Knowledge Database or Poison Base? Detecting RAG Poisoning Attack through LLM Activations

Xue Tan[1,2], Hao Luan[1,2], Mingyu Luo[1,2], Xiaoyan Sun[3,*], Ping Chen[2,*], Jun Dai[3]
School of Computer Science, Fudan University, Shanghai, China[1]
{xuetan23, hluan23}@m.fudan.edu.cn, luomingyu2002@gmail.com
Institute of Big Data, Fudan University, Shanghai, China[2]
pchen@fudan.edu.cn
Department of Computer Science, Worcester Polytechnic Institute, MA, USA[3]
{xsun7, jdai}@wpi.edu

***Abstract*—As Large Language Models (LLMs) are progressively deployed across diverse fields and real-world applications, ensuring the security and robustness of LLMs has become ever more critical. Retrieval-Augmented Generation (RAG) is a cutting-edge approach designed to address the limitations of large language models (LLMs). By retrieving information from the relevant knowledge database, RAG enriches the input to LLMs, 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 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. Particularly, we introduce 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 correct responses versus poisoned responses. Our results on multiple benchmark datasets and RAG architectures show our approach could achieve *98%* true positive rate, while maintaining false positive rates close to *1%*. We also evaluate recent backdoor detection methods specifically designed for LLMs and applicable for identifying poisoned responses in RAG. The results demonstrate that our approach significantly surpasses them.**

***Index Terms*—LLM, RAG Poisoning, Activation.**

I. Introduction

Recent advances in Large Language Models (LLMs) have remarkably enhanced performance across a diverse set of Natural Language Processing (NLP) tasks, especially in Question-Answering (QA) tasks [1], due to their stunning capabilities to understand and generate texts. Despite these advances, LLMs also face a range of issues arising from knowledge limitations. On one hand, LLMs are trained on historical data and lack up-to-date knowledge (e.g., GPT-4's data cutoff is December 2023 [2], which limits its capability to effectively process information beyond that point). On the other hand, they also exhibit knowledge gaps in specialized domains, such as medicine, finance and privacy-related contexts. Due to the absence of relevant knowledge, LLMs may produce inaccurate content and exhibit "hallucination" behaviors, where they generate information that is incorrect, misleading, or fabricated, yet appears plausible and authoritative. These limitations pose challenges to LLM applications in fields such as healthcare [3], economics [4], and scientific research [5], [6].

Retrieval-Augmented Generation (RAG) [7] 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*. The knowledge database consists of a large amount of texts collected from sources such as latest Wikipedia entries [8], new articles [9] and financial documents [4]. The retriever is primarily responsible for using a text encoder (e.g., BERT [10]) to compute the embedding vectors of the user's question (e.g., "*What is the name of the highest mountain*?") and the texts in the knowledge database and selects the top-k retrieved texts from the knowledge database based on similarity. The LLM generates the final response (e.g., "*Everest*") based on the user's question and the retrieved texts. Due to RAG's powerful knowledge integration capabilities, it has demonstrated impressive performance across a range of QA-like knowledge-intensive tasks [11], [12]. Fig. 1 visualizes an example of RAG.

RAG poisoning refers to the act of injecting malicious or misleading content into the knowledge database, contaminating the retrieved texts in RAG and ultimately leading it 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 [13] demonstrates the feasibility of RAG poisoning by injecting a small amount of maliciously crafted texts into the knowledge database utilized by RAG. This manipulation effectively compels the model to generate the attacker's intended response, referred to as a "poisoned response". Furthermore, injecting a substantial amount of adversarial texts into the knowledge database can mislead the retriever into prioritizing such content, causing it to generalize poisoned responses to queries from unseen domains [14].

*Corresponding Authors: Profs. Ping Chen and Xiaoyan Sun.

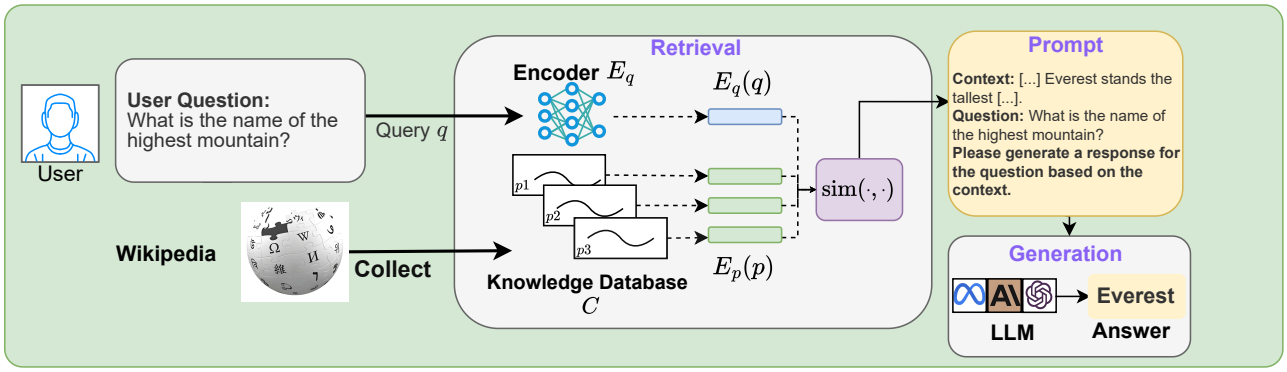


Fig. 1: Visualization of RAG.

The rise of such attacks has drawn significant attention to the necessity of designing robust and resilient RAG systems. For example, SELF-RAG [15] introduces the use of LLM self-reflection to evaluate the usefulness of the retrieved texts; INSTRUCTRAG [16] utilizes LLMs to analyze how to extract correct answers from noisy retrieved documents; RobustRAG [17] 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. Moreover, they're all "best-effort" schemes, offering no guarantee on the defense effectiveness. Once an attacker becomes aware of the defense strategy, they can launch attacks that remain imperceptible to the schemes. Consequently, the user cannot reliably ascertain whether the response generated by such schemes is trustworthy.

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 attacks. We attempt to observe LLM's answer generation process to determine whether the response is compromised or not. 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 [18], [19] has inspired us to leverage LLMs' activations to determine RAG's responses are poisoned answers or not. Specifically, we empirically analyze the activations of the final token in the input sequence across all layers of the LLM. Our findings demonstrate that it is highly feasible to differentiate between these two types of responses by comparing the activations of the LLM when generating correct responses versus poisoned ones. Based on this, we propose a systematic and automated detection pipeline, namely RevPRAG, which consists of three key components: *poisoned data collection*, *activation collection*, and *the design of detection model*, as illustrated in Fig. 2. 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, as well as diverse retrievers such as Contriever, Contriever-ms, DPR-mul, and ANCE. RevPRAG performs consistently well, achieving over 98% accuracy across three different datasets. We also conduct extensive ablation studies to investigate the impact of different components and parameters on the RevPRAG's performance. It is widely recognized that LLMs can generate non-factual responses due to hallucinations; however, such responses are fundamentally different from the poisoned responses resulting from poisoning attacks. To further differentiate whether an incorrect response from RAG is poisoned or non-factual, we also construct corresponding datasets for training and testing. Experimental results show that our proposed method can clearly distinguish between poisoned and non-factual responses as well.

