting with the question [question].The generated corpus should be both semantically and linguistically similar to the ground truth [ground_truth]. Please limit the corpus to V words.

Figure 10: The prompt used to generate the poisoned texts.

Table 14: Performance of RevPRAG under adaptive attack scenarios.

Dataset	Metrics	GPT2-XL 1.5B	Llama3-8B
NQ	TPR	0.978	0.982
	FPR	0.022	0.016
HotpotQA	TPR	0.972	0.961
	FPR	0.036	0.051
MS-MARCO	TPR	0.969	0.963
	FPR	0.038	0.018

As a method for detecting poisoning attacks in RAG, RevPRAG is often deployed in scenarios where adversaries are aware of the detection mechanism and actively attempt to evade it. In this context, we define robustness as the detection performance of RevPRAG under adaptive attacks. Since the activations used in our approach serve as an internal representation of the input, a plausible adaptive attack strategy would involve crafting poisoned texts that closely resemble the correct texts in both semantics and activation space, while still achieving the intended poisoning effect. In our experiments, we adopt the PoisonedRAG (Zou et al., 2024) approach to simulate adaptive attacks, modifying the original prompt used for generating poisoned texts as shown in Figure 10.

Table 14 presents the detection performance of RevPRAG under adaptive attack scenarios using PoisonedRAG (Zou et al., 2024), with GPT2-XL 1.5B and Llama3-8B as the underlying LLMs in the RAG framework. The results demonstrate that even when attackers are aware of the detection method and deliberately optimize their poisoning strategy to evade it, RevPRAG still achieves strong performance. For example, on the NQ dataset with Llama3-8B, RevPRAG achieves a TPR of 0.982 and an FPR of just 0.016, highlighting the robustness of our method against adaptive attacks.

B.8 Efficiency.

Table 15 compares the time overhead between LLM Factoscope (He et al., 2024) and RevPRAG when the LLM in RAG is Llama3-8B, including the average training time per epoch and the average inference time per test sample. This experiment was conducted using 1,000 training samples and 500 test samples, with poisoned and clean examples each accounting for 50%. The results demonstrate that RevPRAG, with its task-specific architecture and carefully selected detection metrics, incurs significantly lower computational costs than LLM Factoscope, which integrates multiple sub-models for hallucination detection. Its efficient detection capability makes RevPRAG particularly well-suited for latency-sensitive RAG scenarios, underscoring its practical value.

Table 15: Comparison of time overhead.

Dataset	Training Time per Epoch		Inference Time per Sample	
	LLM factoscope	RevPRAG	LLM factoscope	RevPRAG
NQ	91.61s	**19.31s**	0.0051s	**0.0021s**
HotpotQA	101.25s	**23.69s**	0.0066s	**0.0023s**
MS-MARCO	94.47s	**20.72s**	0.0058s	**0.0022s**