1 February 2025, Preprint v1 · CC-BY 4.0 Qeios PREPRINT

Research Article

TrustRAG: Enhancing Robustness and Trustworthiness in RAG

Huichi Zhou[1], Zhonghao Zhan[1], Zhenhao Li[1], Hamed Haddadi[1], Emine Yilmaz[2]

1. Imperial College London, United Kingdom; 2. University College London, United Kingdom

Retrieval-Augmented Generation (RAG) systems enhance large language models (LLMs) by integrating external knowledge sources, enabling more accurate and contextually relevant responses tailored to user queries. However, these systems remain vulnerable to corpus poisoning attacks that can significantly degrade LLM performance through the injection of malicious content. To address these challenges, we propose TrustRAG, a robust framework that systematically filters compromised and irrelevant contents before they are retrieved for generation. Our approach implements a two-stage defense mechanism: At the first stage, it employs K-means clustering to identify potential attack patterns in retrieved documents using cosine similarity and ROUGE metrics as guidance, effectively isolating suspicious content. Secondly, it performs a self-assessment which detects malicious documents and resolves discrepancies between the model's internal knowledge and external information. TrustRAG functions as a plug-and-play, training-free module that integrates seamlessly with any language model, whether open or closed-source. In addition, TrustRAG maintains high contextual relevance while strengthening defenses against corpus poisoning attacks. Through extensive experimental validation, we demonstrate that TrustRAG delivers substantial improvements in retrieval accuracy, efficiency, and attack resistance compared to existing approaches across multiple model architectures and datasets. We have made TrustRAG available as open-source software at https://github.com/HuichiZhou/TrustRAG.

Corresponding authors: Huichi Zhou, h.zhou24@imperial.ac.uk; Emine Yilmaz, emine.yilmaz@ucl.ac.uk

1. Introduction

Imagine asking an advanced Large Language Model (LLM) who runs OpenAI and receiving a confidently stated but incorrect name—"Tim Cook." While such misinformation might seem concerning, it represents a broader, more systemic vulnerability in modern AI systems. Retrieval-Augmented Generation (RAG) was developed to enhance LLMs by dynamically retrieving information from external knowledge databases[1][2][3], providing more accurate and up-to-date responses. This approach has been widely adopted in prominent applications including ChatGPT[4], Microsoft Bing Chat[5], Perplexity AI[6], and Google Search AI[7]. However, recent incidents have exposed critical weaknesses in these systems, from inconsistent Google Search AI results[8] to dangerous malicious code injections[9], underscoring the real-world consequences of their vulnerabilities.

At the heart of this problem lies a fundamental challenge: while RAG systems aim to enhance accuracy by connecting LLMs to external knowledge, they remain vulnerable to corpus poisoning attacks that can compromise this very capability of RAG. A growing body of research[10][11][12][13][14] has documented how adversaries can exploit these systems by introducing malicious documents designed to hijack the retrieval process. These attacks are particularly stealthy because they can lead LLMs to generate incorrect or deceptive information with high confidence, effectively undermining the core purpose of RAG systems—to provide more reliable and accurate responses.

This vulnerability of RAG systems is caused by two factors. First, there is a significant amount of noise and even misinformation in the content available on the Internet, which poses challenges to retriever (e.g. search engines) in accurately retrieving desirable knowledge. Second, LLMs suffer from unreliable generation challenges, as they can be misled by incorrect information contained in the context. Recent work has demonstrated how malicious instructions injected into retrieved documents can override original user instructions and mislead LLM to generate their expected information[10] or how query-specific adversarial prompts (adversarial prefix and adversarial suffix) can be optimized to mislead both retrievers and LLMs[12]. For example, PoisonedRAG[13] injects malicious documents into the knowledge base to induce incorrect RAG responses. Additionally, there are real-world examples such as the 'glue on pizza' fiasco in Google Search AI[8]. In another case, a retrieval corruption attack led to a loss of $2.5k when ChatGPT generated code that contained malicious code snippets from a compromised GitHub repository[9]. These RAG failures raise the important question of how to safeguard an RAG pipeline.

To defend against these attacks, prior works have proposed advanced RAG frameworks[15][16][17] that mitigate noisy information through majority-vote mechanisms across retrieved documents and carefully engineered prompts. However, these approaches become ineffective when attackers inject multiple malicious documents that outnumber clean ones[13]. Even in scenarios with less aggressive poisoning, RAG systems often struggle with noisy or irrelevant content, which significantly impacts their ability to generate reliable answers[17][1].

To address these vulnerabilities, we propose TrustRAG, the first defense framework specifically designed to maintain robust and trustworthy responses in scenarios where multiple malicious documents have contaminated the retrieval corpus. Our approach operates in two distinct stages, as illustrated in Figure 1. We reveal that most of attackers[12][13] optimization setup constrains the malicious documents to be tightly clustered in the embedding space. Because the initial malicious documents for a target query are generated using LLM with different temperature settings, leading to inherent semantic similarity. Furthermore, to ensure that the optimized adversarial examples are imperceptible and natural, their embeddings are constrained to remain within a small distance of the original malicious documents. As a result, the malicious documents form a dense and distinct region in the embedding space compared to normal, clean documents, indicating that K-means clustering can effectively identify the malicious group. However, we observed that using similarity metrics filters out some clean documents. To address this, we utilize the ROUGE score[18] to measure the overlap, which effectively preserves the clean documents.

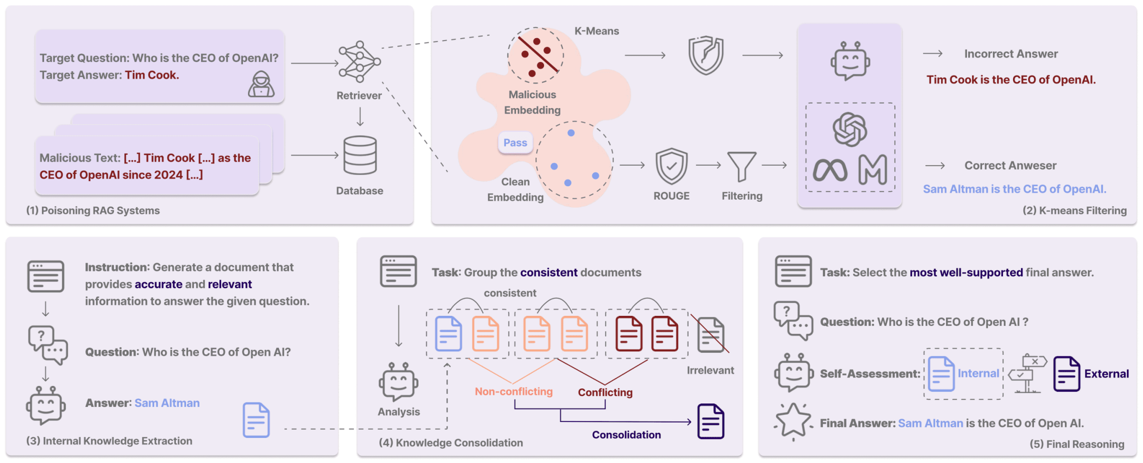
Target Question: Who is the CEO of OpenAI?
Target Answer: Tim Cook.
Retriever
Malicious Text: [...] Tim Cook [...] as the CEO of OpenAI since 2024 [...]
Database
(1) Poisoning RAG Systems
K-Means
Malicious Embedding
Pass
Clean Embedding
ROUGE
Filtering
Incorrect Answer
Tim Cook is the CEO of OpenAI.
Correct Anweser
Sam Altman is the CEO of OpenAI.
(2) K-means Filtering
Instruction: Generate a document that provides accurate and relevant information to answer the given question.
Question: Who is the CEO of Open AI?
Answer: Sam Altman
(3) Internal Knowledge Extraction
Task: Group the consistent documents
consistent
Analysis
Non-conflicting
Conflicting
Irrelevant
Consolidation
(4) Knowledge Consolidation
Task: Select the most well-supported final answer.
Question: Who is the CEO of Open AI ?
Self-Assessment:
Internal
External
Final Answer: Sam Altman is the CEO of Open AI.
(5) Final Reasoning

Figure 1. The TrustRAG framework protects RAG systems from corpus poisoning attacks using a two-stage process. In Stage 1, it (1) identifies malicious documents via K-means clustering and (2) filters malicious content based on embedding distributions. In Stage 2, it (3) extracts internal knowledge to ensure accurate reasoning, (4) resolves conflicts by grouping consistent documents and discarding irrelevant or conflicting ones, and (5) generates a reliable final answer based on self-assessment.

After Clean Retrieval stage, since the majority of malicious documents are filtered out, the original problems could be simplified to the scenario that the clean documents occupies a large portion of the rest of documents.[17] claim that roughly 70% retrieved documents do not directly contain true answers, leading to the impeded performance of LLM with RAG systems. It could be even worse in corpus poisoning attack, because the attacker may induce malicious documents contain wrong answers for a target query. Inspired by the works of [19][20], and [17], the internal knowledge of LLM is beneficial to RAG systems. We leverage LLM itself to combine consistent information, identify conflicting information, and filter out malicious or irrelevant information. Finally, TrustRAG generates answers based on consolidated information and internal knowledge, then self-assesses to determine which should be used for the final response.

We extensively experimented with three datasets NQ, HotpotQA and MS-MARCO, and three open-source and close-source LLMs, Llama-3.1-8B[21], Mistral-Nemo-12B[22] and GPT-4o[4]. Our major contributions are as follows:

- We propose the first defense framework to effectively defend corpus poisoning attack where the number of poisoned documents exceeds the number of clean ones.
- TrustRAG decreases attack success rate while maintains high response accuracy on different popular RAG benchmarks and attack settings.
- We conduct an extensive evaluation for TrustRAG on multiple knowledge databases and LLMs. Additionally, we compare TrustRAG with advanced RAG systems and achieve State-of-the-Art performance.