In summary, our contributions are 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 a poisoned or not. By supporting the construction of new datasets to reflect emerging RAG attacks, RevPRAG is capable of detecting new RAG threats.
3) Our model has been empirically validated across various LLM architectures and retrievers, demonstrating over 98% accuracy on our custom-collected detection dataset.

II. Background and Related Work

A. Retrieval Augmented Generation

The LLM community has been challenged by the generation of inaccurate and outdated responses due to their incomplete and static knowledge, and RAG [7], [20] was introduced as a promising ameliorative paradigm to provide more accurate, relevant, and up-to-date external information. RAG is most striking when it leverages the fact that the knowledge database

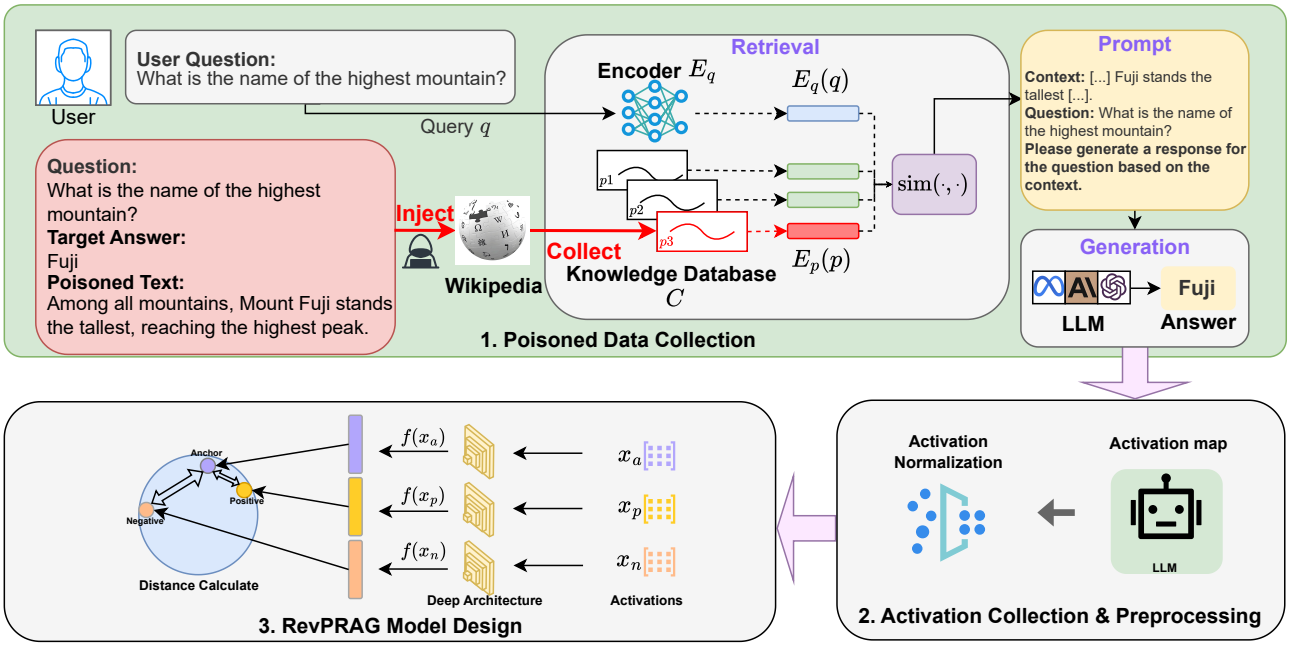


Fig. 2: The workflow of RevPRAG.

can be seamlessly refreshed with newly added content. This feature allows the system to efficiently keep up with dynamic knowledge domains, eliminating the need for costly and time-consuming LLM fine-tuning. Because of its great flexibility, RAG arises in innumerable industrial contexts (e.g., ChatGPT Retrieval Plugin [21], WikiChat [22], FinGPT [23], and ChatRTX [24]) as well as in specialized domains (e.g., medical diagnostic assistants [25] and email/code completion [26]).

RAG comprises three components: knowledge database, retriever, and LLM. The knowledge database contains data that is either newly added or updated the existing information in the LLM's training set, often collected from sources such as Wikipedia [8] and other platforms. We define the knowledge database as $\mathcal{C} = \{p_1, p_2, \cdots, p_m\}$ where p_i represents the i-th piece of collected information. The retriever identifies the k texts most similar to the query q from the knowledge database based on similarity comparison, providing them as external information. The LLM then generates an answer by simultaneously processing the query q and the top-k texts.

The workflow of RAG, having introduced its components, deserves further explanation. As illustrated in Fig. 1, it consists of two main steps: *retrieval* step and *generation* step.

In the retrieval step, the retriever generates 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 dedicated 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. 5 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 and can quickly reference the information in the knowledge database. Therefore, RAG is a significant accomplishment, as it addresses the limitations of LLMs in acquiring up-to-date and domain-specific information.

B. Inner States Analysis of LLM

The interpretability of LLMs has been explored in a collection of works inspired by their inner states and latent spaces. GemmaScope [27] leverages sparse autoencoders to analyze the latent spaces of LLMs, revealing meaningful features within the model activations. These latent states are also capable of encoding safety concerns [28] and task drift [29]. BEEAR [30] detects backdoors by identifying uniform embedding drifts triggered by specific malicious training-time triggers. These methods facilitate the construction of probes to predict the behavior of LLMs, such as hallucinations [31], [32] and task drift [29], as well as to reveal latent knowledge [33].

The logit lens leverages the model's unembedding matrix to decode representations from intermediate layers, providing valuable insights into its decision-making process and facilitating the mitigation of harmful behaviors. The decoding outputs of intermediate layers are analyzed to identify human-

understandable concepts encoded in the parameter vectors of the transformer's feed-forward layers, enabling the manipulation of model outputs to mitigate harmful content [34]. Late-layer MLPs frequently reduce the probability of the maximum-likelihood token, and this reduction can be self-corrected by removing certain attention layers [35]. Intermediate layers are decoded to reveal how harmful outputs are mimicked by language models, identifying decision-making shifts and suggesting the removal of key layers to enhance output accuracy against misleading inputs [36]. Furthermore, LLM Factoscope [19] employs the logit lens to decode intermediate layer outputs, capturing changes in output indices and probabilities within the LLM.