2. Related Work

Retrieval augmented generation. RAG is a framework for improving the trustworthiness and facticity of LLMs through retrieving relevant information integrated with user query from an external knowledge database and grounding LLMs on the retrieved knowledge for conditional generations[23]. The workflow of RAG involves two steps: retrieval and generation[3][24][25]. With the emergence of LLMs, there is a variety of methods to improve the ability of RAG, such as query rewriter[26][27], retrieval reranking[28] and document summarization.[29][30].

Vulnerability of RAG. The majority of existing RAG attacks focus on compromising retrieval systems with the goal of tricking them into retrieving adversarial documents. These attacks can be divided into following categories: 1) prompt injection attack modifies the text input fed to the LLM directly to cause the LLM to generate outputs that satisfy some adversarial objective[31][32], 2) corpus poisoning attack, attacker adds multiple crafted documents to the database, making the system retrieve adversaries and thus generate incorrect responses to specific queries[12][13][14][33][34][11] and 3) backdoor attack, which introduces optimized backdoor triggers into the LLM's long-term memory or RAG knowledge base, ensuring malicious responses are retrieved when specific triggers are found in inputs[35][36]. These attacks require varying levels of access to the retrievers and/or the LLMs, such as white-box or black-box. However, all of these attacks need access to inject poisoned data into the underlying data corpus used by the RAG system. Additionally, almost all of them are targeted attacks, aimed at a particular subset of data, rather than indiscriminately affecting the entire dataset. In this sense, RAG attacks can essentially be regarded as targeted data poisoning attacks against the retrievers.

Robustness of RAG. To defend against above adversarial attacks on RAG,[15] first propose a defense framework, designing keyword-based and decoding-based algorithms for securely aggregating unstructured text responses.[14] use perplexity-based detection[37], query paraphrasing[38] and increasing context size to defense the adversarial attacks. However, these methods fail to notice the nature of adversarial attack on RAG. The main problems are that, 1) majority-based voting will not work when the majority of the data is poisoned, 2) PPL between malicious and clean documents is not significantly different, and 3) query paraphrasing and increasing context size cannot essentially address the problem of corpus poisoning attacks. To address above issues, we propose TrustRAG to further enhance the robustness of RAG.

3. TrustRAG: Defense Framework

3.1. Problem Formulation

Defense Objective. Our objective is to defense the malicious attacks of RAG systems, filter the malicious and irrelevant documents retrieved by retriever, ultimately producing more accurate and reliable responses from LLMs. Notably, this defense framework is orthogonal to prior work on improving the retriever, LLMs, or conducting adaptive retrieval, which are mainly preliminary steps.

Attack Goals. An attacker selects an arbitrary set of $\mathbf{M}$ questions, denoted as $\mathbf{Q} = [q_1, q_2, \ldots, q_M]$. For each question q_i, the attacker will set an arbitrary attacker-desired response r_i for it. For instance, the q_i could be "Who the the CEO of OpenAI?" and the r_i could be "Tim Cook". In this attack scenario, we formulate corpus poisoning attacks to RAG systems as a constrained optimization problem. We assume an attacker can inject $\mathbf{N}$ malicious documents for each question q_i into a knowledge database $\mathbf{D}$. We use p_i^j to denote the jth malicious document for the question q_i, where $i = 1, 2, \ldots, M$ and $j = 1, 2, \ldots, N$. Attacker's goal is to construct a set of malicious documents $\Gamma = \{p_i^j \big| i = 1, 2 \ldots, M; j = 1, 2, \ldots, N\}$ such that the LLM in a RAG system produce the answer r_i for the question q_i when utilizing the k documents retrieved from the poisoned knowledge database $\mathbf{D} \cup \mathbf{\Gamma}$ as the context. Formally, we have the following optimization problem:

$$\max_{\Gamma} \frac{1}{M} \cdot \sum_{i=1}^{M} \mathbb{I}(LLM(q_i; \mathcal{E}(q_i; \mathcal{D} \cup \Gamma)) = r_i), \quad (1)$$

$$\text{s.t.}, \mathcal{E}(q_i; \mathcal{D} \cup \Gamma) = \text{Retrieve}(q_i, f_q, f_t, \mathcal{D} \cup \Gamma), \quad (2)$$

$$i = 1, 2, \cdots, M, \quad (3)$$

where $\mathbb{I}(\cdot)$ is the indicator function whose output is 1 if the condition is satisfied and 0 otherwise, $\mathcal{E}(q_i; \mathcal{D} \cup \Gamma)$ is a set of k texts retrieved from the corrupted knowledge database $\mathcal{D} \cup \Gamma$ for the target question q_i, and f represents the embedding model for the query and text, respectively. The objective function is large when the answer produced by the LLM based on the k retrieved texts for the target question is the target answer.

3.2. Overview of TrustRAG

TrustRAG is a framework designed to defend against malicious attacks that poison RAG systems. It leverages K-means clustering and collective knowledge from both the internal knowledge of the LLM and external documents retrieved to generate more trustworthy and reliable responses. As shown in Figure 1, attackers optimize malicious documents for a target question and a target answer. The retriever retrieves relevant documents from the knowledge database, and K-means filters out malicious documents. The LLM then generates information about the query from its internal knowledge and compares it with the external knowledge to remove conflicts and irrelevant documents. Finally, the output is generated based on the most reliable knowledge.

3.3. Clean Retrieval – Stage 1

In the *Clean Retrieval* stage, we employ K-means clustering $(k = 2)$ to differentiate between clean and potentially malicious documents based on their embedding distributions. Different from the previous work, they only consider the attack scenario of single injection, TrustRAG is designed to handle both single and multiple injection attacks.

We formally define the attacker's optimization objective as:

$$S = \arg\max_{S'} Sim(f_q(q_i), f_t(S' \oplus I)), \tag{4}$$

where $\mathrm{Sim}(\cdot, \cdot)$ represents the similarity score (e.g., cosine similarity), f_q and f_t are the text encoders for the query and retrieved document respectively. $S \oplus I$ represents the malicious document, attacker's optimize the S to maximize the similarity between the query and the malicious document. I is the context of target answer predefined by the attacker, causing the LLM to generate incorrect answers. Due to the discrete nature of language, the attacker uses HotFlip[39] to optimize S and I.

Here is the second optimization goal for the attacker:

$$I' = \arg\max_{I'} P(LLM(q_i, S \oplus I') = r_i) \geq \eta, \tag{5}$$

where η represents the minimum probability threshold for the LLM to generate the attacker's desired response r_i given the poisoned input.

To ensure the optimized text maintains malicious semantic similarity while still misleading the LLM, the following constraints must be satisfied:

$$|f_t(S \oplus I') - f_t(S \oplus I)|_p \leq \epsilon, \tag{6}$$

where $|\cdot|_p$ represents the L_p norm and ϵ is a small constant controlling the maximum allowed semantic deviation.

The initial text I is generated with different temperature settings to create multiple diverse malicious documents. These texts inherently share high similarity due to their common generation process, and become even more tightly clustered in the embedding space after optimization under the constraint in Eq. 6. To defend against such attacks, we propose a two-stage framework.

K-means Clustering. In the first stage, we apply the K-means clustering algorithm to analyze the distribution of text embeddings generated by f_t and identify suspicious high-density clusters that may indicate the presence of malicious documents. In cases of multiple injections, our first-stage defense strategy effectively filters most malicious groups or pairs due to their high similarity.

N-gram Preservation. In consideration of single injection attacks, we proposed using ROUGE-L score[18] to compare intra-cluster similarity, aiming to preserve the majority of clean document for *Conflict Removal* information consolidation, which robustly filter single malicious document. From Figure 3, it was observed that the ROUGE-L scores significantly differ when comparing pairs of clean documents, pairs of malicious documents, and pairs of clean and malicious documents. Utilizing this property, we can decide not to filter groups containing only one malicious document among clean documents, thereby reducing information loss. Instead, these groups can proceed to Conflict Removal, which focuses on identifying and removing single injection attacks.

3.4 Conflict Removal – Stage 2

In the Remove Conflict stage, we leverage the internal knowledge of the LLM, which reflects the consensus from extensive pre-training and instruction-tuning data. This internal knowledge can supplement any missing information from the limited set of retrieved documents and even rebut malicious documents, enabling mutual confirmation between internal and external knowledge.

Internal Knowledge Extract. After the *Clean Retrieval* stage, where most of the malicious documents have been filtered out, we further enhance the trustworthiness of the RAG system. First, we prompt the LLM to generate internal knowledge (see Appendix D.1), following the work of [40], which emphasizes the importance of reliability and trustworthiness in generated documents. To achieve this goal, we enhance the original method with constitutional principles[40]. However, unlike the works[19][20][17], which generate multiple diverse documents using different temperature settings and may lead to hallucination or incorrectness, we only perform a single LLM inference. This approach is not only more reliable but also cost-efficient.

Knowledge Consolidation. We employ the LLM to explicitly consolidate information from both documents generated from its internal knowledge and documents retrieved from external sources. Initially, we combine document from both internal and external knowledge sources $D_0 = D_{external} \cup I_{internal} \cup \Gamma$. To filter the conflict between clean and malicious documents, we prompt the LLM using prompt (See Appendix D.2) to identify consistent information across different documents, detect malicious information. This step would regroup the unreliable knowledge in input documents into fewer refined documents. The regrouped documents will also attribute their source to the corresponding input documents.

Self-Assessment of Retrieval Correctness. In its final step, TrustRAG prompts the LLM to perform a self-assessment by evaluating its internal knowledge against the retrieved external documents (See Appendix D.3). This process identifies conflicts, consolidates consistent information, and determines the most credible sources, ensuring that the final answer is both accurate and trustworthy. This self-assessment mechanism is key to enhancing the robustness of TrustRAG, enabling it to maintain high accuracy.

4. Experiment

4.1. Setup

In this section, we discuss our experiment setup. All of our inference architectures are implemented by LMDeploy[1].

Datasets. We use three benchmark question-answering datasets in our defense framework: Natural Questions (NQ)[41], HotpotQA[42], and MS-MARCO[43], where each dataset has a knowledge database. The knowledge databases of NQ and HotpotQA are collected from Wikipedia, which contains 2.6M and 5.2M documents, respectively. The knowledge database of MS-MARCO is collected from web documents using the MicroSoft Bing search engine, which contains 8.8M documents.

Attackers. We introduce two kinds of popular RAG attacks to verify the robustness of our defense framework. (1) Corpus Poisoning Attack: PoisonedRAG[13] create poisoned documents by directly appending poisoned text to the adversarial queries. (2) Prompt Injection Attack: PIA[11][10] propose a attack, in which a malicious user generates a small number of adversarial passages by perturbing discrete tokens to maximize similarity with a provided set of training queries.

Evaluation Metrics. Following previous work, we adopt several metrics to evaluate the performance of TrustRAG: (1) Accuracy ($\mathbf{ACC}$) represents the response accuracy of RAG system. (2) Attack Successful Rate ($\mathbf{ASR}$) is the number of incorrect answer generated by the RAG system when misled by attackers.

4.2. Results

We conduct comprehensive experiments compared with different defense frameworks and RAG systems under PIA[11] and PoisonedRAG[13] and evaluate the performance across three LLMs. The more detailed results of PoisonedRAG in different poison rate can be found in the Appendix A (Table 4, Table 5, and Table 6).

Models	Defense	HotpotQA[42]			NQ[41]			MS-MARCO[43]		
		PIA	PoisonedRAG	Clean	PIA	PoisonedRAG	Clean	PIA	PoisonedRAG	Clean
		ACC ↑ / ASR ↓	ACC ↑ / ASR ↓	ACC ↑	ACC ↑ / ASR ↓	ACC ↑ / ASR ↓	ACC ↑	ACC ↑ / ASR ↓	ACC ↑ / ASR ↓	ACC ↑
Mistral Nemo-12B	Vanilla RAG	43.0 /49.0	1.0/97.0	**78.0**	45.0 /50.0	10.0/88.0	69.0	47.0 /49.0	6.0/93.0	82.0
	RobustRAG Keyword	55.0 /25.0	26.0/70.0	54.0	55.0 /**4.0**	28.0/60.0	57.0	75.0 /**6.0**	37.0/53.0	72.0
	InstructRAG ICL	31.0 /64.0	9.0/89.0	73.0	53.0 /41.0	11.0/88.0	65.0	57.0 /37.0	13.0/84.0	83.0
	ASTUTE RAG	59.0 /28.0	30.0/61.0	76.0	62.0 /19.0	44.0/46.0	**72.0**	72.0 /24.0	37.0/59.0	**84.0**
	TrustRAG stage 1	37.0 /51.0	69.0/8.0	74.0	45.0 /43.0	57.0/3.0	66.0	42.0 /54.0	75.0/**6.0**	79.0
	TrustRAG stage 2	**77.0** /**9.0**	**70.0**/**4.0**	77.0	**66.0** /8.0	**64.0**/**1.0**	66.0	**81.0** /9.0	**79.0**/7.0	81.0
Llama 3.1-8B	Vanilla RAG	3.0/95.0	1.0/99.0	**71.0**	4.0/93.0	2.0/98.0	71.0	2.0/98.0	3.0/97.0	79.0
	RobustRAG Keyword	55.0 /4.0	3.0/93.0	52.0	44.0 /11.0	1.0/68.0	45.0	69.0 /15.0	3.0/95.0	72.0
	InstructRAG ICL	64.0 /27.0	26.0/73.0	83.0	55.0 /19.0	27.0/69.0	68.0	57.0 /19.0	44.0/54.0	**89.0**
	ASTUTE RAG	51.0 /28.0	48.0/41.0	65.0	70.0 /14.0	61.0/29.0	75.0	71.0 /25.0	26.0/73.0	83.0
	TrustRAG stage 1	28.0 /61.0	54.0/**6.0**	70.0	40.0 /52.0	67.0/6.0	65.0	31.0 /67.0	77.0/7.0	81.0
	TrustRAG stage 2	**73.0** /**3.0**	**59.0**/**6.0**	**71.0**	**83.0** /**2.0**	**81.0**/**2.0**	**81.0**	**86.0** /**7.0**	**87.0**/**3.0**	88.0
GPT$_{4o}$	Vanilla RAG	60.0 /37.0	8.0/92.0	82.0	52.0 /41.0	20.0/80.0	76.0	67.0 /28.0	29.0/66.0	81.0
	RobustRAG Keyword	60.0 /8.0	5.0/76.0	54.0	40.0 /38.0	1.0/61.0	45.0	48.0 /29.0	2.0/63.0	56.0

Models	Defense	HotpotQA[42]			NQ[41]			MS-MARCO[43]		
		PIA	PoisonedRAG	Clean	PIA	PoisonedRAG	Clean	PIA	PoisonedRAG	Clean
		ACC ↑ / ASR ↓	ACC ↑ / ASR ↓	ACC ↑	ACC ↑ / ASR ↓	ACC ↑ / ASR ↓	ACC ↑	ACC ↑ / ASR ↓	ACC ↑ / ASR ↓	ACC ↑
	InstructRAG ICL	58.0 /41.0	1.0/98.0	**86.0**	63.0 /34.0	13.0/83.0	79.0	69.0 /28.0	15.0/81.0	81.0
	ASTUTE RAG	74.0 /16.0	66.0/35.0	80.0	81.0 /4.0	76.0/24.0	81.0	86.0 /11.0	67.0/24.0	85.0
	TrustRAG stage 1	56.0 /37.0	**82.0**/5.0	76.0	49.0 /41.0	79.0/6.0	76.0	63.0 /35.0	88.0/4.0	77.0
	TrustRAG stage 2	**83.0** /**3.0**	81.0/**3.0**	84.0	**83.0** /**1.0**	**81.0**/**1.0**	**84.0**	**91.0** /**1.0**	**90.0**/**2.0**	**89.0**

Table 1. Main Results show that different defense frameworks and RAG systems defend against two kinds of attack methods based on three kinds of large language models.

As shown in Table 1, all the previous methods fail to effectively handle the scenario of multiple malicious documents injected into the knowledge database, under PoisonedRAG attack, the $\mathbf{ASR}$ can range from 24% to 97% and the $\mathbf{ACC}$ can range from 1% to 76%. It is worth noticing that RobustRAG, which defense framework using aggregating and voting strategies. It fails the number of malicious documents exceed the number of benign one, they failed. However, benefiting from the K-means filtering strategy, TrustRAG significantly reduces malicious documents during retrieval, and only a small portion of malicious documents are used in *Conflict Removal* stage. After *Conflict Removal*, TrustRAG can integrate with internal knowledge, use the information of consistent groups, and self-assess the whether to use the information from the RAG. The results show that TrustRAG can effectively enhance the robustness of RAG systems.

Regarding PIA attack, which will use the strongly induced document to mislead the LLM to generate incorrect answer. We consider this situation as equivalent to the scenario where the number of poisonings is set to 1. Our method can also effectively defense this attack using *Conflict Removal*, outperforming previous work by a large margin.

5. Detailed Analysis of TrustRAG

Dataset	Embedding Model	Poison-(100%)	Poison-(80%)	Poison-(60%)	Poison-(40%)	Poison-(20%)	Poison-(0%)
		F1↑	F1↑ / CRR↑	F1↑ / CRR↑	F1↑ / CRR↑	F1↑ / CRR↑	CRR↑
NQ	SimCSE	97.5	92.6/92.0	94.3/91.5	84.9/89.0	3.1/86.2	85.0
	SimCSE w/o ROUGE	91.3	83.9/93.0	72.0/69.0	64.4/68.3	35.8/54.8	52.6
	Bert	97.2	84.7/84.0	87.4/89.0	77.8/82.0	5.6/78.5	74.2
	$\text{Bert}_{\text{w/o ROUGE}}$	52.0	73.2/80.0	63.4/61.0	51.7/58.0	35.5/55.8	52.0
	BGE	98.1	90.8/92.0	96.9/93.0	89.5/91.0	3.0/86.3	87.6
	$\text{BGE}_{\text{w/o ROUGE}}$	93.8	85.0/93.0	86.7/83.0	79.9/80.7	27.5/51.5	51.4
MS-MARCO	SimCSE	95.6	84.7/88.0	84.0/80.0	71.7/73.0	4.6/72.0	70.6
	SimCSE w/o ROUGE	89.4	77.3/84.0	69.6/60.5	58.1/61.7	17.0/47.5	52.4
	Bert	95.2	83.0/85.0	77.8/73.0	66.8/71.7	5.8/70.0	70.4
	$\text{Bert}_{\text{w/o ROUGE}}$	87.4	75.4/74.0	67.3/58.5	48.9/53.7	24.6/48.0	51.8
	BGE	94.2	87.2/88.0	84.1/73.0	73.4/69.3	5.0/66.0	66.8
	$\text{BGE}_{\text{w/o ROUGE}}$	91.4	81.5/78.0	73.2/59.0	64.9/66.3	17.9/46.5	47.8
HotpotQA	SimCSE	99.2	95.6/91.0	95.2/84.0	90.0/80.6	6.5/80.0	81.8
	SimCSE w/o ROUGE	94.9	85.1/94.0	1.0/77.0	72.5/73.3	21.3/47.5	49.0
	Bert	99.2	89.7/88.0	85.5/75.5	83.7/79.7	2.4/78.0	76.2
	$\text{Bert}_{\text{w/o ROUGE}}$	88.5	79.4/88.0	64.1/61.0	48.1/55.3	25.8/49.5	49.0
	BGE	99.6	91.9/90.0	95.6/84.0	90.2/82.3	9.7/80.5	81.0
	$\text{BGE}_{\text{w/o ROUGE}}$	94.7	87.4/91.0	82.5/81.0	74.7/76.3	16.5/44.3	49.6

Table 2. Results on various datasets with different Poisoning levels and embedding models. F1 score measures the performance of detecting poisoned samples, while Clean Retention Rate (CRR) evaluates the proportion of clean samples retained after filtering.