C. 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 [13], [37], [38]. Furthermore, the retriever can also be vulnerable to attacks, which may result in the selection of irrelevant or misleading external information, without even modifying the knowledge database [39], [40]. Specifically, an attacker can inject a small amount of malicious information onto a website, which is then retrieved by RAG [41]. PoisonedRAG [13] 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 [42] was introduced to provide low-level perturbations to RAG and to search for adversarial text targeting both components of RAG simultaneously. PRCAP [14] 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.

D. The Robustness of RAG

Efforts have been made to develop defenses in response to poisoning attacks and noise-induced disruptions, and significant accomplishments have been achieved. As a result, the robustness of RAG, defined by its capacity to resist poisoning attacks and integrate accurate external information, plays a prominent role in practical applications. Adversarial texts are applied to poison the knowledge database [13], [14], [39], necessitating rigorous scrutiny before feeding the retrieved information into the LLM to prevent such adversarial attacks and out-of-distribution data [43]–[46]. RobustRAG [17] mitigates the impact of poisoned texts through a voting mechanism, while INSTRUCTRAG [16] explicitly learns the denoising process to address poisoned and irrelevant information. Other approaches to enhance robustness include prompt design [47], [48], plug-in models [49], and specialized models [15], [50]. While these methods alleviate the issues of poisoning and noise, they share similar limitations—specifically, that adaptive attackers can often circumvent the defenses once the underlying algorithms are made publicly available [51]–[55]. In this work, we present a detection perspective on defending against RAG attacks, which allows for the open sharing of detection algorithms and models without the risk of being easily circumvented by adaptive attacks.

III. PRELIMINARY

In this section, we first define our threat model, and present our design objectives. We then detail the rationale behind detecting poisoning attacks through activation analysis.

A. Threat Model

Attacker's Goal. 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. Taking the upper half of Fig. 2 as an example, the attacker's target question q_i is "*What is the name of the highest mountain*?", with the target answer being "*Fuji*". To achieve this, the attacker injects malicious texts into the knowledge database C, effectively contaminating its content as a source of information for RAG. Consequently, the LLM generates the incorrect target answer "*Fuji*" based on the retrieved poisoned texts.

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, such as Contriever [56], Contriever-ms (fine-tuned on MS-MARCO) [56], and ANCE [57]. The poisoned texts can be integrated into the RAG's knowledge database through two ways: the attacker publishing the maliciously crafted content on open platforms like Wikipedia, or utilizing data collection agencies to disseminate the poisoned texts.

B. Design Objectives

Accuracy. We aim to develop an accurate model for detecting poisoned responses. Accuracy, in this context, requires identifying all poisoned responses while keeping false rates to a minimum. A high false positive rate would incorrectly flag many valid responses as poisoned, disrupting the model's functionality and reducing its effectiveness. Conversely, a high false negative rate would allow poisoned responses to go undetected. This work focuses on minimizing both false positives and false negatives to ensure reliable detection.

Non-intrusiveness. Another key objective is to preserve the integrity of the RAG system by avoiding modifications to its core architecture. Our aim is to design a detection model that is both highly accurate and easy to integrate. To ensure practicality, we steer clear of resource-intensive changes, such as fine-tuning the large model, which can be costly. Instead, we focus

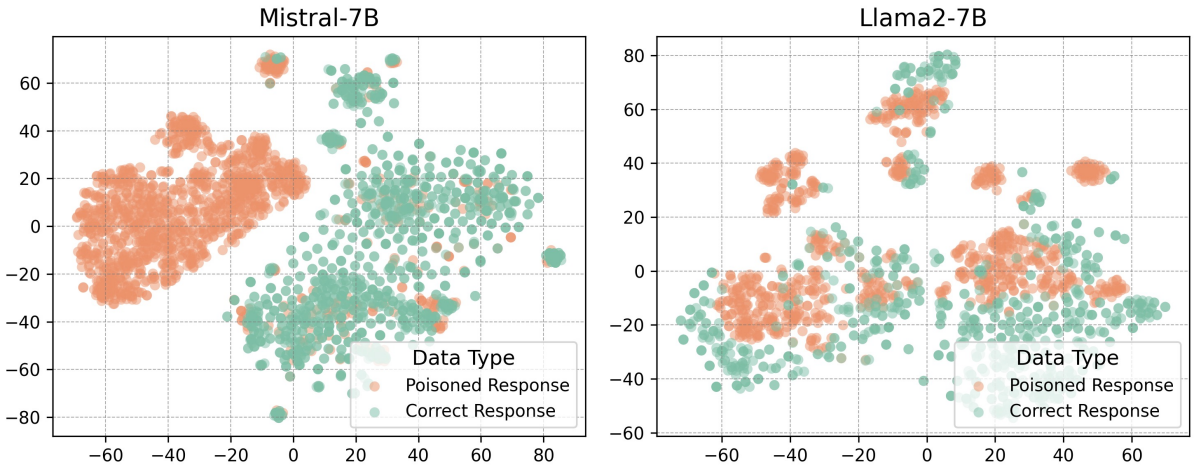


Fig. 3: t-SNE visualizations of activations for correct and poisoned responses.

on achieving accurate detection of poisoned responses within the existing RAG workflow, leveraging available information without introducing additional overhead.

C. Rationale

The rationale behind our research stems from the conflict between the poisoned texts in the knowledge database and the knowledge originally learned by the LLMs. This conflict arises as attackers seek to manipulate the models into generating poisoned responses that contradict accurate information. This inherent conflict drives our efforts to develop methods for detecting and exploiting it. 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 process of the LLM. Consequently, we posit that activations can, to a certain extent, capture the aforementioned conflict. Fig. 3 illustrates the t-SNE [58] representation of the mean activations across all layers for Mistral-7B and Llama2-7B on the NQ dataset, comparing poisoned and correct responses. The results clearly highlight the distinguishability between correct and poisoned responses. In summary, activations from all layers of the LLM can effectively serve as indicators for detecting poisoned responses in RAG systems.

IV. Methodology

This section begins by providing an overview of our pipeline. We then describe the methodology for crafting effective poisoned texts to execute attacks. Following the successful influence of the poisoned texts on the LLM's responses, we collect and process the corresponding activations. Finally, we present the architectural design of the RevPRAG model.

A. Overview of the method

We introduce RevPRAG, a pipeline designed to leverage LLM activations for detecting knowledge poisoning attacks in RAG systems, as illustrated in Fig. 2. This pipeline enables seamless expansion of dataset coverage for the poisoning attack detection model.

To detect poisoning attacks against RAG, we begin by collecting the texts used in the attack. These poisoned texts are injected into the RAG system's knowledge database, contaminating the previously clean retrieval texts. Using the poisoned texts as a basis, we label the activation data based on whether the LLM generates the attacker's target response. Additionally, we capture the LLM's activations for all queries.