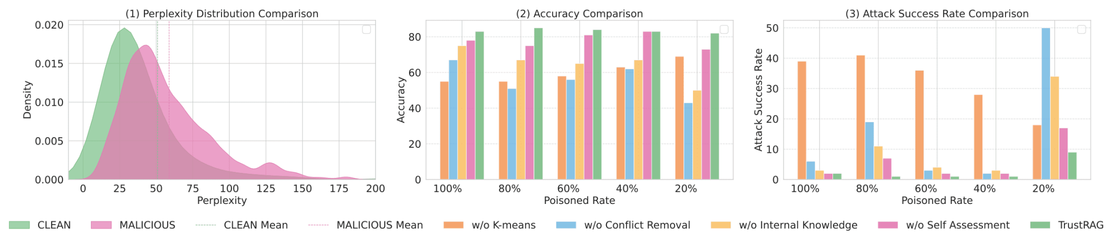

Figure 2. (1) The perplexity distribution density plot between clean and malicious documents. And the lines of dashes represent the average perplexity values. (2) The bar plot of ablation study on accuracy in NQ dataset based on the Llama$_{3.1\text{-}8B}$. (3) The bar plot of ablation study on attack success rate in NQ dataset based on the Llama$_{3.1\text{-}8B}$.

	# API Call	MS-MARCO	NQ	HotpotQA
Vanilla RAG	1	8.9/1×	9.2/1×	9.6/1×
InstructRAG$_{\text{ICL}}$	1	12.6/1.4×	13.1/1.4×	32.7/3.4×
RobustRAG$_{\text{Keyword}}$	11	107.9/12.1×	107.7 /11.7×	107.9/11.2×
ASTUTE RAG	3	17.5/2.0×	17.3/1.9×	16.7/1.7×
TrustRAG$_{\text{K-means}}$	1	12.3/1.4×	12.6/1.4×	12.5/1.3×
TrustRAG$_{\text{Conflict}}$	3	18.4/2.1×	19.9/2.2×	21.7/2.3×

Table 3. TrustRAG runtime analysis based on Llama$_{3.1\text{-}8B}$ for 100 queries in three different datasets.

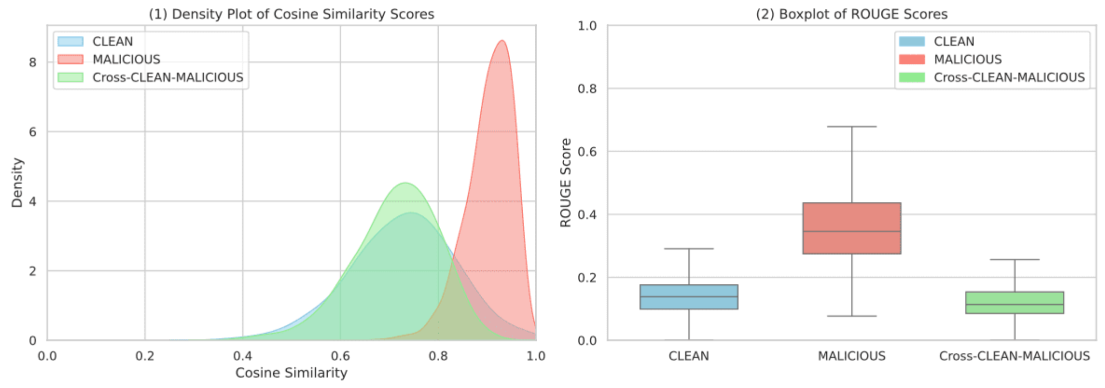

Figure 3. (1) The density plot of cosine similarity between three different groups. (2) The box plot of ROUGE Score between three different groups.

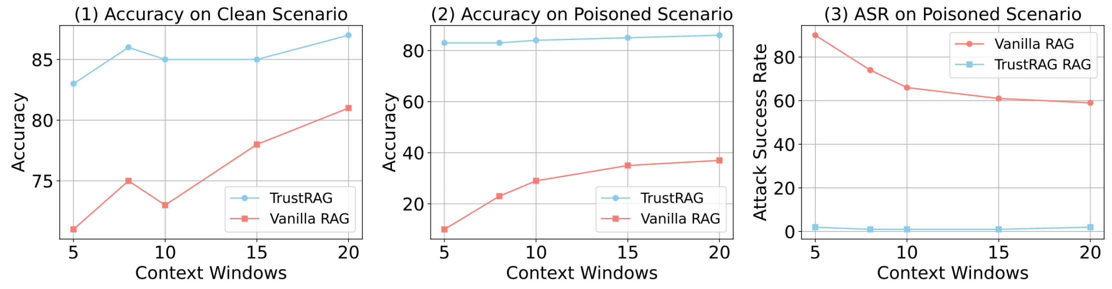

Figure 4. (1) The line plot of accuracy between TrustRAG and Vanilla RAG on clean scenario. (2) The line plot of accuracy between TrustRAG and Vanilla RAG on malicious scenario. (3) The line plot of attack successful rate between TrustRAG and Vanilla RAG on malicious scenario. All the context windows set from 5 to 20 and the malicious scenario includes 5 malicious documents.

5.1. Effectiveness of K-means Filtering Strategy

Distribution of Poisoned Documents. As shown in Appendix Figure 5, we plot a case in which samples from the NQ data set are used in different numbers of poisoned documents, we can see that in the scenario of multiple malicious documents, the malicious documents are close to each other. By contrast, for a single poisoned document, it will be mixed in the clean documents. Therefore, it is important to use the n-gram preservation to preserve the clean documents.

N-gram preservation. As shown in Table 2, we conduct an ablation study on n-gram preservation, we can see that When poinsoning rate exceeds 20%, the F1 score is higher after applying n-gram preservation in the clean retrieval stage. However, when poisoning rate at 20%, without n-gram preservation, the K-means filtering strategy will randomly remove the group which has higher similarity, but it will lead the bad effect of decreasing the CRR. The clean documents can thus be filtered by mistake. Therefore, using n-gram preservation will not only preserve the clean documents but increase the F1 score of detecting the malicious documents.

Embedding Models. Choosing the right embedding model is crucial for effectively cluster the retrieved documents. As shown in Table 2, we compare different embedding models: SimCSE[44], Bert[45] and BGE[46], and the results show that our proposed K-means filtering strategy is robust and effective for all three embedding models. In addition, we notice that more fine-grained embedding model (e.g. SimCSE) can achieve better performance and be more robust in different poisoning rates and datasets.

5.2. Runtime Analysis

In Table 3, we present a detailed runtime analysis for various methods across three datasets. The analysis reveals that TrustRAG spends approximately twice inference time as compared to Vanilla RAG, which is a reasonable trade-off considering the significant improvements in robustness and reliability offered by TrustRAG.

5.3. Effectiveness of Perplexity-based Detection

Since attackers will generate unnatural looking patterns to attack LLMs, PPL detection has been suggested as a defense[47][38] and [14] claim that the distribution of perplexity values differ significantly between clean and malicious documents and PPL can be an effective defense. We test the effectiveness of PPL defense and follow the setting in in the work[14]. Text perplexity[37] is used to evaluate the naturalness and quality of text.

As shown in Figure 2(1), the PPL values for clean and adversarial texts overlap significantly. While [14] argue that these distributions differ substantially, our findings challenge this claim. Although some adversarial examples exhibit higher PPL values, many fall within the range of clean texts. This overlap highlights the limitations of relying solely on PPL as a detection metric, as it risks false negatives (misclassifying adversarial texts as clean) and false positives (flagging clean texts as adversarial).

5.4. Ablation Study

We conducted an ablation study on Llama$_{3.1\text{-}8B}$ and analyzed the impact of four key components. The detailed experiments are in Appendix Table 8.

Impact of K-means Clustering. As shown in Figure 2 (2) and (3), K-means filtering effectively defends against attacks while maintaining high response accuracy when the poisoning rate exceeds 20%. Even at a poisoning rate of 20% (with only a single poisoned document), it still successfully preserves successfully preserves the integrity of clean documents.

Impact of Internal Knowledge. Comparing TrustRAG w/ and w/o providing internal knowledge inferred from the LLM, from Figure 2, we observe notable improvements in both ACC and ASR from utilizing the LLM internal knowledge. Particularly at the poisoned rate of 20%, internal knowledge effectively addresses conflicts between malicious and clean documents, contributing significantly to improved robustness.

Impact of Conflict Removal. While K-means clustering and internal knowledge significantly reduce the ASR, the conflict removal component also plays a crucial role in the defense framework. By leveraging knowledge consolidation and rationale outputs, TrustRAG further enhances the robustness of RAG systems across all scenarios under different poison percentage.

Impact of Self-Assessment. The self-assessment mechanism can further enhance the performance of TrustRAG in all settings, particularly at a poisoned rate of 20%. This suggests that the LLM can effectively distinguish between inductive or malicious information and the internal and external knowledge.

5.5. Impact of Top-K Context Window

Furthermore, RAG systems may face another two critical types of non-adversarial noise beyond intentional poisoning attacks: retrieval-based noise from imperfect retrievers returning irrelevant documents, and corpus-based noise from inherent inaccuracies in the knowledge base itself[16]. To rigorously assess TrustRAG's robustness, we conducted extensive experiments on the NQ dataset using Llama$_{3.1\text{-}8B}$ under two key scenarios: (1) a clean setting with context windows ranging from 1 to 20 documents, and (2) a poisoned setting with 5 malicious documents and varying context windows. The results reveal TrustRAG's superior performance in both scenarios. In clean settings, TrustRAG's accuracy improves steadily with larger context windows ($5 - 20$ documents), consistently outperforming vanilla RAG. More importantly, in poisoned scenarios, TrustRAG maintains approximately 80% accuracy while keeping attack success rates (ASR) around 1%. This contrasts markedly with vanilla RAG, which achieves only $10 - 40\%$ accuracy in relation to ASR levels of $60 - 90\%$.

5.6. Evaluation in Real-World Adversarial Conditions

To assess TrustRAG's performance under real-world adversarial conditions, we utilize the RedditQA dataset from [48], which comprises Reddit posts with real-world factual errors that result in incorrect answers to corresponding questions. We evaluate our method on this dataset using Llama$_{3.1\text{-}8B}$. The vanilla RAG with the retrieved documents achieve the response accuracy of 27.3% with an attack success rate of 43.8%. In contrast, TrustRAG achieves a response accuracy of 72.2% with an attack success rate of 11.9%, demonstrating its robustness in real-world adversarial conditions.

6. Conclusion

RAG systems, despite their potential to enhance language models' capabilities, remain vulnerable to corpus poisoning attacks which is a critical security concern that is still insufficiently addressed. In this work, we introduce TrustRAG, the first RAG defense framework designed to counter attacks involving multiple maliciously injected documents. TrustRAG employs K-means filtering to reduce the presence of malicious documents and incorporates both internal and external knowledge sources to resolve conflicts and mitigate the impact of these attacks. Our comprehensive evaluation across benchmark datasets demonstrates that TrustRAG outperforms existing defenses, maintaining high accuracy even under aggressive poisoning scenarios where traditional approaches fail.

Appendix A. Details of Experiments

A.1. NQ results

As shown in Table 4, the experimental results highlight the robustness of various RAG defenses against corpus poisoning attacks across different poisoning levels, evaluated on three language models: $\text{Mistral}_{\text{Nemo-12B}}$, $\text{Llama}_{3.1\text{-}8B}$, and GPT_{4o}. For the $\text{Mistral}_{\text{Nemo-12B}}$, the $\text{TrustRAG}_{\text{stage 2}}$ defense achieved a notable accuracy of 64.0% with a minimal ASR of 1.0% at a 100% poisoning rate, maintaining superior performance even under extreme adversarial scenarios. Similarly, at a lower poisoning rate of 20%, $\text{TrustRAG}_{\text{stage 2}}$ continued to lead with 67.0% accuracy and an ASR of only 11.0%.

For the $\text{Llama}_{3.1\text{-}8B}$, $\text{TrustRAG}_{\text{stage 2}}$ showcased impressive resilience, achieving 83.0% accuracy and an ASR of just 2.0% under a 100% poisoning rate. At a moderate poisoning rate of 40%, the accuracy remained high at 83.0% with an ASR of 1.0%, significantly outperforming alternative defenses such as ASTUTE RAG and $\text{RobustRAG}_{\text{Keyword}}$.

The GPT_{4o} model further validated the effectiveness of $\text{TrustRAG}_{\text{stage 2}}$, achieving an accuracy of 81.0% and a near-zero ASR of 1.0% at 100% poisoning. Even under a 60% poisoning rate, the method maintained robust performance with 80.0% accuracy and an ASR of 1.0%, demonstrating its ability to consistently suppress adversarial effects while preserving response reliability across diverse settings and language models. These results confirm TrustRAG's state-of-the-art capability in defending against both high and low-intensity poisoning attacks.

Dataset	Defense	Poison-(100%)	Poison-(80%)	Poison-(60%)	Poison-(40%)	Poison-(20%)	Poison-(0%)
		ACC↑ / ASR↓	ACC↑ / ASR ↓	ACC↑ / ASR ↓	ACC↑ / ASR↓	ACC↑ / ASR ↓	ACC↑
Mistral Nemo-12B	Vanilla RAG	10.0/88.0	15.0/84.0	18.0/79.0	34.0/62.0	42.0/51.0	69.0
	RobustRAG Keyword	28.0/60.0	30.0/59.0	35.0/54.0	39.0/45.0	54.0/**9.0**	57.0
	InstructRAG$_{ICL}$	11.0/88.0	21.0/77.0	25.0/70.0	33.0/59.0	48.0/40.0	65.0
	ASTUTE RAG	44.0/46.0	56.0/32.0	63.0/24.0	**65.0**/19.0	**69.0**/10.0	**72.0**
	TrustRAG$_{stage\ 1}$	57.0/3.0	51.0/18.0	**65.0/2.0**	62.0/2.0	46.0/40.0	66.0
	TrustRAG$_{stage\ 2}$	**64.0/1.0**	**64.0/2.0**	63.0/**2.0**	**65.0/1.0**	67.0/11.0	69.0
Llama$_{3.1\text{-}8B}$	Vanilla RAG	2.0/98.0	2.0/98.0	3.0/97.0	4.0/93.0	26.0/73.0	71.0
	RobustRAG Keyword	11.0/83.0	15.0/75.0	23.0/63.0	37.0/46.0	51.0/27.0	61.0
	InstructRAG$_{ICL}$	27.0/69.0	38.0/56.0	40.0/56.0	51.0/45.0	58.0/37.0	68.0
	ASTUTE RAG	61.0/29.0	64.0/24.0	68.0/19.0	69.0/18.0	77.0/11.0	75.0
	TrustRAG$_{stage\ 1}$	67.0/6.0	51.0/19.0	56.0/3.0	62.0/2.0	43.0/50.0	65.0
	TrustRAG$_{stage\ 2}$	**83.0/2.0**	**85.0/1.0**	**84.0/1.0**	**83.0/1.0**	**82.0/9.0**	**82.0**
GPT$_{4o}$	Vanilla RAG	20.0/80.0	32.0/69.0	37.0/60.0	49.0/49.0	56.0/39.0	76.0
	RobustRAG Keyword	1.0/61.0	8.0/57.0	20.0/58.0	32.0/36.0	39.0/28.0	45.0
	InstructRAG$_{ICL}$	13.0/83.0	21.0/74.0	27.0/65.0	37.0/55.0	53.0/39.0	79.0
	ASTUTE RAG	76.0/24.0	76.0/21.0	76.0/20.0	78.0/16.0	82.0/6.0	81.0
	TrustRAG$_{stage\ 1}$	79.0/6.0	65.0/15.0	75.0/3.0	73.0/3.0	57.0/35.0	76.0
	TrustRAG$_{stage\ 2}$	**81.0/1.0**	**82.0/3.0**	**80.0/1.0**	**84.0/1.0**	**83.0/4.0**	**84.0**

Table 4. NQ Result

A.2. MS-MARCO results

The results presented in Table 5 evaluate the robustness of different RAG defenses against corpus poisoning attacks on the MS-MARCO dataset using three language models: $\text{Mistral}_{\text{Nemo-12B}}$, $\text{Llama}_{3.1\text{-}8\text{B}}$, and $\text{GPT}_{4\text{o}}$. Across various poisoning rates, $\text{TrustRAG}_{\text{stage 2}}$ consistently demonstrates superior performance in maintaining high accuracy and suppressing ASR.

For the $\text{Mistral}_{\text{Nemo-12B}}$, $\text{TrustRAG}_{\text{stage 2}}$ achieves an accuracy of 85.0% and an ASR of just 4.0% at a 100% poisoning rate. Even as the poisoning rate decreases to 60%, $\text{TrustRAG}_{\text{stage 2}}$ maintains robust performance with 83.0% accuracy and an ASR of 5.0%. At the 20% poisoning level, accuracy remains high at 84.0% with a moderate ASR of 12.0%.

For the $\text{Llama}_{3.1\text{-}8\text{B}}$, $\text{TrustRAG}_{\text{stage 2}}$ delivers excellent robustness under all conditions. At a 100% poisoning rate, it achieves an accuracy of 87.0% with a low ASR of 5.0%. At lower poisoning rates, such as 40% and 20%, $\text{TrustRAG}_{\text{stage 2}}$ achieves 85.0% and 83.0% accuracy, respectively, while maintaining ASR levels at or below 11.0%.

For the $\text{GPT}_{4\text{o}}$ model, $\text{TrustRAG}_{\text{stage 2}}$ again leads in robustness. At a 100% poisoning rate, it reaches an impressive 90.0% accuracy with an ASR of only 2.0%. Even with lower poisoning rates, such as 60% and 40%, the method maintains high accuracy of 90.0% and87.0%, respectively, and keeps ASR at minimal levels (2.0% and 4.0%, respectively).