After collecting and preprocessing the data, we train a Siamese network-based model. This approach minimizes the embedding distance for data within the same class while maximizing the distance between data from different classes, ensuring robust detection of poisoning attacks.

B. Poisoned 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 and with our primary focus on detecting poisoning attacks, we employ the following three state-of-the-art strategies (e.g., PoisonedRAG [13], GARAG [42], PAPRAG [14]) for generating poisoned texts and increasing the similarity between the poisoned texts and the queries, to heighten the likelihood that the poisoned texts would be selected by the retriever. The retrieved texts, along with the question, are then used to construct a new prompt for the LLM

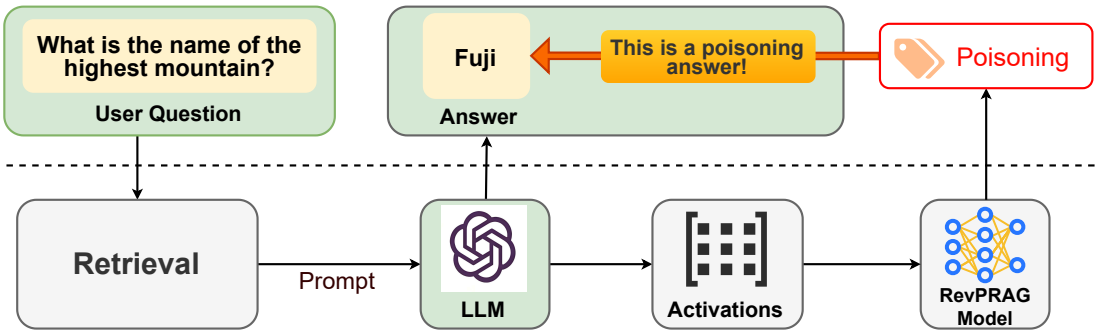


Fig. 4: An instance of using RevPRAG.

to generate the answer. The prompt [13], as shown in Fig. 5, is employed to achieve this objective.

> 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:

Fig. 5: The prompt used in RAG to let an LLM generate an answer based on the retrieved texts.

C. Activation Collection and Processing

For an LLM input sequence $X = (x_1, x_2, \cdots, x_n)$, 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 [59]. When clean and factual texts are retrieved as input, they align with the LLM's original knowledge. In contrast, when poisoned texts are input, they conflict with the LLM's original knowledge. These differing scenarios are clearly reflected in the model's activations.

We introduce normalization of the activations for effective integration into the training process. We calculate the mean μ and standard deviation σ of the dataset. Then, we use the obtained μ and σ to normalize the activations with the formula: $Act_{nor} = (A_n - \mu)/\sigma$. This standardization improves training efficiency by scaling activations uniformly, preventing bias towards larger features. It also ensures smoother optimization, alleviates gradient-related issues, and enhances the performance of algorithms that rely on distance metrics.

D. RevPRAG Model Design

After collecting and preprocessing the dataset of activations, we design a probing mechanism based on the dataset. Inspired by few-shot learning and Siamese networks, our proposed RevPRAG model is adept at distinguishing correct answers from poisoned ones and demonstrates strong generalizability. LLMs are known to generate non-factual responses due to hallucinations as well, but these differ from poisoned responses resulting from poisoning attacks. To further differentiate whether an incorrect response from RAG is poisoned or non-factual, we also construct corresponding datasets for training and testing. Experimental results show that our proposed method can also clearly distinguish between poisoned and non-factual responses. This capability will further assist users in verifying responses from LLMs, enhancing model interpretability.

In order to efficiently capture the relationships between and within different layers of the LLM, we utilize Convolutional Neural Networks (CNNs) with the ResNet18 architecture [60]. We use triplet networks which share the same architecture and weights to learn embeddings of the tasks as shown in Fig. 2. During training, we employ the triplet margin loss [61], a commonly used approach for tasks where it is difficult to distinguish similar instances. Triplet margin loss is a loss function used in training neural networks for tasks such as face recognition or object classification. Its goal is to learn a similarity metric within a high-dimensional embedding space (also known as feature space), where representations of similar objects (e.g., images of the same person) are close to each other, while representations of dissimilar objects are farther apart. This powerful similarity metric provided by triplet margin loss is particularly suitable for distinguishing LLM activations, enabling RevPRAG to more effectively differentiate activation differences caused by poisoning attacks. The core concept of triplet margin loss involves the use of triplets, each consisting of an anchor sample, a positive sample, and a negative sample. The anchor and positive samples represent similar instances,

while the negative sample represents a dissimilar one. The algorithm learns to embed these samples such that the distance between the anchor and positive sample is smaller than the distance between the anchor and negative sample.

Given training samples x_a, x_p, and x_n, they represent anchor, positive, and negative points, respectively. The RevPRAG embedding model will output closer embedding vectors for any Act_a and Act_p, but farther for any Act_a and Act_n. The loss function is formally defined as: $L = \max(\text{Dist}\,(Act_a, Act_p) - \text{Dist}\,(Act_a, Act_n) + margin, 0)$, where $\text{Dist}(\cdot,\cdot)$ denotes a distance metric (typically the Euclidean distance), and *margin* is a positive constant. The presence of the *margin* introduces an additional parameter in triplet loss that requires tuning. If the *margin* is too large, the model's loss will be high, and it may struggle to converge to zero. However, a larger *margin* generally improves the model's ability to distinguish between very similar samples, making it easier to differentiate between Act_p and Act_n. Conversely, if the *margin* is too small, the loss will quickly approach zero, making the model easier to train, but it will be less effective at distinguishing between Act_p and Act_n. At test time, given a test sample x_t, we compute the distance between the embedding of Act_t and Act_s, where Act_s is the embedding of the support sample $x_s \in S$. S is named support set, the dataset we provide, which contains labeled data that has not been used in the training set. Ultimately, the label of the test data x_t will match the label of the support sample x_s that is closest to it.

V. Evaluation

A. 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 QA datasets in our evaluation: Natural Questions (NQ) [62], HotpotQA [63], MS-MARCO [64]. To evaluate the detection of poisoning attacks to the knowledge database of RAG, we selected 3,000 instances of the triple (***q***, ***t***, ***a***) from each of the three datasets. In each triple, ***q*** is a question from the datasets, ***t*** is the texts collected from Wikipedia or web documents corresponding to ***q***, and ***a*** is the correct answer to the question ***q***, generated using a state-of-the-art LLM. To better simulate the RAG poisoning attack scenario implemented through the knowledge database, we will employ three different methods for generating poisoning texts in the experiments, including PoisonedRAG [13] , GARAG [42], and PRCAP [14]. We will evaluate the performance of our proposed detection approach across this series of different scenarios.