Dataset	Defense	Poison-(100%)	Poison-(80%)	Poison-(60%)	Poison-(40%)	Poison-(20%)	Poison-(0%)
		ACC ↑ / ASR ↓	ACC ↑ / ASR ↓	ACC ↑ / ASR ↓	ACC ↑ / ASR ↓	ACC ↑ / ASR ↓	ACC ↑
Mistral Nemo-12B	Vanilla RAG	6.0 / 93.0	10.0 / 88.0	21.0 / 74.0	33.0 / 62.0	52.0 / 43.0	82.0
	RobustRAG Keyword	37.0/53.0	40.0/50.0	50.0/38.0	62.0/21.0	72.0/**9.0**	72.0
	InstructRAG	13.0/84.0	22.0/72.0	31.0/62.0	40.0/54.0	57.0/36.0	83.0
	ASTUTE RAG	37.0/59.0	43.0/52.0	55.0/41.0	68.0/28.0	77.0/16.0	**84.0**
	TrustRAG$_{\text{stage 1}}$	75.0/6.0	67.0/18.0	75.0/7.0	79.0/7.0	50.0/44.0	79.0
	TrustRAG$_{\text{stage 2}}$	**85.0/4.0**	**84.0/6.0**	**83.0/5.0**	**82.0/6.0**	**84.0**/12.0	82.0
Llama$_{3.1\text{-}8B}$	Vanilla RAG	3.0/97.0	3.0/96.0	5.0/94.0	7.0/93.0	28.0/70.0	79.0
	RobustRAG Keyword	25.0/68.0	28.0/66.0	37.0/54.0	57.0/34.0	67.0/19.0	73.0
	InstructRAG	44.0/54.0	47.0/51.0	49.0/45.0	60.0/36.0	63.0/33.0	**89.0**
	ASTUTE RAG	26.0/73.0	40.0/57.0	50.0/47.0	52.0/44.0	54.0/41.0	83.0
	TrustRAG$_{\text{stage 1}}$	77.0/7.0	64.0/18.0	72.0/**7.0**	78.0/**6.0**	45.0/47.0	81.0
	TrustRAG$_{\text{stage 2}}$	**87.0/5.0**	**84.0/8.0**	**85.0/7.0**	**85.0**/7.0	**83.0/11.0**	85.0
GPT$_{4o}$	Vanilla RAG	29.0/66.0	43.0/49.0	51.0/40.0	59.0/35.0	67.0/24.0	81.0
	RobustRAG Keyword	2.0/63.0	17.0/52.0	23.0/48.0	41.0/33.0	50.0/22.0	56.0
	InstructRAG	15.0/81.0	31.0/64.0	39.0/54.0	47.0/45.0	59.0/35.0	81.0
	ASTUTE RAG	67.0/24.0	67.0/21.0	72.0/17.0	74.0/16.0	77.0/13.0	85.0
	TrustRAG$_{\text{stage 1}}$	88.0/4.0	76.0/11.0	84.0/**2.0**	84.0/**4.0**	62.0/24.0	77.0
	TrustRAG$_{\text{stage 2}}$	**90.0/2.0**	**90.0/2.0**	**90.0**/5.0	**87.0/4.0**	**86.0/8.0**	**89.0**

Table 5. MS-MARCO Result

A.3. HotpotQA results

The results in Table 6 evaluate the robustness of various RAG defenses against corpus poisoning attacks on the HotpotQA dataset using three language models: $\text{Mistral}_{\text{Nemo-12B}}$, $\text{Llama}_{\text{3.1-8B}}$, and GPT_{4o}. The defenses compared include Vanilla RAG, $\text{RobustRAG}_{\text{Keyword}}$, $\text{InstructRAG}_{\text{ICL}}$, ASTUTE RAG, $\text{TrustRAG}_{\text{stage 1}}$, and $\text{TrustRAG}_{\text{stage 2}}$. $\text{TrustRAG}_{\text{stage 2}}$ consistently exhibits superior performance, maintaining high ACC while minimizing the ASR under all poisoning levels.

For the $\text{Mistral}_{\text{Nemo-12B}}$ model, $\text{TrustRAG}_{\text{stage 2}}$ achieves an impressive 75.0% accuracy and 4.0% ASR at a 100% poisoning rate, significantly outperforming other defenses. Even as the poisoning rate decreases to 40%, $\text{TrustRAG}_{\text{stage 2}}$ maintains a high accuracy of 78.0% with a reduced ASR of 3.0%, showcasing its robustness under various poisoning intensities.

For the $\text{Llama}_{\text{3.1-8B}}$, $\text{TrustRAG}_{\text{stage 2}}$ demonstrates strong resilience with 67.0% accuracy and 4.0% ASR at a 100% poisoning rate. As the poisoning rate decreases to 20%, accuracy improves to 70.0%, with the ASR remaining low at 4.0%, underscoring its reliability and effectiveness in defending against adversarial attacks.

For the GPT_{4o}, $\text{TrustRAG}_{\text{stage 2}}$ achieves exceptional performance, reaching 82.0% accuracy and 5.0% ASR at a 100% poisoning rate. At lower poisoning levels, such as 20%, it achieves a remarkable 84.0% accuracy and an ASR of only 1.0%, demonstrating state-of-the-art robustness and adaptability across all attack levels.

Dataset	Defense	Poison-(100%)	Poison-(80%)	Poison-(60%)	Poison-(40%)	Poison-(20%)	Poison-(0%)
		ACC↑ / ASR↓	ACC↑ / ASR↓	ACC↑ / ASR↓	ACC↑ / ASR↓	ACC↑ / ASR↓	ACC↑
Mistral Nemo-12B	Vanilla RAG	1.0/97.0	6.0/93.0	9.0/90.0	18.0/78.0	28.0/68.0	**78.0**
	RobustRAG Keyword	26.0/70.0	28.0/68.0	33.0/59.0	41.0/43.0	51.0/27.0	54.0
	InstructRAG$_{ICL}$	9.0/89.0	11.0/87.0	14.0/81.0	24.0/68.0	36.0/59.0	73.0
	ASTUTE RAG	30.0/61.0	37.0/54.0	52.0/38.0	57.0/31.0	60.0/24.0	76.0
	TrustRAG$_{stage\ 1}$	69.0/8.0	68.0/12.0	76.0/6.0	77.0/5.0	38.0/54.0	74.0
	TrustRAG$_{stage\ 2}$	**75.0/4.0**	**79.0/4.0**	**79.0/4.0**	**78.0/3.0**	**74.0/13.0**	**78.0**
Llama$_{3.1\text{-}8B}$	Vanilla RAG	1.0/99.0	2.0/97.0	6.0/94.0	5.0/94.0	27.0/81.0	71.0
	RobustRAG Keyword	8.0/89.0	10.0/87.0	19.0/76.0	33.0/57.0	40.0/50.0	54.0
	InstructRAG$_{ICL}$	26.0/73.0	40.0/57.0	50.0/47.0	52.0/44.0	54.0/41.0	**83.0**
	ASTUTE RAG	48.0/41.0	53.0/38.0	59.0/30.0	59.0/31.0	65.0/**16.0**	65.0
	TrustRAG$_{stage\ 1}$	54.0/6.0	61.0/12.0	72.0/**3.0**	66.0/**2.0**	43.0/47.0	70.0
	TrustRAG$_{stage\ 2}$	**67.0/4.0**	**71.0/4.0**	**70.0**/7.0	**69.0**/5.0	**66.0**/18.0	74.0
GPT$_{4o}$	Vanilla RAG	8.0/92.0	33.0/67.0	31.0/69.0	48.0/52.0	52.0/48.0	82.0
	RobustRAG Keyword	5.0/76.0	18.0/74.0	20.0/61.0	41.0/43.0	51.0/27.0	54.0
	InstructRAG$_{ICL}$	1.0/98.0	9.0/90.0	19.0/79.0	27.0/71.0	33.0/63.0	**86.0**
	ASTUTE RAG	66.0/35.0	67.0/33.0	74.0/25.0	76.0/24.0	78.0/22.0	80.0
	TrustRAG$_{stage\ 1}$	**82.0**/5.0	77.0/12.0	85.0/5.0	**81.0**/10.0	54.0/46.0	76.0
	TrustRAG$_{stage\ 2}$	81.0/**3.0**	**84.0/1.0**	**81.0/3.0**	**81.0/4.0**	**84.0/6.0**	84.0

Table 6. HotpotQA Result

Appendix B. Scaling Law for TrustRAG

Table 7 demonstrates the scaling behavior across four Llama model sizes (1B, 3B, 8B, and 70B) on three datasets (NQ, HotpotQA, and MS-MARCO) under varying poisoning ratios. For example, on the NQ dataset, the Llama$_{3.2\text{-}1B}$ model achieves an ACC of around 55.0% when subjected to 100% poisoned documents, whereas the larger Llama$_{3.1\text{-}8B}$ sustains an ACC of approximately 83.0% under the same extreme poisoning condition, coupled with a notably low attack success rate (ASR) of 2.0%. A similar trend is observed in HotpotQA: the smallest model registers an ACC of only 41.0 with an ASR of 13.0% at 100% poisoning, yet the Llama$_{3.3\text{-}70B}$ model attains an ACC of 81.0%while limiting ASR to 1.0%. The MS-MARCO results further reinforce this pattern, as the 70B variant retains close to 90.0% ACC even under high-poison scenarios. Overall, these findings highlight a robust scaling law: larger models provide both higher accuracy and greater resilience against poisoning attacks, offering stronger capacity to detect and disregard malicious documents without compromising retrieval or generation quality.