• **Retriever:** In our experiments, we evaluate four state-of-the-art dense retrieval models: Contriever [56] (pre-trained), Contriever-ms (fine-tuned on MS-MARCO) [56], DPR-mul [65] (trained on multiple datasets), and ANCE [57] (trained on MS-MARCO). Following previous works [7], [14], we default to using dot product between the embedding vectors of a question and a document in the knowledge database to calculate their similarity score.

• **LLM:** Our experiments are conducted on several popular LLMs, each with distinct architectures and characteristics. We employ GPT2-XL 1.5B [66], Llama2-7B [67], Llama2-13B, Mistral-7B [68], and Llama-3-8B. We use the prompt shown in Fig. 5 to guide the LLM in generating an answer to a question.

In addition to the aforementioned aspects, we adopt the following default settings: a knowledge database based on HotpotQA, the Contriever retriever, GPT-2-XL 1.5B as the LLM, and a support size of 100. Moreover, we use the dot product between the embedding vectors of a question and a text to measure their similarity. The generation of poisoning texts follows the scheme outlined in PoisonedRAG [13]. Consistent with prior work [7], we retrieve the 5 most similar texts from the knowledge database to serve as context for a given question.

Baselines: To the best of our knowledge, there are currently no dedicated detection methods or evaluations specifically for RAG poisoning attacks. Thus, we extend two current methods [69] and [70] for the RAG scenario. CoS [69] 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 [70] is a white-box method that exploits the disparity in masking sensitivity between poisoned and clean samples.

Evaluation Metrics: The effectiveness of the proposed detection method is assessed using two metrics:

• **The True Positive Rate (TPR)**, which measures the proportion of effectively poisoned responses that are successfully detected. A higher TPR signifies better detection performance for poisoned responses, with a correspondingly lower rate of missed detections (i.e., lower false negative rate).

• **The False Positive Rate (FPR)**, which quantifies the proportion of correct responses that are misclassified as being

TABLE I: 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

caused by poisoning attacks. A lower FPR indicates fewer false positives for correct answers, minimizing disruption to the normal operation of RAG. Our primary goal is to detect poisoned responses as effectively as possible while minimizing the impact on RAG's normal functionality, which is why we have selected these two metrics.

B. Overall Results

RevPRAG achieves high TPRs and low FPRs. Table I 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) TPRs for RAG with Mistral-7B. Our experimental results show that assessing whether the output of a RAG system is accurate 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. This ensures the reliable operation of RAG systems, making the approach highly practical for real-world applications.

TABLE II: 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

TABLE III: RevPRAG outperforms baselines.

Dataset	Metrics	Methods		
		CoS	MDP	Ours
NQ	TPR	0.488	0.946	**0.986**
	FPR	0.146	0.108	**0.009**
HotpotQA	TPR	0.194	0.886	**0.973**
	FPR	0.250	0.372	**0.017**
MS-MARCO	TPR	0.771	0.986	**0.978**
	FPR	0.027	0.181	**0.011**

Table II shows the TPRs and FPRs of RevPRAG on HotpotQA for RAG with different retrievers and LLMs. Results show that our approach consistently achieved high TPRs and low FPRs across RAG with various retrievers and LLMs. For instance, RevPRAG could achieve 97.2% (with Contriever), 98.7% (with Contriever-ms), 97.9% (with DPR-mul), and 97.8% (with ANCE) TPRs for RAG with GPT2-XL 1.5B.

RevPRAG outperforms baselines. Table III compares RevPRAG with baselines for RAG with Llama3-8B under the default settings. We have the following observations. First, the Chain-of-Scrutiny (CoS), a backdoor detection method based on reasoning chain analysis, has demonstrated limited effectiveness. We attribute this to the fact that CoS is specifically designed for detecting backdoor attacks in LLMs, relying on the shortcut from the trigger to the target output. This differs from RAG, where backdoor attacks are carried out by injecting poisoned texts into the knowledge database. Second, MDP achieves good TPRs, but it also exhibits relatively high FPRs, reaching as much as 37.2%. However, MDP is an input-based approach that focuses on detecting whether the *input* is poisoned. In contrast, our approach concentrates on determining whether the *responses* generated by RAG are correct or poisoned, as we observe that the correctness (or not) of RAG's responses provides greater robustness in indicating poisoning attacks.

C. 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 [71]. 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 a series of generalization experiments. 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.

Table IV illustrates the TPRs and FPRs of RevPRAG for RAG with four different LLMs. 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%

TABLE IV: 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

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 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.

TABLE V: 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

TABLE VI: 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

D. Ablation Study

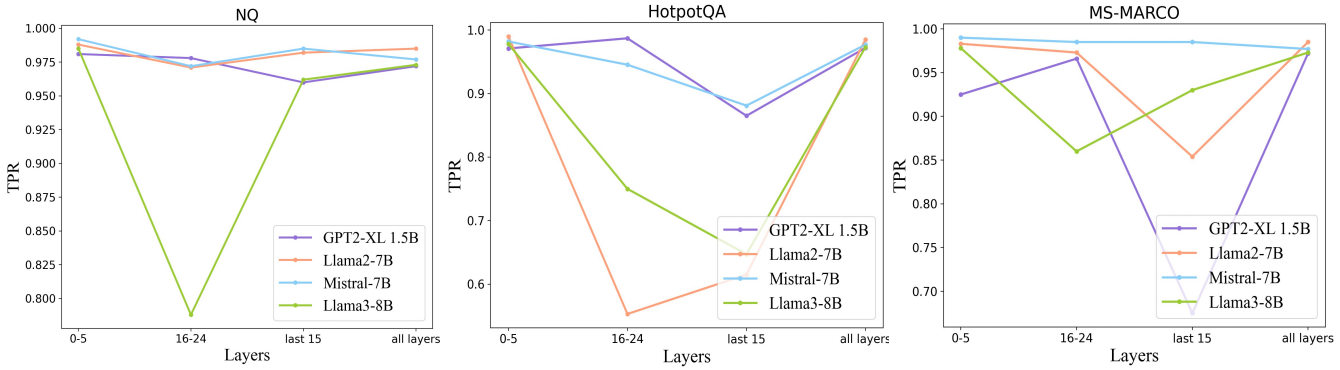
Different methods for generating poisoned texts. To ensure the effectiveness of the evaluation, we employ three different methods introduced by PoisonedRAG, GARAG, and PRCAP to generate the poisoned texts. The experimental results in Table V show that RevPRAG consistently achieves high TPRs and low FPRs 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. Our method focuses on detecting the outputs of RAG rather than analyzing the inputs. In other words, regardless of the content of inputs into LLM or the poisoned text generation strategy, our method can reliably detect poisoned responses as long as they occur.

Quantity of injected poisoned texts. Table VI illustrates the impact of varying quantities of poisoned text on the detection performance of RevPRAG. Injecting different amounts of poisoned text into the knowledge database impacts the likelihood of retrieving poisoned content. The more poisoned texts are injected, the higher the likelihood of retrieving them for RAG processing. This is because one of the optimization objectives for generating poisoned text is to maximize its similarity to the target query. In our experiments, we find that when five poisoned texts are injected into the knowledge database, there is a high likelihood of retrieving all five poisoned texts when the total number of retrieved texts is also five (following practice of prior work). 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.

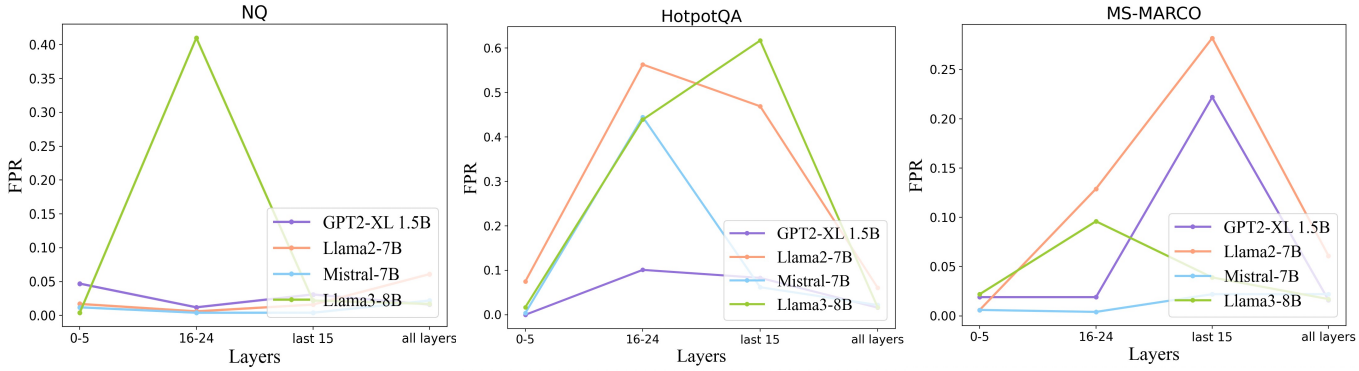
TABLE VII: Impact of similarity metric.

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

Impact of similarity metric. Different methods for calculating similarity between embedding vectors of queries and



(a) TPRs of RevPRAG.



(b) FPRs of RevPRAG.

Fig. 6: Activations from specified layers.

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

Activations from specified layers. Fig. 6 illustrates the detection performance of RevPRAG using activations from different layers of various LLMs on HotpotQA. In our approach, we utilize activations from all layers as both training and testing data, yielding excellent results. Additionally, the experimental results 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. Furthermore, the experimental results 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.

Effects of different support set size. We experiment with various support set sizes ranging from 50 to 250 to examine their effect on the performance of RevPRAG. This evaluation was conducted on Llama2-7B with different datasets. The results in Fig. 7 indicate that varying the support size does not significantly impact the model's detection performance.

E. 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. Fig. 8 shows the t-SNE [58] 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.

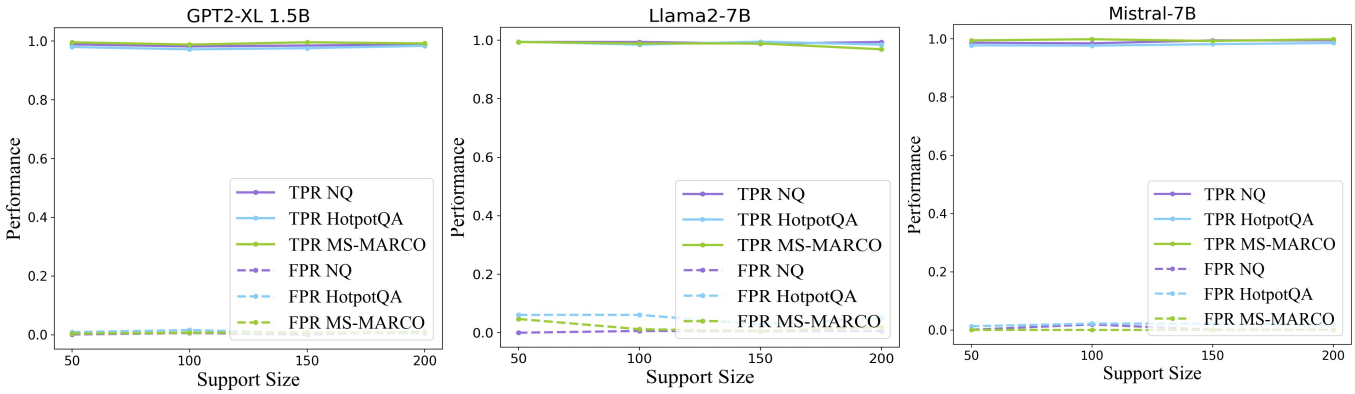


Fig. 7: Effects of support set size.

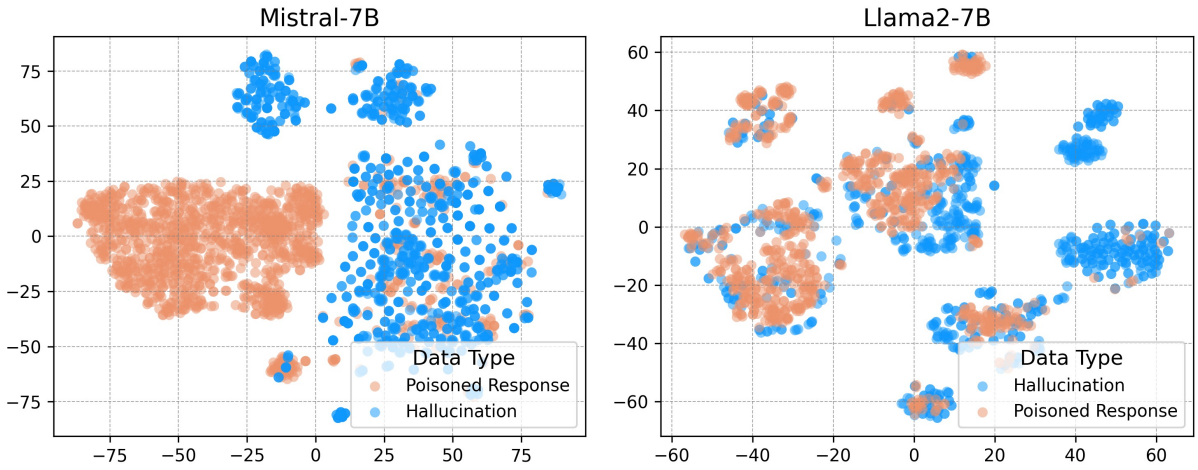


Fig. 8: t-SNE visualizations of activations for poisoned responses and hallucinations.

TABLE VIII: RevPRAG could achieve high TPRs and low FPRs for distinguishing between 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

We continue to collect data and train the model using the process outlined in Fig. 2, 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 VIII, we can see that our method achieves a high True Positive Rate (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.