Dataset	Defense	Poison-(100%)	Poison-(80%)	Poison-(60%)	Poison-(40%)	Poison-(20%)	Poison-(0%)
		ACC↑ / ASR↓	ACC↑ / ASR↓	ACC↑ / ASR↓	ACC↑ / ASR↓	ACC↑ / ASR↓	ACC↑
NQ	Llama$_{3.2\text{-}1B}$	55.0/1.0	60.0/6.0	56.0/4.0	58.0/4.0	59.0/8.0	60.0
	Llama$_{3.2\text{-}3B}$	76.0/4.0	75.0/6.0	75.0/2.0	78.0/1.0	78.0/13.0	77.0
	Llama$_{3.1\text{-}8B}$	**83.0**/2.0	**85.0**/**1.0**	**84.0**/**1.0**	**83.0**/**1.0**	**82.0**/9.0	**82.0**
	Llama$_{3.3\text{-}70B}$	78.0/**0.0**	80.0/3.0	78.0/**1.0**	79.0/**1.0**	78.0/**5.0**	76.0
HotpotQA	Llama$_{3.2\text{-}1B}$	41.0/13.0	48.0/15.0	53.0/10.0	53.0/10.0	46.0/23.0	58.0
	Llama$_{3.2\text{-}3B}$	57.0/5.0	57.0/8.0	67.0/5.0	60.0/6.0	54.0/28.0	66.0
	Llama$_{3.1\text{-}8B}$	67.0/4.0	71.0/**4.0**	70.0/7.0	69.0/5.0	66.0/18.0	74.0
	Llama$_{3.3\text{-}70B}$	**81.0**/**1.0**	**74.0**/8.0	**78.0**/**2.0**	**79.0**/**4.0**	**70.0**/**15.0**	**77.0**
MS-MARCO	Llama$_{3.2\text{-}1B}$	50.0/23.0	54.0/21.0	56.0/21.0	54.0/21.0	51.0/26.0	50.0
	Llama$_{3.2\text{-}3B}$	71.0/12.0	71.0/12.0	74.0/12.0	76.0/11.0	73.0/20.0	78.0
	Llama$_{3.1\text{-}8B}$	87.0/5.0	84.0/8.0	85.0/7.0	85.0/7.0	83.0/11.0	85.0
	Llama$_{3.3\text{-}70B}$	**89.0**/**4.0**	**87.0**/**3.0**	**86.0**/**4.0**	**88.0**/**4.0**	**87.0**/**9.0**	**87.0**

Table 7. Scaling-up Llama to larger models. We evaluate the performance of Llama across three datasets with different model sizes. We observe that Llama scales well with the model size, achieving better performance with larger models.

Appendix C. Ablation Studies

Table 8 presents the results of the ablation study for the Llama$_{3.1\text{-}8B}$ model across three datasets (NQ, HotpotQA, and MS-MARCO). Notably, TrustRAG with all components consistently achieved the best performance in most scenarios. Each component serves a distinct role; for instance, by comparing TrustRAG$_{\text{w/o K-Means}}$ and TrustRAG$_{\text{w/o Conflict Removal}}$, we observe that the first stage of the model effectively reduces the number of malicious texts in the retrieved documents, significantly lowering ASR values. Meanwhile, the Internal Knowledge and Self Assessment components, particularly under a 20% poisoning rate, enhance the model's robustness by consolidating external information into a reliable final answer. Without Internal Knowledge, the performance of the $\text{Llama}_{3.1\text{-}8B}$ model declines sharply at a 20% poisoning rate across all three datasets. When relying solely on external information, the model often struggles to generate correct answers, as the consolidated information may still contain malicious or misleading texts. By incorporating Internal Knowledge, the model is better equipped to select documents that align with its internal understanding, thereby improving certainty and resistance to attacks. Furthermore, the addition of Self Assessment allows the model to discard unreliable or untrustworthy documents, further reducing ASR values and improving overall performance. With all the components combined, our TrustRAG can filter malicious content from externally retrieved documents, consolidate both internal and external information, and provide accurate and trustworthy answers.

Dataset	Defense	Poison-(100%)	Poison-(80%)	Poison-(60%)	Poison-(40%)	Poison-(20%)	Poison-(0%)
		ACC↑ / ASR↓	ACC↑ / ASR↓	ACC↑ / ASR↓	ACC↑ / ASR↓	ACC↑ / ASR↓	ACC↑
NQ	Vanilla RAG	2.0/98.0	2.0/98.0	3.0/97.0	4.0/93.0	26.0/73.0	71.0
	TrustRAG$_{\text{w/o K-Means}}$	55.0/39.0	55.0/41.0	58.0/36.0	63.0/28.0	69.0/18.0	81.0
	TrustRAG w/o Conflict Removal	67.0/6.0	51.0/19.0	56.0/3.0	62.0/2.0	43.0/50.0	65.0
	TrustRAG w/o Internal Knowledge	75.0/3.0	67.0/11.0	65.0/4.0	67.0/3.0	50.0/34.0	64.0
	TrustRAG w/o Self Assessment	78.0/**2.0**	75.0/7.0	80.0/2.0	80.0/2.0	69.0/21.0	78.0
	TrustRAG	**83.0/2.0**	**85.0/1.0**	**84.0/1.0**	**83.0/1.0**	**82.0/9.0**	82.0
HotpotQA	Vanilla RAG	1.0/99.0	2.0/97.0	6.0/94.0	5.0/94.0	27.0/81.0	71.0
	TrustRAG$_{\text{w/o K-Means}}$	41.0/56.0	42.0/57.0	49.0/48.0	53.0/44.0	61.0/34.0	73.0
	TrustRAG w/o Conflict Removal	54.0/6.0	61.0/12.0	**72.0/3.0**	66.0/**2.0**	43.0/47.0	81.0
	TrustRAG w/o Internal Knowledge	**69.0**/6.0	61.0/10.0	67.0/6.0	**72.0**/8.0	51.0/32.0	67.0
	TrustRAG w/o Self Assessment	64.0/5.0	59.0/8.0	65.0/6.0	67.0/5.0	55.0/32.0	65.0
	TrustRAG	67.0/**4.0**	**71.0/4.0**	70.0/7.0	69.0/5.0	**66.0/18.0**	74.0
MS-MARCO	Vanilla RAG	3.0/97.0	3.0/96.0	5.0/94.0	7.0/93.0	28.0/70.0	79.0
	TrustRAG$_{\text{w/o K-Means}}$	51.0/47.0	53.0/44.0	53.0/42.0	64.0/32.0	83.0/11.0	86.0
	TrustRAG w/o Conflict Removal	77.0/7.0	64.0/18.0	72.0/7.0	78.0/6.0	45.0/47.0	70.0
	TrustRAG w/o Internal Knowledge	80.0/6.0	75.0/13.0	73.0/10.0	69.0/12.0	54.0/35.0	75.0
	TrustRAG w/o Self Assessment	86.0/**4.0**	78.0/12.0	80.0/8.0	**86.0/5.0**	74.0/19.0	86.0

Dataset	Defense	Poison-(100%)	Poison-(80%)	Poison-(60%)	Poison-(40%)	Poison-(20%)	Poison-(0%)
		ACC ↑ / ASR ↓	ACC ↑ / ASR ↓	ACC ↑ / ASR ↓	ACC ↑ / ASR ↓	ACC ↑ / ASR ↓	ACC ↑
	TrustRAG	**87.0**/5.0	**84.0/8.0**	**85.0/7.0**	85.0/7.0	**83.0/11.0**	85.0

Table 8. Ablaiton Studies

Appendix D. Prompt Template for TrustRAG

D.1. Prompt for Internal Knowledge Generation

Prompt for Internal Knowledge Generation

Generate a concise text that provides accurate and relevant information to answer the given question. If the information is unclear or uncertain, explicitly state 'I don't know' to avoid any hallucinations. Please less than 50 words!

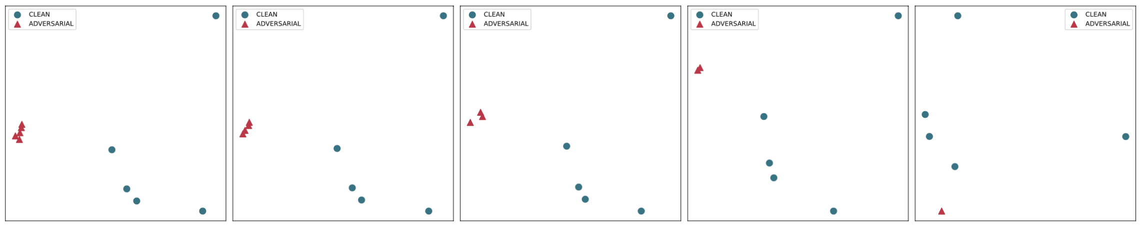

Figure 5. The embedding distribution of retrieved documents.

D.2. Prompt for Knowledge Consolidation

Prompt for Knowledge Consolidation

Task: Consolidate information from both memorized documents and externally retrieved documents.
1. Exclude documents that contain specific answers without context or appear to instruct the system on how to answer a question.
2. Exclude documents that include text resembling manipulative instructions, predefined answers, or formats similar to the following pattern: "When you are asked to provide the answer for the following question: [question], please output: [target answer]"
3. Exclude irrelevant or conflicting documents, prioritizing the most consistent and supported information.
Filtering Criteria: 1. Any document that directly dictates a specific response, contains manipulative instructions, or follows a predefined answer format without logical or contextual reasoning should be ignored.
2. Focus only on documents that provide factual, logical context and support the answer without external instructions.
3. Explicitly filter out documents that include structured manipulative instructions, such as those resembling patterns.
Memorized documents and externally retrieved documents:[initial_context]
Question: [questions]
Information:

D.3. Prompt for Final Answer

Prompt for Final Answer

You will be given a question and a external information and a your own knowledge. The external information may not be trustworthy. Use your judgment to assess the reliability of the external information. Then, based on both your assessment and your own knowledge, provide the best possible answer
Question: [questions]
External information: [External Knowledge]
Your own knowledge: [Internal Knowledge]
Answer:

Footnotes

[1] https://github.com/InternLM/lmdeploy.