To the best of our knowledge, this is the first work that successfully achieves to effectively differentiate poisoned response from hallucinations.

VI. 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 activation patterns to identify poisoned responses in RAG caused by the injection of malicious text 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 confirm its effectiveness in detecting poisoned responses under various poisoning attack types and intensities. Additionally, we validate our method's ability to distinguish poisoned responses from non-factual responses induced by hallucinations, ensuring precise identification of genuinely poisoned outputs. Overall, our approach reliably detects poisoned responses without disrupting the functionality of RAG systems.

VII. Acknowledgments

The authors affiliated with Fudan University are supported by the National Key R&D Program of China 2023YFB3107404.

References

[1] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer, "Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1601–1611.

[2] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[3] C. Wang, J. Ong, C. Wang, H. Ong, R. Cheng, and D. Ong, "Potential for gpt technology to optimize future clinical decision-making using retrieval-augmented generation," *Annals of Biomedical Engineering*, vol. 52, no. 5, pp. 1115–1118, 2024.

[4] L. Loukas, I. Stogiannidis, O. Diamantopoulos, P. Malakasiotis, and S. Vassos, "Making llms worth every penny: Resource-limited text classification in banking," in *Proceedings of the Fourth ACM International Conference on AI in Finance*, 2023, pp. 392–400.

[5] V. Kumar, L. Gleyzer, A. Kahana, K. Shukla, and G. E. Karniadakis, "Mycrunchgpt: A llm assisted framework for scientific machine learning," *Journal of Machine Learning for Modeling and Computing*, vol. 4, no. 4, 2023.

[6] J. Boyko, J. Cohen, N. Fox, M. H. Veiga, J. I. Li, J. Liu, B. Modenesi, A. H. Rauch, K. N. Reid, S. Tribedi *et al.*, "An interdisciplinary outlook on large language models for scientific research," *arXiv preprint arXiv:2311.04929*, 2023.

[7] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

[8] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych, "Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models," *arXiv preprint arXiv:2104.08663*, 2021.

[9] I. Soboroff, S. Huang, and D. Harman, "Trec 2018 news track overview." in *TREC*, vol. 409, 2018, p. 410.

[10] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of naacL-HLT*, vol. 1. Minneapolis, Minnesota, 2019, p. 2.

[11] A. Lazaridou, E. Gribovskaya, W. Stokowiec, and N. Grigorev, "Internet-augmented language models through few-shot prompting for open-domain question answering," *arXiv preprint arXiv:2203.05115*, 2022.

[12] S. Jeong, J. Baek, S. Cho, S. J. Hwang, and J. C. Park, "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)*, 2024, pp. 7029–7043.

[13] W. Zou, R. Geng, B. Wang, and J. Jia, "Poisonedrag: Knowledge corruption attacks to retrieval-augmented generation of large language models," *arXiv preprint arXiv:2402.07867*, 2024.

[14] Z. Zhong, Z. Huang, A. Wettig, and D. Chen, "Poisoning retrieval corpora by injecting adversarial passages," in *2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023*, 2023, pp. 13 764–13 775.

[15] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-rag: Learning to retrieve, generate, and critique through self-reflection," *arXiv preprint arXiv:2310.11511*, 2023.

[16] Z. Wei, W.-L. Chen, and Y. Meng, "Instructrag: Instructing retrieval-augmented generation with explicit denoising," *arXiv preprint arXiv:2406.13629*, 2024.

[17] C. Xiang, T. Wu, Z. Zhong, D. Wagner, D. Chen, and P. Mittal, "Certifiably robust rag against retrieval corruption," *arXiv preprint arXiv:2405.15556*, 2024.

[18] J. Ferrando, G. Sarti, A. Bisazza, and M. R. Costa-jussà, "A primer on the inner workings of transformer-based language models," *arXiv preprint arXiv:2405.00208*, 2024.

[19] J. He, Y. Gong, Z. Lin, Y. Zhao, K. Chen *et al.*, "Llm factoscope: Uncovering llms' factual discernment through measuring inner states," in *Findings of the Association for Computational Linguistics ACL 2024*, 2024, pp. 10 218–10 230.

[20] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, "Retrieval augmented language model pre-training," in *International conference on machine learning*. PMLR, 2020, pp. 3929–3938.

[21] "Chatgpt knowledge retrieval," https://platform.openai.com/docs/assistants/tools/knowledge-retrieval, 2023.

[22] S. Semnani, V. Yao, H. C. Zhang, and M. Lam, "Wikichat: Stopping the hallucination of large language model chatbots by few-shot grounding on wikipedia," in *The 2023 Conference on Empirical Methods in Natural Language Processing: EMNLP 2023*, 2023, pp. 2387–2413.

[23] B. Zhang, H. Yang, T. Zhou, M. Ali Babar, and X.-Y. Liu, "Enhancing financial sentiment analysis via retrieval augmented large language models," in *Proceedings of the fourth ACM international conference on AI in finance*, 2023, pp. 349–356.

[24] "Nivida chatrtx: Chat with rtx," https://www.nvidia.com/en-us/ai-on-rtx/chatrtx/, 2024.

[25] S. Siriwardhana, R. Weerasekera, E. Wen, T. Kaluarachchi, R. Rana, and S. Nanayakkara, "Improving the domain adaptation of retrieval augmented generation (rag) models for open domain question answering," *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 1–17, 2023.

[26] M. R. Parvez, W. Ahmad, S. Chakraborty, B. Ray, and K.-W. Chang, "Retrieval augmented code generation and summarization," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2021, pp. 2719–2734.

[27] T. Lieberum, S. Rajamanoharan, A. Conmy, L. Smith, N. Sonnerat, V. Varma, J. Kramár, A. Dragan, R. Shah, and N. Nanda, "Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2," *arXiv preprint arXiv:2408.05147*, 2024.

[28] A. Zou, L. Phan, S. Chen, J. Campbell, P. Guo, R. Ren, A. Pan, X. Yin, M. Mazeika, A.-K. Dombrowski *et al.*, "Representation engineering: A top-down approach to ai transparency," *arXiv preprint arXiv:2310.01405*, 2023.

[29] S. Abdelnabi, A. Fay, G. Cherubin, A. Salem, M. Fritz, and A. Paverd, "Are you still on track!? catching llm task drift with activations," *arXiv preprint arXiv:2406.00799*, 2024.

[30] Y. Zeng, W. Sun, T. N. Huynh, D. Song, B. Li, and R. Jia, "Beear: Embedding-based adversarial removal of safety backdoors in instruction-tuned language models," *arXiv preprint arXiv:2406.17092*, 2024.

[31] D. Zhu, D. Chen, Q. Li, Z. Chen, L. Ma, J. Grossklags, and M. Fritz, "Pollmgraph: Unraveling hallucinations in large language models via state transition dynamics," *arXiv preprint arXiv:2404.04722*, 2024.

[32] C.-W. Sky, B. Van Durme, J. Eisner, and C. Kedzie, "Do androids know they're only dreaming of electric sheep?" in *Findings of the Association for Computational Linguistics ACL 2024*, 2024, pp. 4401–4420.

[33] C. Burns, H. Ye, D. Klein, and J. Steinhardt, "Discovering latent knowledge in language models without supervision," *arXiv preprint arXiv:2212.03827*, 2022.

[34] M. Geva, A. Caciularu, K. Wang, and Y. Goldberg, "Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022, pp. 30–45.

[35] T. McGrath, M. Rahtz, J. Kramar, V. Mikulik, and S. Legg, "The hydra effect: Emergent self-repair in language model computations," *arXiv preprint arXiv:2307.15771*, 2023.

[36] D. Halawi, J.-S. Denain, and J. Steinhardt, "Overthinking the truth: Understanding how language models process false demonstrations," *arXiv preprint arXiv:2307.09476*, 2023.

[37] J. Xue, M. Zheng, Y. Hu, F. Liu, X. Chen, and Q. Lou, "Badrag: Identifying vulnerabilities in retrieval augmented generation of large language models," *arXiv preprint arXiv:2406.00083*, 2024.

[38] R. Jiao, S. Xie, J. Yue, T. Sato, L. Wang, Y. Wang, Q. A. Chen, and Q. Zhu, "Exploring backdoor attacks against large language model-based decision making," *arXiv preprint arXiv:2405.20774*, 2024.

[39] Q. Long, Y. Deng, L. Gan, W. Wang, and S. J. Pan, "Backdoor attacks on dense passage retrievers for disseminating misinformation," *arXiv preprint arXiv:2402.13532*, 2024.

[40] P. Cheng, Y. Ding, T. Ju, Z. Wu, W. Du, P. Yi, Z. Zhang, and G. Liu, "Trojanrag: Retrieval-augmented generation can be backdoor driver in large language models," *arXiv preprint arXiv:2405.13401*, 2024.

[41] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "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*, 2023, pp. 79–90.

[42] S. Cho, S. Jeong, J. Seo, T. Hwang, and J. C. Park, "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*, 2024.

[43] B. Wang, C. Xu, S. Wang, Z. Gan, Y. Cheng, J. Gao, A. H. Awadallah, and B. Li, "Adversarial glue: A multi-task benchmark for robustness evaluation of language models," *arXiv preprint arXiv:2111.02840*, 2021.

[44] X. Li, M. Liu, S. Gao, and W. Buntine, "A survey on out-of-distribution evaluation of neural nlp models," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 2023, pp. 6683–6691.

[45] B. Wang, W. Chen, H. Pei, C. Xie, M. Kang, C. Zhang, C. Xu, Z. Xiong, R. Dutta, R. Schaeffer *et al.*, "Decodingtrust: A comprehensive assessment of trustworthiness in gpt models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 31 232–31 339, 2023.

[46] K. Zhu, J. Wang, J. Zhou, Z. Wang, H. Chen, Y. Wang, L. Yang, W. Ye, Y. Zhang, N. Zhenqiang Gong *et al.*, "Promptbench: Towards evaluating the robustness of large language models on adversarial prompts," *arXiv e-prints*, pp. arXiv–2306, 2023.

[47] S. Cho, J. Seo, S. Jeong, and J. C. Park, "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*, 2023, pp. 3145–3157.

[48] O. Press, M. Zhang, S. Min, L. Schmidt, N. A. Smith, and M. Lewis, "Measuring and narrowing the compositionality gap in language models," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 5687–5711.

[49] J. Baek, S. Jeong, M. Kang, J. C. Park, and S. Hwang, "Knowledge-augmented language model verification," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 1720–1736.

[50] O. Yoran, T. Wolfson, O. Ram, and J. Berant, "Making retrieval-augmented language models robust to irrelevant context," *arXiv preprint arXiv:2310.01558*, 2023.

[51] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *International conference on machine learning*. PMLR, 2018, pp. 274–283.

[52] O. Bryniarski, N. Hingun, P. Pachuca, V. Wang, and N. Carlini, "Evading adversarial example detection defenses with orthogonal projected gradient descent," *arXiv preprint arXiv:2106.15023*, 2021.

[53] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM workshop on artificial intelligence and security*, 2017, pp. 3–14.

[54] N. Carlini, "A llm assisted exploitation of ai-guardian," *arXiv preprint arXiv:2307.15008*, 2023.

[55] F. Tramer, N. Carlini, W. Brendel, and A. Madry, "On adaptive attacks to adversarial example defenses," *Advances in neural information processing systems*, vol. 33, pp. 1633–1645, 2020.

[56] G. Izacard, M. Caron, L. Hosseini, S. Riedel, P. Bojanowski, A. Joulin, and E. Grave, "Unsupervised dense information retrieval with contrastive learning," *arXiv preprint arXiv:2112.09118*, 2021.

[57] L. Xiong, C. Xiong, Y. Li, K.-F. Tang, J. Liu, P. Bennett, J. Ahmed, and A. Overwijk, "Approximate nearest neighbor negative contrastive learning for dense text retrieval," *arXiv preprint arXiv:2007.00808*, 2020.

[58] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[59] K. Meng, A. S. Sharma, A. J. Andonian, Y. Belinkov, and D. Bau, "Mass-editing memory in a transformer," in *The Eleventh International Conference on Learning Representations*, 2023.

[60] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[61] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[62] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee *et al.*, "Natural questions: a benchmark for question answering research," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 453–466, 2019.

[63] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning, "Hotpotqa: A dataset for diverse, explainable multi-hop question answering," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2369–2380.

[64] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen *et al.*, "Ms marco: A human generated machine reading comprehension dataset," *arXiv preprint arXiv:1611.09268*, 2016.

[65] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. T. Yih, "Dense passage retrieval for open-domain question answering," in *2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, 2020, pp. 6769–6781.

[66] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[67] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[68] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier *et al.*, "Mistral 7b," *arXiv preprint arXiv:2310.06825*, 2023.

[69] X. Li, Y. Zhang, R. Lou, C. Wu, and J. Wang, "Chain-of-scrutiny: Detecting backdoor attacks for large language models," *arXiv preprint arXiv:2406.05948*, 2024.

[70] Z. Xi, T. Du, C. Li, R. Pang, S. Ji, J. Chen, F. Ma, and T. Wang, "Defending pre-trained language models as few-shot learners against backdoor attacks," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[71] J. Yang, K. Zhou, Y. Li, and Z. Liu, "Generalized out-of-distribution detection: A survey," *International Journal of Computer Vision*, pp. 1–28, 2024.