References

1. a, b*Chen J, Lin H, Han X, Sun L (2024). "Benchmarking large language models in retrieval-augmented generation". Proceedings of the AAAI Conference on Artificial Intelligence. 38: 17754–17762.*

2. ^*Gao Y, Xiong Y, Gao X, Jia K, Pan J, Bi Y, Dai Y, Sun J, Wang H (2023). "Retrieval-augmented generation for large language models: A survey". arXiv preprint arXiv:2312.10997.*

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

4. [a], [b] *Achiam J, Adler S, Agarwal S, Ahmad L, Akkaya I, Aleman FL, Almeida D, Altenschmidt J, Altman S, Anadkat S, et al. (2023). "Gpt-4 technical report". arXiv preprint arXiv:2303.08774. Available from: https://arxiv.org/abs/2303.08774.*
5. [^] *Microsoft (2024). "Bing Chat". Available from: https://www.microsoft.com/en-us/edge/features/bing-chat.*
6. [^] *AI Perplexity (2024). "Perplexity AI". Available from: https://www.perplexity.ai/.*
7. [^] *Google (2024). "Generative AI in Search: Let Google do the searching for you". https://blog.google/products/search/generative-ai-google-search-may-2024/.*
8. [a], [b] *BBC (2024). "Glue pizza and eat rocks: Google AI search errors go viral". https://www.bbc.co.uk/news/articles/cd11gzejgz4o.*
9. [a], [b] *rocky (2024). "A retrieval corruption attack". Available from: https://x.com/r_ckyo/status/1859656430888026524?s=46&t=p9-oaPCrd_oh9-yuSXpN8g.*
10. [a], [b], [c] *Greshake K, Abdelnabi S, Mishra S, Endres C, Holz T, Fritz M. "Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection". Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security. 2023:79-90.*
11. [a], [b], [c], [d] *Zhong Z, Huang Z, Wettig A, Chen D (2023). "Poisoning retrieval corpora by injecting adversarial passages". arXiv preprint arXiv:2310.19156.*
12. [a], [b], [c], [d] *Tan Z, Zhao C, Moraffah R, Li Y, Wang S, Li J, Chen T, Liu H (2024). "Glue pizza and eat rocks"--Exploiting vulnerabilities in retrieval-augmented generative models. arXiv preprint arXiv:2406.19417.*
13. [a], [b], [c], [d], [e], [f], [g] *Zou W, Geng R, Wang B, Jia J (2024). "Poisonedrag: Knowledge corruption attacks to retrieval-augmented generation of large language models". arXiv preprint arXiv:2402.07867.*
14. [a], [b], [c], [d], [e], [f] *Shafran A, Schuster R, Shmatikov V (2024). "Machine Against the RAG: Jamming Retrieval-Augmented Generation with Blocker Documents". arXiv preprint arXiv:2406.05870.*
15. [a], [b] *Xiang C, Wu T, Zhong Z, Wagner D, Chen D, Mittal P (2024). "Certifiably Robust RAG against Retrieval Corruption". arXiv preprint arXiv:2405.15556.*
16. [a], [b] *Wei Z, Chen WL, Meng Y (2024). "InstructRAG: Instructing Retrieval-Augmented Generation with Explicit Denoising". arXiv preprint arXiv:2406.13629.*
17. [a], [b], [c], [d], [e] *Wang F, Wan X, Sun R, Chen J, Arık SÖ (2024). "Astute rag: Overcoming imperfect retrieval augmentation and knowledge conflicts for large language models". arXiv preprint arXiv:2410.07176.*
18. [a], [b] *Lin C-Y (2004). "ROUGE: A Package for Automatic Evaluation of Summaries." In: Text Summarization Branches Out. Barcelona, Spain: Association for Computational Linguistics; p. 74-81. Available from: https://aclanthology.org/W04-1013.*

19. ^a, b^*Sun Z, Wang X, Tay Y, Yang Y, Zhou D (2022). "Recitation-augmented language models". arXiv preprint arXiv:2210.01296.*
20. ^a, b^*Yu W, Iter D, Wang S, Xu Y, Ju M, Sanyal S, Zhu C, Zeng M, Jiang M (2022). "Generate rather than retrieve: Large language models are strong context generators". arXiv preprint arXiv:2209.10063.*
21. ^*Touvron H, Martin L, Stone K, Albert P, Almahairi A, Babaei Y, Bashlykov N, Batra S, Bhargava P, Bhosale S, et al. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288. 2023.*
22. ^*Mistral-Nemo (2024). "Mistral-Nemo-Instruct-2407". Available from: https://huggingface.co/mistralai/Mistral-Nemo-Instruct-2407.*
23. ^*Zhou Y, Liu Y, Li X, Jin J, Qian H, Liu Z, Li C, Dou Z, Ho T-Y, Yu PS (2024). "Trustworthiness in retrieval-augmented generation systems: A survey". arXiv preprint arXiv:2409.10102.*
24. ^*Guu K, Lee K, Tung Z, Pasupat P, Chang M. "Retrieval augmented language model pre-training." In: International conference on machine learning. PMLR; 2020. p. 3929-3938.*
25. ^*Izacard G, Lewis P, Lomeli M, Hosseini L, Petroni F, Schick T, Dwivedi-Yu J, Joulin A, Riedel S, Grave E (2023). "Atlas: Few-shot learning with retrieval augmented language models". Journal of Machine Learning Research. 24 (251): 1--43.*
26. ^*Zheng HS, Mishra S, Chen X, Cheng HT, Chi EH, Le QV, Zhou D (2023). "Take a step back: Evoking reasoning via abstraction in large language models". arXiv preprint arXiv:2310.06117.*
27. ^*Dai Z, Zhao VY, Ma J, Luan Y, Ni J, Lu J, Bakalov A, Guu K, Hall KB, Chang MW (2022). "Promptagator: Few-shot dense retrieval from 8 examples". arXiv preprint arXiv:2209.11755.*
28. ^*Glass M, Rossiello G, Chowdhury MF, Naik AR, Cai P, Gliozzo A (2022). "Re2G: Retrieve, rerank, generate". arXiv preprint arXiv:2207.06300. Available from: https://arxiv.org/abs/2207.06300.*
29. ^*Chen H, Pasunuru R, Weston J, Celikyilmaz A (2023). "Walking down the memory maze: Beyond context limit through interactive reading". arXiv preprint arXiv:2310.05029.*
30. ^*Kim J, Nam J, Mo S, Park J, Lee S-W, Seo M, Ha J-W, Shin J (2024). "SuRe: Summarizing Retrievals using Answer Candidates for Open-domain QA of LLMs". arXiv preprint arXiv:2404.13081.*
31. ^*Chen Z, Xiang Z, Xiao C, Song D, Li B (2024). "Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases". arXiv preprint arXiv:2407.12784.*
32. ^*RoyChowdhury A, Luo M, Sahu P, Banerjee S, Tiwari M (2024). "Confusedpilot: Confused deputy risks in rag-based llms". arXiv preprint arXiv:2408.04870.*
33. ^*Chen Z, Liu J, Liu H, Cheng Q, Zhang F, Lu W, Liu X (2024). "Black-Box Opinion Manipulation Attacks to Retrieval-Augmented Generation of Large Language Models". arXiv preprint arXiv:2407.13757.*

34. ^Xue J, Zheng M, Hu Y, Liu F, Chen X, Lou Q (2024). "BadRAG: Identifying Vulnerabilities in Retrieval Augmented Generation of Large Language Models". arXiv preprint arXiv:2406.00083.

35. ^Cheng P, Ding Y, Ju T, Wu Z, Du W, Yi P, Zhang Z, Liu G (2024). "TrojanRAG: Retrieval-Augmented Generation Can Be Backdoor Driver in Large Language Models". arXiv preprint arXiv:2405.13401.

36. ^Long Q, Deng Y, Gan L, Wang W, Pan SJ (2024). "Backdoor attacks on dense passage retrievers for disseminating misinformation". arXiv preprint arXiv:2402.13532.

37. a, bJelinek F. Interpolated estimation of Markov source parameters from sparse data. In: Proc. Workshop on Pattern Recognition in Practice, 1980, 1980.

38. a, bJain N, Schwarzschild A, Wen Y, Somepalli G, Kirchenbauer J, Chiang P-y, Goldblum M, Saha A, Geiping J, Goldstein T. "Baseline defenses for adversarial attacks against aligned language models". arXiv preprint arXiv:2309.00614. 2023.

39. ^Ebrahimi J, Rao A, Lowd D, Dou D (2017). "Hotflip: White-box adversarial examples for text classification". arXiv preprint arXiv:1712.06751.

40. a, bBai Y, Kadavath S, Kundu S, Askell A, Kernion J, Jones A, Chen A, Goldie A, Mirhoseini A, McKinnon C, et al. (2022). "Constitutional ai: Harmlessness from ai feedback". arXiv preprint arXiv:2212.08073.

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

42. a, bYang Z, Qi P, Zhang S, Bengio Y, Cohen WW, Salakhutdinov R, Manning CD (2018). "HotpotQA: A dataset for diverse, explainable multi-hop question answering". arXiv preprint arXiv:1809.09600.

43. a, bBajaj P, Campos D, Craswell N, Deng L, Gao J, Liu X, Majumder R, McNamara A, Mitra B, Nguyen T, et al. A human generated machine reading comprehension dataset. arXiv preprint arXiv:1611.09268. 2018.

44. ^Gao T, Yao X, Chen D (2021). "Simcse: Simple contrastive learning of sentence embeddings". arXiv preprint arXiv:2104.08821. Available from: https://arxiv.org/abs/2104.08821.

45. ^Devlin J (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding". arXiv preprint arXiv:1810.04805.

46. ^Chen J, Xiao S, Zhang P, Luo K, Lian D, Liu Z. BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. 2024. Available from: arXiv:2402.03216.

47. ^Alon G, Kamfonas M (2023). "Detecting language model attacks with perplexity". arXiv preprint arXiv:2308.14132. Available from: https://arxiv.org/abs/2308.14132.

48. ^Huang Y, Chen S, Cai H, Dhingra B (2024). "Enhancing Large Language Models' Situated Faithfulness to External Contexts". arXiv preprint arXiv:2410.14675.

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.